

```
        return oddCount;
    }

private static int getEvenCount(long num) {
    int evenCount = 0;

    for (char ch : Long.toString(num).toCharArray()) {
        if (Character.digit(ch, 10) % 2 == 0) {
            ++evenCount;
        }
    }

    return evenCount;
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.println("Enter the number");
    long num = scanner.nextLong();

    System.out.println("Sum of digits");
    System.out.println(getSum(num));

    System.out.println("Numerology number");
    System.out.println(getNumerology(num));

    System.out.println("Number of odd numbers");
    System.out.println(getOddCount(num));

    System.out.println("Number of even numbers");
    System.out.println(getEvenCount(num));
}
```

## OIL STORES

### Oil.java\*

```
import java.util.Scanner;

public class Oil{

    //Fill the code here

    private String name;
    private int pack;
    private char category;
    private float cost;

    public Oil(String name,int pack,char category,float cost){

        this.name=name;
        this.pack=pack;
        this.category=category;
        this.cost=cost;
    }

    public void setName(String name){

        this.name=name;
    }

    public String getName(){

        return name;
    }

    public void setPack(int pack){

        this.pack=pack;
    }

    public int getPack(){

        return pack;
    }

    public void setCategory(char category){

        this.category=category;
    }

    public char getCategory(){

        return category;
    }
}
```

```
}

public void setCost(float cost){

    this.cost=cost;
}

public float getCost(){

    return cost;
}

public float calculateTotalCost(float qty){

    float price=((qty*1000)/pack)*cost;

    return price;
}

}
```

### Main.java\*

```
import java.util.Scanner;

public class Main{

    public static void main (String[] args) {

        Scanner sc=new Scanner(System.in);

        //Fill the code

        System.out.println("Enter oil name");

        String n=sc.next();

        System.out.println("Enter pack capacity");

        int pc=sc.nextInt();

        System.out.println("Enter category");

        char cat=sc.next().charAt(0);

        System.out.println("Enter cost");

        float c=sc.nextFloat();

        Oil obj=new Oil(n,pc,cat,c);

        obj.setName(n);

        obj.setPack(pc);

        obj.setCategory(cat);
    }
}
```

```
obj.setCost(c);
System.out.println("Enter Quantity to purchase");
float qty=sc.nextFloat();
System.out.println("Oil cost rs."+obj.calculateTotalCost(qty));
}
}
```

## PAYMENT-INHERITENCE

**Bill.java\***

```
public class Bill {

    public String processPayment(Payment payObj) {
        String result="Payment not done and your due amount is "+payObj.getDueAmount();
        // Fill your code
        if(payObj instanceof Cheque)
        {
            Cheque cheque=(Cheque)payObj;
            if(cheque.payAmount())
                result="Payment done successfully via cheque";
        }
        else if(payObj instanceof Cash)
        {
            Cash cash=(Cash)payObj;
            if(cash.payAmount())
                result="Payment done successfully via cash";
        }
        else if(payObj instanceof Credit)
        {
            Credit credit=(Credit)payObj;
            if(credit.payAmount())
                result="Payment done successfully via credit card. Remaining amount in your "+credit.getCardType()+" card
is "+credit.getCreditCardAmount();
        }
        return result;
    }
}
```

**Cash.java\***

```
public class Cash extends Payment
{
    private int cashAmount;

    public int getCashAmount() {
        return cashAmount;
    }

    public void setCashAmount(int cashAmount) {
        this.cashAmount = cashAmount;
    }

    public boolean payAmount()
    {
        return getCashAmount() >= getDueAmount();
    }
}
```

**Cheque.java**

```
import java.util.*;
import java.util.GregorianCalendar;
import java.text.ParseException;
import java.util.Calendar;
import java.util.Date;
import java.text.SimpleDateFormat;

public class Cheque extends Payment
{
    private String chequeNo;
    private int chequeAmount;
    private Date dateOfIssue;

    public String getChequeNo() {
```

```

        return chequeNo;
    }

    public void setChequeNo(String chequeNo) {
        this.chequeNo = chequeNo;
    }

    public int getChequeAmount() {
        return chequeAmount;
    }

    public void setChequeAmount(int chequeAmount) {
        this.chequeAmount = chequeAmount;
    }

    public Date getDateOfIssue() {
        return dateOfIssue;
    }

    public void setDateOfIssue(Date dateOfIssue) {
        this.dateOfIssue = dateOfIssue;
    }

    private int findDifference(Date date){
        Calendar myDate=new GregorianCalendar();
        myDate.setTime(date);
        return (2020-myDate.get(Calendar.YEAR))*12+(0-myDate.get(Calendar.MONTH));
    }

    @Override
    public boolean payAmount()
    {
        int months=findDifference(getDateOfIssue());
        return (getChequeAmount()>=getDueAmount()&months<=6);
    }

// Fill your code

    public void generateDate(String date)
    {
        try{
            Date issueDate=new SimpleDateFormat("dd-MM-yyyy").parse(date);

```

```
        setDateOfIssue(issueDate);

    }

    catch(ParseException e)
    {
        e.printStackTrace();
    }

}

}
```

### Credit.java\*

```
public class Credit extends Payment
{
    private int creditCardNo;

    private String cardType; //(silver,gold,platinum) String,
    private int creditCardAmount;

    public int getCreditCardNo() {
        return creditCardNo;
    }

    public void setCreditCardNo(int creditCardNo) {
        this.creditCardNo = creditCardNo;
    }

    public String getCardType() {
        return cardType;
    }

    public void setCardType(String cardType) {
        this.cardType = cardType;
    }

    public int getCreditCardAmount() {
        return creditCardAmount;
    }

    public void setCreditCardAmount(int creditCardAmount) {
```

```
        this.creditCardAmount = creditCardAmount;  
    }  
  
    public Credit(int creditCardNo, String cardType) {  
        super();  
  
        // Fill your code  
  
    }  
  
    public Credit()  
    {  
  
    }  
  
    // Fill your code  
  
    @Override  
  
    public boolean payAmount()  
    {  
        int tax=0;  
  
        boolean isDeducted=false;  
  
        switch(cardType)  
        {  
            case "silver": setCreditCardAmount(10000);  
                tax=(int)(0.02*getDueAmount())+getDueAmount();  
                if(tax<=getCreditCardAmount())  
                {  
                    setCreditCardAmount(getCreditCardAmount()-tax);  
                    isDeducted=true;  
                }  
                break;  
  
            case "gold": setCreditCardAmount(50000);  
                tax=(int)(0.05*getDueAmount())+getDueAmount();  
                if(tax<=getCreditCardAmount())  
        }
```

```

    {
        setCreditCardAmount(getCreditCardAmount()-tax);
        isDeducted=true;
    }
    break;

case "platinum": setCreditCardAmount(100000);
    tax=(int)(0.1*getDueAmount())+getDueAmount();
    if(tax<=getCreditCardAmount())
    {
        setCreditCardAmount(getCreditCardAmount()-tax);
        isDeducted=true;
    }
    break;
}

return isDeducted;
}
}

```

### **Payment.java\***

```

public class Payment {

    private int dueAmount;

    public int getDueAmount() {
        return dueAmount;
    }

    public void setDueAmount(int dueamount) {
        this.dueAmount = dueamount;
    }

    public boolean payAmount() {

        return false;
    }
}

```

**Main.java\***

```
import java.util.*;  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        // Fill your code  
        Bill bill=new Bill();  
        Scanner sc=new Scanner(System.in);  
        System.out.println("Enter the due amount:");  
        int dueAmount=sc.nextInt();  
        System.out.println("Enter the mode of payment(cheque/cash/credit):");  
        String mode=sc.next();  
        switch(mode){  
            case "cash" : System.out.println("Enter the cash amount:");  
                int cashAmount=sc.nextInt();  
                Cash cash=new Cash();  
                cash.setCashAmount(cashAmount);  
                cash.setDueAmount(dueAmount);  
                System.out.println(bill.processPayment(cash));  
                break;  
            case "cheque" : System.out.println("Enter the cheque number:");  
                String number=sc.next();  
                System.out.println("Enter the cheque amount:");  
                int chequeAmount=sc.nextInt();  
                System.out.println("Enter the date of issue:");  
                String date=sc.next();  
                Cheque cheque=new Cheque();  
                cheque.setChequeAmount(chequeAmount);  
                cheque.setChequeNo(number);  
                cheque.generateDate(date);  
                cheque.setDueAmount(dueAmount);  
                System.out.println(bill.processPayment(cheque));  
        }  
    }  
}
```

```
        break;

    case "credit" : System.out.println("Enter the credit card number:");

        int creditNumber = sc.nextInt();

        System.out.println("Enter the card type(silver,gold,platinum):");

        String cardType=sc.next();

        Credit credit=new Credit();

        credit.setCardType(cardType);

        credit.setCreditCardNo(creditNumber);

        credit.setDueAmount(dueAmount);

        System.out.println(bill.processPayment(credit));

        break;

    default:

        break;

    }

    sc.close();

}

}

}
```

## POWER PROGRESS

### Main.java\*

```
import java.util.*;  
  
public class Main  
{  
  
    public static void main(String[] args)  
    {  
  
        Scanner sc=new Scanner(System.in);  
  
        //Fill the code  
  
        int m=sc.nextInt();  
  
        if(m<=0){  
  
            System.out.println(""+m+" is an invalid");  
  
            return;  
  
        }  
  
        int n=sc.nextInt();  
  
        if(n<=0){  
  
            System.out.println(""+n+" is an invalid");  
  
            return;  
  
        }  
  
        if(m>=n){  
  
            System.out.println(""+m+" is not less than "+n);  
  
            return;  
  
        }  
  
        for(int i=1;i<=n;i++){  
  
            System.out.print((int)Math.pow(m,i)+" ");  
  
        }  
  
    }  
}
```

## PRIME NUMBERS ENDING WITH 1

### Main.java\*

```
import java.util.Scanner;

public class Main {

    public static void main(String args[]) {

        Scanner sc = new Scanner(System.in);

        // fill the code

        int low, high;
        int last=0;
        int flag = 0;

        System.out.println("Enter the first number");
        low = sc.nextInt();

        System.out.println("Enter the last number");
        high = sc.nextInt();

        if (low > high || low < 0 || high < 0 || low == high)
            ;
        else {
            int i = low;
            high = high + 30;

            while (i <= high-1) {

                int x = i % 10;

                for (int j = 2; j <= i / 2; j++) {
                    if (i % j != 0 && x == 1) {
                        flag = 1;
                    } else {
                        flag = 0;
                        break;
                    }
                }

                if (flag == 1 )
                    System.out.println(i);
                i++;
            }
        }
    }
}
```

## SINGAPORE TOURISM

### Main.java\*

```
import java.util.*;  
  
public class Main  
{  
    public static void main(String[] args)  
    {  
        Scanner sc=new Scanner(System.in);  
        Map<String,Integer> map=new HashMap<>();  
        map.put("BEACH",270);  
        map.put("PILGRIMAGE",350);  
        map.put("HERITAGE",430);  
        map.put("HILLS",780);  
        map.put("FALLS",1200);  
        map.put("ADVENTURES",4500);  
        System.out.println("Enter the Passenger Name");  
        String pname=sc.next();  
        System.out.println("Enter the Place");  
        String name=sc.next();  
        if(!map.containsKey(name.toUpperCase()))  
        {  
            System.out.println(name+" is an invalid place");  
        }  
        else  
        {  
            System.out.println("Enter the no of Days");  
            int nod=sc.nextInt();  
            if(nod<=0)  
            {  
                System.out.println(nod+" is an invalid days");  
            }  
            else  
            {  
                System.out.println("Enter the no of Tickets");  
            }  
        }  
    }  
}
```

```
int not=sc.nextInt();
if(not<=0)
{
    System.out.println(not+" is an invalid tickets");
}
else
{
    double d=(double)map.get(name.toUpperCase());
    double totalcost=d*(double)not*(double)nod;
    if(totalcost>=1000)
    {
        totalcost=totalcost-((totalcost*15)/100);
    }
    System.out.printf("Bill Amount is %.2f", totalcost);
}

}

//Fill the code
}
```

## SUBSTITUTION CYPHER TECHNIQUE

### Main.java\*

```
import java.util.Scanner;

public class Main {

    public static void main (String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the Encrypted text: ");
        String code= sc.nextLine();
        int shift = 7;
        int f=0;

        String decryptMessage = " ";
        for(int i = 0; i < code.length(); i++){
            char alpha = code.charAt(i);
            if(alpha >='a' && alpha <= 'z'){
                f=1;
                alpha=(char)(alpha - shift);
                if(alpha < 'a'){
                    alpha = (char)(alpha - 'a'+'z'+1);
                }
                decryptMessage=decryptMessage+alpha;
            }
            else if (alpha >='A' && alpha <= 'Z'){
                f=1;
                alpha=(char)(alpha - shift);
                if(alpha < 'A'){
                    alpha = (char)(alpha - 'A'+'Z'+1);
                }
                decryptMessage=decryptMessage+alpha;
            }
            else if(alpha == ' '){
                decryptMessage=decryptMessage+alpha;
            }
        }
    }
}
```

```
    }

}

if(decryptMessage.length() == 0 || f == 0){

    System.out.println("No hidden Message");

    System.exit(0);

}

System.out.println("Decrpted Text:\n"+decryptMessage);

}

}
```

## ZEE ZEE BANK

### Account.java\*

```
public class Account {  
    long accountNumber;  
    double balanceAmount;  
  
    public Account(long accno, double bal){  
        super();  
        this.accountNumber=accno;  
        this.balanceAmount=bal;  
    }  
    public long getAccountNumber(){  
        return accountNumber;  
    }  
    public void setAccountNumber(long accno){  
        this.accountNumber=accno;  
    }  
    public double getBalanceAmount(){  
        return balanceAmount;  
    }  
    public void setBalanceAmount(double bal) {  
        this.balanceAmount=bal;  
    }  
    public void deposit(double depositAmt){  
        float total=(float)(balanceAmount+depositAmt);  
        balanceAmount=total;  
    }  
    public boolean withdraw(double withdrawAmt){  
        float total;  
        if(withdrawAmt>balanceAmount){  
            System.out.println("Insufficient balance");  
  
            return false;  
        }  
    }  
}
```

```
 }else{
    total=(float)(balanceAmount-withdrawAmt);
    setBalanceAmount(total);
    return true;
}
}
```

### Main.java\*

```
import java.util.Scanner;

public class Main{
    static Account ac=new Account(0, 0);
    public static void main (String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the account number:");
        ac.setAccountNumber(sc.nextLong());
        System.out.println("Enter the available amount in the account:");
        ac.setBalanceAmount(sc.nextDouble());
        System.out.println("Enter the amount to be deposited:");
        ac.deposit(sc.nextDouble());
        System.out.printf("Available balance is:%.2f",ac.getBalanceAmount());
        System.out.println();
        System.out.println("Enter the amount to be withdrawn:");
        ac.withdraw(sc.nextDouble());
        System.out.printf("Available balance is:%.2f",ac.getBalanceAmount());
        //Fill the code
    }
}
```

## THE NEXT RECHARGE DATE

### Main.java\*

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Main {

    public static void main(String [] args)throws Exception {
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Recharged date");
        String date=br.readLine();
        String currentDate="29/10/2019";
        if(Main.isValidFormat(date)&&(Main.dateCompare(date,currentDate))){
            System.out.println("Validity days");
            int days=Integer.parseInt(br.readLine());
            if(days>0)
                System.out.println(Main.futureDate(date,days));
            else
                System.out.println(days+ "is not a valid days");
        }
        else
            System.out.println(date+ "is not a valid date");
    }

    public static boolean isValidFormat(String date){
        String regex="^(3[01]|12[0-9]|0[1-9])/([1-2]|0[1-9])/[0-9]{4}$";
        Pattern pattern=Pattern.compile(regex);
        Matcher matcher=pattern.matcher((CharSequence)date);
        return matcher.matches();
    }
}
```

```
}

public static boolean dateCompare(String date1,String date2) throws ParseException{
    SimpleDateFormat sdfformat=new SimpleDateFormat("dd/MM/yyyy");
    Date d1=sdfformat.parse(date1);
    Date d2=sdfformat.parse(date2);
    if((d1.compareTo(d2)<0)|| (d1.compareTo(d2)==0))
        return true;
    else
        return false;
}

public static String futureDate(String date,int days){
    Calendar c=Calendar.getInstance();
    SimpleDateFormat sdfformat=new SimpleDateFormat("dd/MM/yyyy");
    try{
        Date mydate=sdfformat.parse(date);
        c.setTime(mydate);
        c.add(Calendar.DATE, days);
    }catch(ParseException e){
        e.printStackTrace();
    }
    String toDate=sdfformat.format(c.getTime());
    return toDate;
}
}
```

## A New You Spa

### DiamondMembers.java\*

```
public class DiamondMembers extends Members{

    // Fill the code

    public DiamondMembers(String customerId,String customerName,long mobileNumber,String memberType,String emailId){

        super(customerId,customerName,numberOfMembers,memberType,emailId);

        /*this.customerId = customerId;
        this.customerName = customerName;
        this.mobileNumber = mobileNumber;
        this.memberType = memberType;
        this.emailId = emailId;*/

    }

    public boolean validateCustomerId(){

        // Fill the code

        boolean b=true;

        String s1 = this.customerId.toUpperCase();

        String regex="[DIAMOND]{7}[0-9]{3}";

        if(s1.matches(regex)){

            b=true;

        }

        else{

            b=false;

        }

        return b;

    }

    public double calculateDiscount(double purchaseAmount){

        // Fill the code

    }

}
```

```
        double discount=purchaseAmount*0.45;
        double updateamount=purchaseAmount-discount;
        return updateamount;
    }

}
```

### **GoldMembers.java\***

```
public class GoldMembers extends Members {
    public GoldMembers(String customerId,String customerName,long mobileNumber,String memberType,String emailId){
        super(customerId,customerName,mobileNumber,memberType,emailId);
    }
    // Fill the code
    public boolean validateCustomerId(){
        boolean b=true;
        String s1 = this.customerId.toUpperCase();
        String regex="[GOLD]{4}[0-9]{3}";
        if(s1.matches(regex)){
            b=true;
        }
        else{
            b=false;
        }
        return b;
    }
    // Fill the code
}
public double calculateDiscount(double purchaseAmount){
    // Fill the code
    double discount=purchaseAmount*0.15;
    double updateamount=purchaseAmount-discount;
    return updateamount;
}
```

## **Members.java\***

```
abstract public class Members {  
    protected String customerId;  
    protected String customerName;  
    protected long mobileNumber;  
    protected String memberType;  
    protected String emailId;  
  
    abstract public double calculateDiscount(double purchaseAmount);  
  
    public String getCustomerId() {  
        return customerId;  
    }  
    public void setCustomerId(String customerId) {  
        this.customerId = customerId;  
    }  
    public String getCustomerName() {  
        return customerName;  
    }  
    public void setCustomerName(String customerName) {  
        this.customerName = customerName;  
    }  
    public long getMobileNumber() {  
        return mobileNumber;  
    }  
    public void setMobileNumber(long mobileNumber) {  
        this.mobileNumber = mobileNumber;  
    }  
  
    public String getMemberType() {  
        return memberType;  
    }  
    public void setMemberType(String memberType) {
```

```

        this.memberType = memberType;
    }

    public String getEmailId() {
        return emailId;
    }

    public void setEmailId(String emailId) {
        this.emailId = emailId;
    }

    public Members(String customerId, String customerName, long mobileNumber, String memberType, String
emailId) {
        this.customerId = customerId;
        this.customerName = customerName;
        this.mobileNumber = mobileNumber;
        this.memberType = memberType;
        this.emailId = emailId;
    }

}

```

### **PlatinumMembers.java\***

```

public class PlatinumMembers extends Members {

    // Fill the code

    public PlatinumMembers(String customerId, String customerName, long mobileNumber, String
memberType, String emailId) {
        super(customerId, customerName, mobileNumber, memberType, emailId);
        /*customerId = customerId;
        customerName = customerName;
        mobileNumber = mobileNumber;
        memberType = memberType;
        emailId = emailId;
    */

}

```

```

public boolean validateCustomerId(){
    // Fill the code
    boolean b=true;
    String s1 = this.customerId.toUpperCase();
    String regex="[PLATINUM]{8}[0-9]{3}";
    if(s1.matches(regex)){
        b=true;
    }
    else{
        b=false;
    }

    return b;
}

public double calculateDiscount(double purchaseAmount){
    // Fill the code
    double discount=purchaseAmount*0.3;
    double updateamount=purchaseAmount-discount;
    return updateamount;
}

}

```

### **UserInterface.java\***

```

import java.util.Scanner;

public class UserInterface {

    public static void main(String[] args){

        Scanner sc=new Scanner(System.in);
        System.out.println("Enter Customer Id");
    }
}

```

```

String cid=sc.nextLine();
System.out.println("Enter Customer name");
String cname=sc.nextLine();
System.out.println("Enter mobile number");
long mob=sc.nextLong();
sc.nextLine();
System.out.println("Enter Member type");
String mem=sc.nextLine();
System.out.println("Enter Email Id");
String email=sc.nextLine();
System.out.println("Enter amount Purchased");
double amount=sc.nextDouble();
DiamondMembers d=new DiamondMembers(cid,cname,mob,mem,email);
GoldMembers g=new GoldMembers(cid,cname,mob,mem,email);
PlatinumMembers p=new PlatinumMembers(cid,cname,mob,mem,email);

double res=0.0;
if(d.validateCustomerId()){
    res= d.calculateDiscount(amount);
    System.out.println("Name :" +d.getCustomerName());
    System.out.println("Id :" +d.getCustomerId());
    System.out.println("Email Id :" +d.getEmailId());
    System.out.println("Amount to be paid :" +res);
}

} else if(g.validateCustomerId()){
    res= g.calculateDiscount(amount);
    System.out.println("Name :" +g.getCustomerName());
    System.out.println("Id :" +g.getCustomerId());
    System.out.println("Email Id :" +g.getEmailId());
    System.out.println("Amount to be paid :" +res);
}

} else if(p.validateCustomerId()){
    res= p.calculateDiscount(amount);
}

```

```
System.out.println("Name :" + p.getCustomerName());  
System.out.println("Id :" + p.getCustomerId());  
System.out.println("Email Id :" + p.getEmailId());  
System.out.println("Amount to be paid :" + res);  
  
} else{  
    System.out.println("Provide a valid Customer Id");  
}  
  
// Fill the code  
}
```

## BATTING AVERAGE

### UserInterface.java\*

```
package com.ui;

import com.utility.Player;

import java.util.ArrayList;

import java.util.Scanner;

public class UserInterface {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        Player player=new Player();

        player.setScoreList(new ArrayList<>());

        boolean flag=true;

        while(flag)

        {

            System.out.println("1. Add Runs Scored");

            System.out.println("2. Calculate average runs scored");

            System.out.println("3. Exit");

            System.out.println("Enter your choice");

            int choice=sc.nextInt();

            switch(choice)

            {

                case 1:

                    System.out.println("Enter the runs scored");

                    int runScored=sc.nextInt();

                    player.addScoreDetails(runScored);

                    break;

                }

                case 2:

                    System.out.println("Average runs secured");

                    System.out.println(player.getAverageRunScored());

                    break;

            }

        }

    }

}
```

```
        case 3: {
            System.out.println("Thank you for using the Application");
            flag=false;
            break;
        }
    }
}

}
```

### **Player.java\***

```
package com.utility;

import java.util.List;

public class Player {

    private List<Integer> scoreList;

    public List<Integer> getScoreList() {
        return scoreList;
    }

    public void setScoreList(List<Integer> scoreList) {
        this.scoreList = scoreList;
    }

    //This method should add the runScored passed as the argument into the scoreList
    public void addScoreDetails(int runScored) {
```

```
// fill the code  
scoreList.add(runScored);  
  
}  
  
}
```

/\* This method should return the average runs scored by the player

Average runs can be calculated based on the sum of all runScored available in the scoreList divided by the number of elements in the scoreList.

For Example:

```
List contains[150,50,50]  
average runs secured=(150+50+50)/3=83.3333333333333  
so this method should return 83.3333333333333  
If list is empty return 0  
*/  
  
public double getAverageRunScored() {  
  
    // fill the code  
    if(scoreList.isEmpty()) {  
        return 0.0;  
    }  
    int size=scoreList.size();  
    int totalScore=0;  
    for(int score : scoreList)  
    {  
        totalScore+=score;  
    }  
    return (double) totalScore / (double) size;  
}  
  
}
```

## Change The Cash

### Main.java\*

```
import java.util.*;  
  
public class Main{  
  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        String a = sc.next();  
        if(a.length() < 3) {  
            System.out.println("String length of " + a + " is too short");  
            return;  
        }  
        else if(a.length() > 10) {  
            System.out.println("String length of " + a + " is too long");  
            return;  
        }  
  
        char[] arr = a.toCharArray();  
        char[] arr1 = new char[arr.length];  
        int j = 0;  
        for(int i = 0; i < a.length(); i++) {  
            if((arr[i]<65 || ((arr[i]>90) && (arr[i]<97)) || arr[i]>122)) {  
                arr1[j++] = arr[i];  
            }  
        }  
        if(j!=0) {  
            System.out.print("String should not contain ");  
            for(int i = 0; i<j; i++) {  
                System.out.print(arr1[i]);  
            }  
            return;  
        }  
        char b = sc.next().charAt(0);  
    }  
}
```

```
int present = 0;
for(int i = 0; i<a.length(); i++) {
    if(arr[i] == Character.toUpperCase(b)) {
        arr[i] = Character.toLowerCase(b);
        present = 1;
    }
}
if(present == 0) {
    System.out.println("Character " + b + " is not found");
}
else {
    for(int i = 0; i <a.length(); i++) {
        System.out.print(arr[i]);
    }
}
}
```

## Check Number Type

### NumberType.java\*

```
public interface NumberType
{
    public boolean checkNumberType(int n);
}
```

### NumberTypeUtility.java\*

```
import java.util.Scanner;

public class NumberTypeUtility
{
    public static NumberType isOdd()
    {
        return n -> n%2 != 0;
    }

    public static void main (String[] args)
    {

        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        if(isOdd().checkNumberType(n))
        {
            System.out.println(n+" is odd");
        }
        else
        {
            System.out.println(n+" is not odd");
        }
    }
}
```

## Cheque Payment Process

### PaymentDao.java\*

```
package com.cts.paymentProcess.dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import com.cts.paymentProcess.model.Payment;
import com.cts.paymentProcess.util.DatabaseUtil;

public class PaymentDao {

    private Connection connection;

    public List<Payment> getAllRecord(){

        connection=DatabaseUtil.getConnection();
        PreparedStatement statement=null;
        ResultSet resultSet=null;
        List<Payment> paymentList=new ArrayList<Payment>();
        try {
            statement=connection.prepareStatement("select * from cheque_payments");
            resultSet=statement.executeQuery();
            while(resultSet.next()){
                Payment payment =new Payment();
                payment.setCustomerNumber(resultSet.getInt("customerNumber"));
                payment.setChequeNumber(resultSet.getString("chequeNumber"));
                payment.setPaymentDate(resultSet.getDate("paymentDate"));
                payment.setAmount(resultSet.getInt("amount"));
                paymentList.add(payment);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return paymentList;
    }
}
```

```
        }

    } catch (SQLException e) {

        e.printStackTrace();

    }finally{

        try{

            resultSet.close();

            statement.close();

        }catch(Exception e){

            e.printStackTrace();

        }

    }

}

return paymentList;

}
```

#### **Payment.java\***

```
package com.cts.paymentProcess.model;

import java.util.Date;

public class Payment {

    private int customerNumber;

    private String chequeNumber;

    private Date paymentDate;

    private int amount;

    public int getCustomerNumber() {

        return customerNumber;

    }

}
```

```
public void setCustomerNumber(int customerNumber) {
    this.customerNumber = customerNumber;
}

public String getChequeNumber() {
    return chequeNumber;
}

public void setChequeNumber(String chequeNumber) {
    this.chequeNumber = chequeNumber;
}

public Date getPaymentDate() {
    return paymentDate;
}

public void setPaymentDate(Date paymentDate) {
    this.paymentDate = paymentDate;
}

public int getAmount() {
    return amount;
}

public void setAmount(int amount) {
    this.amount = amount;
}

@Override
public String toString() {

    return String.format("%15s%15s%15s%15s", customerNumber, chequeNumber, paymentDate,
amount);
}

}
```

**PaymentService.java\***

```
package com.cts.paymentProcess.service;

import java.util.*;
import java.util.Calendar;
import java.util.List;
import java.util.stream.Collectors;

import com.cts.paymentProcess.dao.PaymentDao;
import com.cts.paymentProcess.model.Payment;

public class PaymentService {

    private PaymentDao paymentDao=new PaymentDao();

    public List<Payment> findCustomerByNumber(int customerNumber){

        List<Payment> list=paymentDao.getAllRecord();

        List<Payment> list2 = new ArrayList<>();
        list2 = list.stream().filter(x->x.getCustomerNumber()==customerNumber).collect(Collectors.toList());

        return list2;
    }

    public List<Payment> findCustomerByYear(int year){

        List<Payment> list=paymentDao.getAllRecord();

        List<Payment> list2 = new ArrayList<>();
        list2 = list.stream().filter(x->x.getPaymentDate().getYear()==(year-1900)).sorted(Comparator.comparingInt(Payment::getAmount)).collect(Collectors.toList());

        return list2;
    }
}
```

**SkeletonValidator.java\***

```
package com.cts.paymentProcess.skeletonValidator;

import java.lang.reflect.Method;
import java.util.logging.Level;
import java.util.logging.Logger;

public class SkeletonValidator {

    public SkeletonValidator(){

        validateClassName("com.cts.paymentProcess.dao.PaymentDao");
        validateMethodSignature("getAllRecord:java.util.List","com.cts.paymentProcess.dao.PaymentDao");

        validateClassName("com.cts.paymentProcess.model.Payment");
        validateMethodSignature("toString:java.lang.String","com.cts.paymentProcess.model.Payment");

        validateClassName("com.cts.paymentProcess.service.PaymentService");

        validateMethodSignature("findCustomerByNumber:java.util.List,findCustomerByYear:java.util.List","com.cts.
paymentProcess.service.PaymentService");

        validateClassName("com.cts.paymentProcess.util.DatabaseUtil");

        validateMethodSignature("getConnection:java.sql.Connection","com.cts.paymentProcess.util.DatabaseUtil")
    ;

    }

    private static final Logger LOG = Logger.getLogger("SkeletonValidator");

    protected final boolean validateClassName(String className) {
```

```

boolean iscorrect = false;

try {
    Class.forName(className);
    iscorrect = true;
    LOG.info("Class Name " + className + " is correct");

} catch (ClassNotFoundException e) {
    LOG.log(Level.SEVERE, "You have changed either the " + "class name/package. Use the
correct package "
        + "and class name as provided in the skeleton");

} catch (Exception e) {
    LOG.log(Level.SEVERE,
        "There is an error in validating the " + "Class Name. Please manually verify
that the "
        + "Class name is same as skeleton before uploading");
}

return iscorrect;

}

protected final void validateMethodSignature(String methodWithExcptn, String className) {
    Class cls = null;
    try {

        String[] actualmethods = methodWithExcptn.split(",");
        boolean errorFlag = false;
        String[] methodSignature;
        String methodName = null;
        String returnType = null;

        for (String singleMethod : actualmethods) {
            boolean foundMethod = false;
            methodSignature = singleMethod.split(":");

```

```

methodName = methodSignature[0];
returnType = methodSignature[1];
cls = Class.forName(className);
Method[] methods = cls.getMethods();
for (Method findMethod : methods) {
    if (methodName.equals(findMethod.getName())) {
        foundMethod = true;
        if (!(findMethod.getReturnType().getName().equals(returnType))) {
            errorFlag = true;
            LOG.log(Level.SEVERE, " You have changed the " + "return
type in '" + methodName
                                         + "' method. Please stick to the " + "skeleton
provided");
    }
}
} else {
    LOG.info("Method signature of " + methodName + " is
valid");
}
}

if (!foundMethod) {
    errorFlag = true;
    LOG.log(Level.SEVERE, " Unable to find the given public method " +
methodName
                                         + ". Do not change the " + "given public method name. " +
"Verify it with the skeleton");
}
}

if (!errorFlag) {
    LOG.info("Method signature is valid");
}
}

} catch (Exception e) {
    LOG.log(Level.SEVERE,

```

```
" There is an error in validating the " + "method structure. Please manually  
verify that the "  
+ "Method signature is same as the skeleton before  
uploading");  
}  
}  
}
```

### **DatabaseUtil.java\***

```
package com.cts.paymentProcess.util;  
  
import java.io.FileInputStream;  
import java.io.IOException;  
import java.io.InputStream;  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.SQLException;  
import java.util.Properties;  
import java.util.ResourceBundle;  
  
public class DatabaseUtil {  
  
    private DatabaseUtil() {  
    }  
  
    private static Connection con = null;  
    private static Properties props = new Properties();  
  
    //ENSURE YOU DON'T CHANGE THE BELOW CODE WHEN YOU SUBMIT  
    public static Connection getConnection() {  
        try{
```

```

FileInputStream fis = null;
fis = new FileInputStream("resource/database.properties");
props.load(fis);

// load the Driver Class
try {
    Class.forName(props.getProperty("DB_DRIVER_CLASS"));
} catch (ClassNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

// create the connection now
try {
    con =
DriverManager.getConnection(props.getProperty("DB_URL"),props.getProperty("DB_USERNAME"),props.getProperty("DB_PASSWORD"));
} catch (SQLException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}

catch(IOException e){
    e.printStackTrace();
}

return con;
}

}

```

**App.java(Main)\***

```
package com.cts.paymentProcess;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.List;
import java.util.Scanner;

import com.cts.paymentProcess.model.Payment;
import com.cts.paymentProcess.service.PaymentService;
import com.cts.paymentProcess.skeletonValidator.SkeletonValidator;

public class App {

    public static void main(String[] args) throws ParseException {

        new SkeletonValidator();

        Payment payment=null;
        Scanner scanner=new Scanner(System.in);

        do{
            System.out.println("Select Option:");
            System.out.println("1.Customer list\n2.Yearly Customer List\n3.Exit");
            int choice=scanner.nextInt();
            switch(choice){

                case 1:System.out.println("Enter customer number");
                int number=scanner.nextInt();
                List<Payment> numberList=new PaymentService().findCustomerByNumber(number);
                if(numberList.size()==0){
                    System.out.println("\nNo Records Found\n");
                }else{


```

```
System.out.format("%15s%15s%15s%15s\n","Customer Number","Cheque Number","Payment Date","Amount");

    numberList.stream()

        .forEach(System.out::println);

    }

    break;

    case 2:System.out.println("Enter year");

    int year=scanner.nextInt();

    List<Payment> yearList=new PaymentService().findCustomerByYear(year);

    if(yearList.size()==0){

        System.out.println("\nNo Records Found\n");

    }else{

        System.out.format("%15s%15s%15s%15s\n","Customer Number","Cheque Number","Payment Date","Amount");

        yearList.stream()

            .forEach(System.out::println);

        }

        break;

        case 3:System.exit(0);

        default:System.out.println("\nWrong Choice\n");

    }

}

}

}

}
```

## Employee Eligibility for Promotion

### Main.java\*

```
import java.time.format.DateTimeFormatter;

import java.time.temporal.ChronoUnit;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.time.*;

import java.util.TreeMap;
import java.util.Date;
import java.util.Map;

import java.util.Map.Entry;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        //System.out.println("In-time");
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
        LocalDate currTime = LocalDate.of(2019, 01, 01);
        String fdt=currTime.format(formatter);
        //
        System.out.println(fdt);
        int no = sc.nextInt();
        Map<String, String> m = new TreeMap<>();
        for (int i = 1; i <= no; i++) {
            String id = sc.next();
            String date = sc.next();
            m.put(id, date);
        }
    }
}
```

```

int count = 0;
int val = 0;

for (Entry<String, String> entry : m.entrySet()) {
    if (entry.getValue().matches("(0[1-9]|1[1-2][0-9]|3[0-1])/"
        "(0[1-9]|1[0-2])/[0-9]{4}"))
    {
        val++;
        LocalDate InTime = LocalDate.parse(entry.getValue(), formatter);
        Period in = Period.between(InTime, currTime);
        long lin = in.get(ChronoUnit.YEARS);
        if (lin >= 5)
        {
            count++;
            System.out.println(entry.getKey());
        }
    }
    else
    {
        System.out.println("Invalid date format");
        break;
    }
}

if (count == 0 && val == no)
    System.out.println("No one is eligible");
}
}

```

## Exam Scheduler

### AssessmentDao.java\*

```
package com.cts.cc.dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.time.Duration;
import java.time.LocalDate;
import java.time.LocalTime;
import java.time.Period;
import java.util.List;
import java.sql.*;

import com.cts.cc.model.Assessment;
import com.cts.cc.util.DatabaseUtil;

public class AssessmentDAO {

    public int uploadAssessments(List<Assessment> assessments) throws Exception {
        if(assessments==null || assessments.isEmpty()) {
            throw new Exception("List is Empty");
        }
        int rowsCount = 0;

        //Write your logic here...

        try{
            Connection con = DatabaseUtil.getConnection();
            for(Assessment a:assessments)
            {
                PreparedStatement st = con.prepareStatement("insert into assessment values(?,?,?,?,?,?)");
                st.setString(1,a.getExamCode());
                st.setString(2,a.getExamTitle());
                st.setString(3,a.getExamDate().toString());
                st.setString(4,a.getExamTime().toString());
            }
        }
    }
}
```

```

        st.setString(5,a.getExamDuration().toString());
        st.setString(6,a.getEvalDays().toString());
        int rs=st.executeUpdate();
        if(rs!=-1)
            rowsCount=rowsCount+1;
    }

} catch(SQLException e){

}

return rowsCount;
}

public Assessment findAssessment(String code) throws Exception {
    Assessment assessment = null;
    Connection conn = DatabaseUtil.getConnection();
    String sql = "SELECT * FROM assessment where code=?";
    PreparedStatement ps = conn.prepareStatement(sql);
    ps.setString(1, code);
    ResultSet rs = ps.executeQuery();
    if(rs.next()) {
        assessment = new Assessment();
        assessment.setExamCode(rs.getString(1));
        assessment.setExamTitle(rs.getString(2));
        assessment.setExamDate(LocalDate.parse(rs.getString(3)));
        assessment.setExamTime(LocalTime.parse(rs.getString(4)));
        assessment.setExamDuration(Duration.parse(rs.getString(5)));
        assessment.setEvalDays(Period.parse(rs.getString(6)));
    }

    return assessment;
}
}

```

**GenerateAssessmentFunction.java\***

```
package com.cts.cc.functions;

import java.time.Duration;
import java.time.LocalDate;
import java.time.LocalTime;
import java.time.Period;
import java.util.function.Function;

import com.cts.cc.model.Assessment;

public class GenerateAssessmentFunction implements Function<String, Assessment>{

    @Override
    public Assessment apply(String t) {
        //Write your code here...
        String temp[] = t.split(",");
        Assessment a = new
        Assessment(temp[0],temp[1],LocalDate.parse(temp[2]),LocalTime.parse(temp[3]),Duration.parse(temp[4]),Period.p
        arse(temp[5]));
        return a;
    }
}
```

**Assessment.java\***

```
package com.cts.cc.model;

import java.time.Duration;
import java.time.LocalDate;
import java.time.LocalTime;
import java.time.Period;
import java.time.format.DateTimeFormatter;

public class Assessment {
```

```
private String examCode;
private String examTitle;
private LocalDate examDate;
private LocalTime examTime;
private Duration examDuration;
private Period evalDays;

public Assessment(String examCode, String examTitle, LocalDate examDate, LocalTime examTime, Duration
examDuration,
                    Period evalDays) {
    super();
    this.examCode = examCode;
    this.examTitle = examTitle;
    this.examDate = examDate;
    this.examTime = examTime;
    this.examDuration = examDuration;
    this.evalDays = evalDays;
}

public Assessment() {
}

public String getExamCode() {
    return examCode;
}

public void setExamCode(String examCode) {
    this.examCode = examCode;
}

public String getExamTitle() {
    return examTitle;
}
```

```
public void setExamTitle(String examTitle) {
    this.examTitle = examTitle;
}

public LocalDate getExamDate() {
    return examDate;
}

public void setExamDate(LocalDate examDate) {
    this.examDate = examDate;
}

public LocalTime getExamTime() {
    return examTime;
}

public void setExamTime(LocalTime examTime) {
    this.examTime = examTime;
}

public Duration getExamDuration() {
    return examDuration;
}

public void setExamDuration(Duration examDuration) {
    this.examDuration = examDuration;
}

public Period getEvalDays() {
    return evalDays;
}

public void setEvalDays(Period evalDays) {
    this.evalDays = evalDays;
}
```

```
}

public void printDetails() {
    DateTimeFormatter date1=DateTimeFormatter.ofPattern("dd-MMM-y");
    DateTimeFormatter date2=DateTimeFormatter.ofPattern("HH:mm");
    LocalTime t=examTime.plus(examDuration);
    String d=DateTimeFormatter.ofPattern("HH:mm").format(t);
    LocalDate t1=examDate.plus(evalDays);
    String d1=DateTimeFormatter.ofPattern("dd-MMM-y").format(t1);
    System.out.println("Assessment Code: "+examCode);
    System.out.println("Title: "+examTitle);
    System.out.println("Assessment Date: "+examDate.format(date1));
    System.out.println("Start Time: "+examTime.format(date2));
    System.out.println("End Time: "+d);
    System.out.println("Result Date: "+d1);
    //Write your code here...
}
```

```
}
```

#### **DatabaseUtil.java\***

```
package com.cts.cc.util;

import java.io.FileInputStream;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;

public class DatabaseUtil {

    private static Connection con = null;
```

```

private static Properties props = new Properties();

//ENSURE YOU DON'T CHANGE THE BELOW CODE WHEN YOU SUBMIT

public static Connection getConnection() throws ClassNotFoundException, SQLException {
    try{
        FileInputStream fis = null;
        fis = new FileInputStream("resource/connection.properties");
        props.load(fis);

        // load the Driver Class
        Class.forName(props.getProperty("DB_DRIVER_CLASS"));

        // create the connection now
        con =
        DriverManager.getConnection(props.getProperty("DB_URL"),props.getProperty("DB_USERNAME"),props.getProperty("DB_PASSWORD"));
    }
    catch(IOException e){
        e.printStackTrace();
    }
    return con;
}
}

```

### **FileUtil.java\***

```

package com.cts.cc.util;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;
import java.util.function.Function;

```

```

import java.util.stream.Collectors;
import java.util.stream.Stream;
import java.io.*;
import java.util.*;

import com.cts.cc.functions.GenerateAssessmentFunction;
import com.cts.cc.model.Assessment;

public class FileUtils {

    public static List<Assessment> loadData(String fileName) throws IOException {
        List<Assessment> list = null;
        Function<String, Assessment> function = new GenerateAssessmentFunction();
        BufferedReader br=new BufferedReader(new FileReader(fileName));
        String line="";
        list=new ArrayList<Assessment>();
        while((line=br.readLine())!=null)
        {
            list.add(function.apply(line));
        }
        //Write your code here...
    }

    return list;
}
}

```

### **SkeletonValidator.java\***

```

package com.cts.cc;

import java.lang.reflect.Constructor;
import java.lang.reflect.Method;
import java.sql.Connection;
import java.time.Duration;
import java.time.LocalDate;
import java.time.LocalTime;

```

```
import java.time.Period;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

import com.cts.cc.model.Assessment;

public class SkeletonValidator {
    private static final Logger LOG = Logger.getLogger("SkeletonValidator");

    public SkeletonValidator() {
        String assessmentClass = "com.cts.cc.model.Assessment";
        String assessmentDAOClass = "com.cts.cc.dao.AssessmentDAO";
        String functionalClass = "com.cts.cc.functions.GenerateAssessmentFunction";
        String databaseUtilClass = "com.cts.cc.util.DatabaseUtil";
        String fileUtilClass = "com.cts.cc.util.FileUtil";

        Class[] assessmentParams = { String.class, String.class, LocalDate.class, LocalTime.class,
Duration.class,
                                         Period.class };

        String[] assessmentFields = { "examCode", "examTitle", "examDate", "examTime", "examDuration",
"evalDays" };

        testClass(assessmentClass, assessmentParams);
        testClass(assessmentDAOClass, null);
        testClass(functionalClass, null);
        testClass(databaseUtilClass, null);
        testClass(fileUtilClass, null);

        testFields(assessmentClass, assessmentFields);
        testMethods(assessmentClass, "printDetails", null, null);
        testMethods(assessmentDAOClass, "uploadAssessments", new Class[] { List.class }, boolean.class);
        testMethods(assessmentDAOClass, "findAssessment", new Class[] { String.class }, Assessment.class);
        testMethods(functionalClass, "apply", new Class[] { String.class }, Assessment.class);
    }
}
```

```

    testMethods(databaseUtilClass, "getConnection", null, Connection.class);
    testMethods(fileUtilClass, "loadData", new Class[] { String.class }, List.class);
}

public void testClass(String className, Class[] paramTypes) {
    try {
        Class classUnderTest = Class.forName(className);
        LOG.info("Class Name " + className + " is correct");
        Constructor<?> constructor = classUnderTest.getConstructor(paramTypes);
        constructor.equals(constructor);
        LOG.info(className + " Constructor is valid");
    } catch (ClassNotFoundException e) {
        LOG.log(Level.SEVERE, "You have changed either the class name/package. "
            + "Use the correct package and class name as provided in the skeleton");
    } catch (NoSuchMethodException e) {
        LOG.log(Level.SEVERE, " Unable to find the given constructor. "
            + "Do not change the given public constructor. " + "Verify it with the
skeleton");
    } catch (SecurityException e) {
        LOG.log(Level.SEVERE,
            "There is an error in validating the " + className + ". " + "Please verify the
skeleton manually");
    }
}

public void testFields(String className, String[] fields) {
    try {
        Class classUnderTest = Class.forName(className);
        for (String field : fields) {
            classUnderTest.getDeclaredField(field);
        }
        LOG.info("Fields in " + className + " are correct");
    } catch (ClassNotFoundException e) {
        LOG.log(Level.SEVERE, "You have changed either the class name/package. "
            + "Use the correct package and class name as provided in the skeleton");
    }
}

```

```

} catch (NoSuchFieldException e) {
    LOG.log(Level.SEVERE,
        "You have changed one/more field(s). " + "Use the field name(s) as provided
in the skeleton");

} catch (SecurityException e) {
    LOG.log(Level.SEVERE,
        "There is an error in validating the " + className + ". " + "Please verify the
skeleton manually");
}

}

public void testMethods(String className, String methodName, Class[] paramTypes, Class returnType) {
    try {
        Class classUnderTest = Class.forName(className);
        Method method = classUnderTest.getDeclaredMethod(methodName, paramTypes);
        Class retType = method.getReturnType();
        if (returnType != null && !retType.equals(returnType)) {
            LOG.log(Level.SEVERE, " You have changed the " + "return type in '" + methodName
                + "' method. Please stick to the " + "skeleton provided");
            throw new NoSuchMethodException();
        }
        LOG.info(methodName + " signature is valid.");
    } catch (ClassNotFoundException e) {
        LOG.log(Level.SEVERE, "You have changed either the class name/package. "
            + "Use the correct package and class name as provided in the skeleton");
    } catch (NoSuchMethodException e) {
        LOG.log(Level.SEVERE, "You have changed/removed method " + methodName + ". "
            + "Use the method signature as provided in the skeleton");
    } catch (SecurityException e) {
        LOG.log(Level.SEVERE,
            "There is an error in validating the " + className + ". " + "Please verify the
skeleton manually");
    }
}

```

**Main.java\***

```
package com.cts.cc;

import java.util.List;

import com.cts.cc.dao.AssessmentDAO;
import com.cts.cc.model.Assessment;
import com.cts.cc.util.FileUtil;

public class Main {

    public static void main(String[] args) {
        // CODE SKELETON - VALIDATION STARTS
        // DO NOT CHANGE THIS CODE
        new SkeletonValidator();
        // CODE SKELETON - VALIDATION ENDS

        try {
            List<Assessment> assessments = FileUtil.loadData("resource/data.txt");
            AssessmentDAO dao = new AssessmentDAO();
            dao.uploadAssessments(assessments);
            Assessment assessment = dao.findAssessment("ASEJE025");
            assessment.printDetails();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

## Book Details

### Main.java\*

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        //fill the code
        String string = sc.nextLine();
        if(string.length()==18)
        {
            String substr1 = string.substring(0,3);
            int i1 = Integer.parseInt(substr1);
            if(i1>100 && i1<104)
            {
                String substr2 = string.substring(3,7);
                int i2 = Integer.parseInt(substr2);
                if(i2>=1900 && i2<=2020)
                {
                    String substr3 = string.substring(7,12);
                    //System.out.println(substr3);
                    int i3 = Integer.parseInt(substr3);
                    if(i3>=10)
                    {
                        String substr4 = string.substring(12,13);
                        String substr5 = string.substring(13,18);

                        if((substr4.charAt(0)>='A'&&substr4.charAt(0)<='Z')|| (substr4.charAt(0)>='a'&&substr4.charAt(0)<='z'))
                        {

                            if((substr5.charAt(0)>='0'&&substr5.charAt(0)<='9')&&(substr5.charAt(1)>='0'&&substr5.charAt(1)<='9')&&
                            (substr5.charAt(2)>='0'&&substr5.charAt(2)<='9')&&(substr5.charAt(3)>='0'&&substr5.charAt(3)<='9')&&
                            (substr5.charAt(4)>='0'&&substr5.charAt(4)<='9'))
                            {

```

```
String substr6 = string.substring(12,18);

System.out.println("Department Code: "+substr1);

if(i1==101)

System.out.println("Department Name: "+"Accounting");

else if(i1==102)

System.out.println("Department Name: "+"Economics");

else if(i1==103)

System.out.println("Department Name: "+"Engineering");

System.out.println("Year of Publication: "+substr2);

System.out.println("Number of Pages: "+substr3);

System.out.println("Book Id: "+substr6);

}

else

{

    String substr6 = string.substring(12,18);

    System.out.printf("%s is invalid book id",substr6);

    System.out.printf("\n");

}

else

{

    String substr6 = string.substring(12,18);

    System.out.printf("%s is invalid book id",substr6);

    System.out.printf("\n");

}

else

{

    System.out.printf("%s are invalid pages",substr3);

    System.out.printf("\n");

}
```

```
    }

    else

    {

        System.out.printf("%d is invalid year",i2);

        System.out.printf("\n");

    }

}

else

{

    System.out.printf("%d is invalid department code",i1);

    System.out.printf("\n");

}

}

else

{

    System.out.printf("%s is invalid input",string);

    System.out.printf("\n");

}

}

}
```

## Find Membership Category

**Member.java\***

```
public class Member {  
  
    private String memberId;  
    private String memberName;  
    private String category;  
  
    public String getMemberId() {  
        return memberId;  
    }  
    public void setMemberId(String memberId) {  
        this.memberId = memberId;  
    }  
    public String getMemberName() {  
        return memberName;  
    }  
    public void setMemberName(String memberName) {  
        this.memberName = memberName;  
    }  
    public String getCategory() {  
        return category;  
    }  
    public void setCategory(String category) {  
        this.category = category;  
    }  
  
    public Member(String memberId, String memberName, String category) {  
        super();  
        this.memberId = memberId;  
        this.memberName = memberName;  
        this.category = category;  
    }  
}
```

```
}
```

### **ZeeShop.java\***

```
import java.util.List;

public class ZEEShop extends Thread
{
    private String memberCategory;
    private int count;
    private List<Member> memberList;

    public ZEEShop(String memberCategory, List<Member> memberList) {
        super();
        this.memberCategory = memberCategory;
        this.memberList = memberList;
    }

    public String getMemberCategory() {
        return memberCategory;
    }

    public void setMemberCategory(String memberCategory) {
        this.memberCategory = memberCategory;
    }
}
```

```
public int getCount() {
    return count;
}

public void setCount(int count) {
    this.count = count;
}

public List<Member> getMemberList() {
    return memberList;
}

public void setMemberList(List<Member> memberList) {
    this.memberList = memberList;
}

public void run()
{
    synchronized(this)
    {
        for(Member m:memberList)
        {
            if(m.getCategory().equals(memberCategory))
                count++;
        }
    }
}
```

```
}
```

```
}
```

```
}
```

### Main.java\*

```
import java.util.ArrayList;  
import java.util.List;  
import java.util.Scanner;
```

```
public class Main {
```

```
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
  
        List<Member> mList=new ArrayList<Member>();  
        System.out.println("Enter the number of Members:");  
        Scanner sc=new Scanner(System.in);  
        int tot=sc.nextInt();  
        String[] str=new String[tot];  
        for(int i=0;i<str.length;i++)  
        {  
            System.out.println("Enter the Member Details:");  
            str[i]=sc.next();  
        }
```

```
        Member m[]=new Member[tot];  
        for(int i=0;i<m.length;i++)  
        {  
            String s[]=str[i].split(":");
```

```

m[i]=new Member(s[0],s[1],s[2]);
mList.add(m[i]);

}

System.out.println("Enter the number of times Membership category needs to be searched:");
int tot1=sc.nextInt();
ZEEShop t1[]=new ZEEShop[tot1];
for(int i=0;i<tot1;i++)
{
    System.out.println("Enter the Category");
    String s1=sc.next();
    t1[i]=new ZEEShop(s1,mList);
    t1[i].start();

    //System.out.println(s1+" "+t1.getCount());
}

try {
    Thread.currentThread().sleep(2000);
} catch (InterruptedException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

for(ZEEShop s:t1)
{
    System.out.println(s.getMemberCategory()+":"+s.getCount());
}

}

```

## Go Hospitals

### InPatient.java\*

```
public class InPatient extends Patient{  
  
    private double roomRent;  
  
    public InPatient(String patientId, String patientName, long mobileNumber, String gender, double roomRent) {  
        super(patientId,patientName,mobileNumber,gender);  
        this.roomRent=roomRent;  
    }  
  
    public double getRoomRent() {  
        return roomRent;  
    }  
  
    public void setRoomRent(double roomRent) {  
        this.roomRent = roomRent;  
    }  
  
    // Fill the code  
  
    public double calculateTotalBill(int noOfDays,double medicinalBill){  
        // Fill the code  
        double bill_amount;  
        bill_amount=((this.roomRent*noOfDays)+medicinalBill);  
        return bill_amount;  
    }  
}
```

### OutPatient.java\*

```
public class OutPatient extends Patient{
```

```

private double consultingFee;

public OutPatient(String patientId, String patientName, long mobileNumber, String gender, double consultingFee)
{
    super(patientId,patientName,mobileNumber,gender);
    this.consultingFee=consultingFee;
}

public double getConsultingFee() {
    return consultingFee;
}

public void setConsultingFee(double consultingFee) {
    this.consultingFee = consultingFee;
}

// Fill the code

public double calculateTotalBill(double scanPay,double medicinalBill){
    // Fill the code
    double bill_amount;
    bill_amount=this.consultingFee+scanPay+medicinalBill;
    return bill_amount;
}

```

### Patient.java\*

```

public class Patient {

    protected String patientId;
    protected String patientName;

```

```
protected long mobileNumber;
protected String gender;

public Patient(String patientId, String patientName, long mobileNumber, String gender) {
    this.patientId = patientId;
    this.patientName = patientName;
    this.mobileNumber = mobileNumber;
    this.gender = gender;
}

public String getPatientId() {
    return patientId;
}

public void setPatientId(String patientId) {
    this.patientId = patientId;
}

public String getPatientName() {
    return patientName;
}

public void setPatientName(String patientName) {
    this.patientName = patientName;
}

public long getMobileNumber() {
    return mobileNumber;
}

public void setMobileNumber(long mobileNumber) {
    this.mobileNumber = mobileNumber;
}

public String getGender() {
    return gender;
}

public void setGender(String gender) {
    this.gender = gender;
}
```

```
}
```

```
}
```

### **UserInterface.java\***

```
import java.util.Scanner;

public class UserInterface {

    public static void main(String[] args){
        Scanner read=new Scanner(System.in);

        System.out.println("1.In Patient");
        System.out.println("1.Out Patient");
        System.out.println("Enter the choice");
        int ch=read.nextInt();
        System.out.println("Enter the details");
        System.out.println("Patient Id");
        String id=read.nextLine();
        System.out.println("Patient Name");
        String name=read.nextLine();
        read.nextLine();
        System.out.println("Phone Number");
        long num=read.nextLong();
        System.out.println("Gender");
        String gen=read.next();

        if(ch==1){

            System.out.println("Room Rent");
            double rent=read.nextDouble();
            InPatient in=new InPatient(id,name,num,gen,rent);
            System.out.println("Medicinal Bill");
        }
    }
}
```

```
double bill=read.nextDouble();

System.out.println("Number of Days of Stay");

int days=read.nextInt();

System.out.println("Amount to be paid "+in.calculateTotalBill(days,bill));

}

else{

System.out.println("Consultancy Fee");

double fee=read.nextDouble();

OutPatient out=new OutPatient(id,name,num,gen,fee);

System.out.println("Medicinal Bill");

double medbill=read.nextDouble();

System.out.println("Scan Pay");

double pay=read.nextDouble();

System.out.println("Amount to be paid "+out.calculateTotalBill(pay,medbill));

}

// Fill the code

}
```

## Grade Calculation

### GradeCalculator.java\*

```
public class GradeCalculator extends Thread {
```

```
    private String studName;
```

```
    private char result;
```

```
    private int[] marks;
```

```
    public String getStudName(){
```

```
        return studName;
```

```
}
```

```
    public void setStudName(String studName){
```

```
        this.studName = studName;
```

```
}
```

```
    public char getResult(){
```

```
        return result;
```

```
}
```

```
    public void setResult(char result){
```

```
        this.result = result;
```

```
}
```

```
    public int[] getMarks(){
```

```
        return marks;
```

```
}
```

```
    public void setMarks(int[] marks){
```

```
        this.marks = marks;
```

```
}
```

```

public GradeCalculator(String studName, int[] marks){
    this.studName = studName;
    this.marks = marks;
}

public void run(){
    int sum = 0;
    int[] score = getMarks();
    for(int i = 0;i<score.length;i++)
        sum = sum+score[i];
    if((400<=sum)&&(sum<=500))
        System.out.println(getStudName()+":"+ 'A');
    if((300<=sum)&&(sum<=399))
        System.out.println(getStudName()+":"+ 'B');
    if((200<=sum)&&(sum<=299))
        System.out.println(getStudName()+":"+ 'C');
    if(sum<200)
        System.out.println(getStudName()+":"+ 'E');
}
}

```

### Main.java\*

```

import java.util.Scanner;
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class Main {

    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter the number of Threads:");
        int th = Integer.parseInt(br.readLine());
        GradeCalculator obj = null;
        String str = "";

```

```
String details[] = new String[th];
for(int i=0; i<th; i++){
    System.out.println("Enter the String:");
    str = br.readLine();
    details[i]=str;
}

for(int i=0; i<th; i++){
    String sp[] = details[i].split(":");
    int k = 0;
    int arr[] = new int[sp.length];
    for(int j = 1; j<sp.length; j++)
        arr[k++] = Integer.parseInt(sp[j]);
    obj = new GradeCalculator(sp[0],arr);
    obj.start();
    try{
        Thread.sleep(1000);
    }
    catch(Exception e)
    {
        System.out.println(e);
    }
}

}

}
```

## Passanger Amenity

### Main.java\*

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        int num,n,i,count1=0,count2=0,y;
        char alpha,ch;
        String n1,n2;

        Scanner sc = new Scanner(System.in);

        //fill the code

        System.out.println("Enter the number of passengers");

        n=sc.nextInt();

        if(n<=0){

            System.out.println(n+" is invalid input");
            System.exit(0);

        }

        String[] arr1=new String[n];
        String[] arr2=new String[n];

        for(i=0;i<n;i++,count1=0,count2=0){

            System.out.println("Enter the name of the passenger "+(i+1));
            arr1[i] =sc.next();

            System.out.println("Enter the seat details of the passenger "+(i+1));
            arr2[i]= sc.next();

            num =Integer.parseInt(arr2[i].substring(1,(arr2[i].length())));

            alpha= arr2[i].charAt(0);

            if(num>=10 && num<=99){

                count2++;

            }

            for(ch=65;ch<84;ch++){

                if(ch==alpha){

                    count1++;

                }

            }

            if(count1==0){


```

```

        System.out.println(""+alpha+" is invalid coach");
        System.exit(0);
    }

    if(count2==0){

        System.out.println(num+" is invalid seat number");
        System.exit(0);
    }

}

for(i=0;i<n;i++){

    for(int j=i+1;j<n;j++){

        if(arr2[i].charAt(0)==arr2[j].charAt(0)){

if((Integer.parseInt(arr2[i].substring(1,(arr2[i].length()))))<(Integer.parseInt(arr2[j].substring(1,arr2[j].length())))){



n1=arr1[i];
n2=arr2[i];
arr1[i]=arr1[j];
arr2[i]=arr2[j];
arr1[j]=n1;
arr2[j]=n2;

}
}
else
if(arr2[i].charAt(0)<arr2[j].charAt(0))

{
n1=arr1[i];
n2=arr2[i];
arr1[i]=arr1[j];
arr2[i]=arr2[j];
arr1[j]=n1;
arr2[j]=n2;

}
}
}
}

```

```
for(i=0;i<n;i++){  
    String a=arr1[i].toUpperCase();  
    String b=arr2[i];  
    System.out.print(a+" "+b);  
    System.out.println("");  
}  
}  
}
```

## Perform Calculation

### **Calculate.java\***

```
public interface Calculate {  
  
    float performCalculation(int a,int b);  
}
```

### **Calculator.java\***

```
import java.util.Scanner;
```

```
public class Calculator {
```

```
    public static void main (String[] args) {
```

```
        Scanner sc=new Scanner(System.in);
```

```
        int a = sc.nextInt();
```

```
        int b= sc.nextInt();
```

```
        Calculate Perform_addition = performAddition();
```

```
        Calculate Perform_subtraction = performSubtraction();
```

```
        Calculate Perform_product = performProduct();
```

```
        Calculate Perform_division = performDivision();
```

```
System.out.println("The sum is "+Perform_addition.performCalculation(a,b));  
  
System.out.println("The difference is "+Perform_subtraction.performCalculation(a,b));  
  
System.out.println("The product is "+Perform_product.performCalculation(a,b));  
  
System.out.println("The division value is "+Perform_division.performCalculation(a,b));  
  
}
```

```
public static Calculate performAddition(){  
  
    Calculate Perform_calculation = (int a,int b)->a+b;  
  
    return Perform_calculation;  
  
}
```

```
public static Calculate performSubtraction(){  
  
    Calculate Perform_calculation = (int a,int b)->a-b;  
  
    return Perform_calculation;  
  
}
```

```
public static Calculate performProduct(){  
  
    Calculate Perform_calculation = (int a,int b)->a*b;  
  
    return Perform_calculation;
```

```
}
```

```
public static Calculate performDivision(){
```

```
Calculate Perform_calculation = (int a,int b)->{
```

```
    float c = (float)a;
```

```
    float d = (float)b;
```

```
    return (c/d);
```

```
};
```

```
return Perform_calculation;
```

```
}
```

```
}
```

## Query DataSet

### Query.java\*

```
//Write the required business logic as expected in the question description
```

```
public class Query {  
  
    private String queryId;  
    private String queryCategory;  
    private DataSet primaryDataSet;  
    private DataSet secondaryDataSet;  
  
    @Override  
    public String toString()  
    {  
        String g="";  
        g+="Primary data set"+'\n';  
        g+="Theatre id : "+primaryDataSet.getTheatreId()+'\n';  
        g+="Theatre name : "+primaryDataSet.getTheatreName()+'\n';  
        g+="Location :" +primaryDataSet.getLocation()+'\n';  
        g+="No of Screen : "+primaryDataSet.getNoOfScreen()+'\n';  
        g+="Ticket Cost : "+primaryDataSet.getTicketCost()+'\n';  
  
        g+="Secondary data set"+'\n';  
        g+="Theatre id : "+secondaryDataSet.getTheatreId()+'\n';  
        g+="Theatre name : "+secondaryDataSet.getTheatreName()+'\n';  
        g+="Location :" +secondaryDataSet.getLocation()+'\n';  
        g+="No of Screen : "+secondaryDataSet.getNoOfScreen()+'\n';  
        g+="Ticket Cost : "+secondaryDataSet.getTicketCost()+'\n';  
  
        g+="Query id : "+queryId+'\n';  
        g+="Query category : "+queryCategory+'\n';  
  
        return g;  
    }
```

```
public class DataSet{  
    private String theatreId;  
    private String theatreName;  
    private String location;  
    private int noOfScreen;  
    private double ticketCost;  
  
    public double getTicketCost()  
    {  
        return ticketCost;  
    }  
  
    public void setTicketCost(double a)  
    {  
        ticketCost=a;  
    }  
  
    public int getNoOfScreen()  
    {  
        return noOfScreen;  
    }  
  
    public void setNoOfScreen(int a)  
    {  
        noOfScreen=a;  
    }  
    public String getLocation ()  
    {  
        return location;  
    }  
    public void setLocation(String a)  
    {  
        location=a;  
    }
```

```
}

public String getTheatreName ()
{
    return theatreName;
}

public void setTheatreName(String a)
{
    theatreName=a;
}

public String getTheatreId ()
{
    return theatreId;
}

public void setTheatreId(String a)
{
    theatreId=a;
}

public void setSecondaryDataSet(DataSet pD)
{
    this.secondaryDataSet=pD;
}

public DataSet getSecondaryDataSet()
{
    return this.secondaryDataSet;
}

public void setPrimaryDataSet(DataSet pD)
{
    this.primaryDataSet=pD;
}
```

```

public DataSet getPrimaryDataSet()
{
    return this.primaryDataSet;
}

public void setQueryId (String queryId)
{
    this.queryId=queryId;
}

public void setQueryCategory(String queryCategory)
{
    this.queryCategory=queryCategory;
}

public String getQueryId()
{
    return this.queryId;
}

public String getQueryCategory()
{
    return this.queryCategory;
}
}

```

### **TestApplication.java\***

```

import java.util.*;
public class TestApplication {
    //Write the required business logic as expected in the question description
    public static void main (String[] args) {
        Scanner sc= new Scanner (System.in);
        Query q= new Query();
        Query.DataSet pd= q.new DataSet();
        Query.DataSet sd= q.new DataSet();
        System.out.println("Enter the Details for primary data set");
        System.out.println("Enter the theatre id");
    }
}

```

```
pd.setTheatreId(sc.nextLine());
System.out.println("Enter the theatre name");
pd.setTheatreName(sc.nextLine());
System.out.println("Enter the location");
pd.setLocation(sc.nextLine());
System.out.println("Enter the no of screens");
pd.setNoOfScreen(sc.nextInt());
System.out.println("Enter the ticket cost");
pd.setTicketCost(sc.nextDouble());

System.out.println("Enter the Details for secondary data set");
System.out.println("Enter the theatre id");
String id2=sc.next();
// System.out.println(id2);
sd.setTheatreId(id2);
System.out.println("Enter the theatre name");
sc.nextLine();
sd.setTheatreName(sc.nextLine());
System.out.println("Enter the location");
String gll=sc.nextLine();
sd.setLocation(gll);
System.out.println("Enter the no of screens");

// System.out.println(gll);
// String pp=sc.nextLine();
// System.out.println(pp);
sd.setNoOfScreen(sc.nextInt());
System.out.println("Enter the ticket cost");
sd.setTicketCost(sc.nextDouble());
sc.nextLine();
System.out.println("Enter the query id");
q.setQueryId(sc.nextLine());
System.out.println("Enter the query category");
q.setQueryCategory(sc.nextLine());
```

```
q.setSecondaryDataSet(sd);
q.setPrimaryDataSet(pd);
System.out.println(q.toString());
}

}
```

## Retrieve Flight Based On Source And Destination

### Flight.java\*

```
public class Flight {  
  
    private int flightId;  
  
    private String source;  
  
    private String destination;  
  
    private int noOfSeats;  
  
    private double flightFare;  
  
    public int getFlightId() {  
        return flightId;  
    }  
  
    public void setFlightId(int flightId) {  
        this.flightId = flightId;  
    }  
  
    public String getSource() {  
        return source;  
    }  
  
    public void setSource(String source) {  
        this.source = source;  
    }  
  
    public String getDestination() {  
        return destination;  
    }  
  
    public void setDestination(String destination) {  
        this.destination = destination;  
    }  
  
    public int getNoOfSeats() {  
        return noOfSeats;  
    }  
  
    public void setNoOfSeats(int noOfSeats) {  
        this.noOfSeats = noOfSeats;  
    }  
}
```

```

public double getFlightFare() {
    return flightFare;
}

public void setFlightFare(double flightFare) {
    this.flightFare = flightFare;
}

public Flight(int flightId, String source, String destination,
        int noOfSeats, double flightFare) {
    super();
    this.flightId = flightId;
    this.source = source;
    this.destination = destination;
    this.noOfSeats = noOfSeats;
    this.flightFare = flightFare;
}

}

}

```

### **FlightManagement.java\***

```

import java.util.ArrayList;
import java.sql.*;

public class FlightManagementSystem {

    public ArrayList<Flight> viewFlightBySourceDestination(String source, String destination){
        ArrayList<Flight> flightList = new ArrayList<Flight>();
        try{
            Connection con = DB.getConnection();
            String query="SELECT * FROM flight WHERE source= '" + source + "' AND destination= '" + destination + "' ";

```

```

Statement st=con.createStatement();

ResultSet rst= st.executeQuery(query);

while(rst.next()){

    int flightId= rst.getInt(1);

    String src=rst.getString(2);

    String dst=rst.getString(3);

    int noofseats=rst.getInt(4);

    double flightfare=rst.getDouble(5);

    flightList.add(new Flight(flightId, src, dst, noofseats, flightfare));

}

}catch(ClassNotFoundException | SQLException e){

    e.printStackTrace();

}

return flightList;

}

}

```

### **DB.java\***

```

import java.io.FileInputStream;

import java.io.IOException;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.util.Properties;

public class DB {

    private static Connection con = null;

    private static Properties props = new Properties();

```

```

//ENSURE YOU DON'T CHANGE THE BELOW CODE WHEN YOU SUBMIT

public static Connection getConnection() throws ClassNotFoundException, SQLException {
    try{
        FileInputStream fis = null;
        fis = new FileInputStream("database.properties");
        props.load(fis);

        // load the Driver Class
        Class.forName(props.getProperty("DB_DRIVER_CLASS"));

        // create the connection now
        con =
        DriverManager.getConnection(props.getProperty("DB_URL"),props.getProperty("DB_USERNAME"),props.getProperty("DB_PASSWORD"));
    }
    catch(IOException e){
        e.printStackTrace();
    }
    return con;
}
}

```

### **Main.java\***

```

import java.util.Scanner;
import java.util.ArrayList;

public class Main{
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the source");
        String source=sc.next();
        System.out.println("Enter the destination");
        String destination=sc.next();
    }
}

```

```
FlightManagementSystem fms= new FlightManagementSystem();
ArrayList<Flight> flightList=fms.viewFlightBySourceDestination(source,destination);
if(flightList.isEmpty()){
    System.out.println("No flights available for the given source and destination");
    return;
}
System.out.println("Flightid Noofseats Flightfare");
for(Flight flight : flightList){
    System.out.println(flight.getFlightId()+" "+flight.getNoOfSeats()+" "+flight.getFlightFare());
}
}
```

## Silver Health Plan Insurance

### FamilyInsurancePolicy.java

```
public class FamilyInsurancePolicy extends InsurancePolicies{  
  
    public FamilyInsurancePolicy(String clientName, String policyId, int age, long mobileNumber, String emailId) {  
        super(clientName, policyId, age, mobileNumber, emailId);  
  
    }  
  
    public boolean validatePolicyId()  
    {  
        int count=0;  
        if(policyId.contains("FAMILY"));  
        count++;  
        char ch[]=policyId.toCharArray();  
        for(int i=6;i<9;i++)  
        {  
            if(ch[i]>='0' && ch[i]<='9')  
                count++;  
        }  
        if(count==4)  
            return true;  
        else  
            return false;  
    }  
    public double calculateInsuranceAmount(int months, int no_of_members)  
    {  
        double amount=0;  
        if(age>=5 && age<=25)  
            amount=2500*months*no_of_members;  
        else if (age>25 && age<60)  
            amount=5000*months*no_of_members;  
        else if (age>=60)  
            amount=10000*months*no_of_members;  
    }  
}
```

```
    return amount;  
}  
  
}
```

### **IndividualInsurancePolicy.java\***

```
public class IndividualInsurancePolicy extends InsurancePolicies{  
  
    public IndividualInsurancePolicy(String clientName, String policyId, int age, long mobileNumber, String  
emailId) {  
        super(clientName, policyId, age, mobileNumber, emailId);  
  
    }  
  
    public boolean validatePolicyId()  
    {  
        int count=0;  
        if(policyId.contains("SINGLE"));  
        count++;  
        char ch[]=policyId.toCharArray();  
        for(int i=6;i<9;i++)  
        {  
            if(ch[i]>='0' && ch[i]<='9')  
                count++;  
        }  
        if(count==4)  
            return true;  
        else  
            return false;  
    }  
  
    public double calculateInsuranceAmount(int months)  
    {  
        double amount=0;
```

```
if(age>=5 && age<=25)
    amount=2500*months;
else if (age>25 && age<60)
    amount=5000*months;
else if (age>=60)
    amount=10000*months;
return amount;
}

}
```

### **InsurancePolicies.java\***

```
public class InsurancePolicies {
    protected String clientName;
    protected String policyId;
    protected int age;
    protected long mobileNumber;
    protected String emailId;
    public String getClientName() {
        return clientName;
    }
    public void setClientName(String clientName) {
        this.clientName = clientName;
    }
    public String getPolicyId() {
        return policyId;
    }
    public void setPolicyId(String policyId) {
        this.policyId = policyId;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
```

```

        this.age = age;
    }

    public long getMobileNumber() {
        return mobileNumber;
    }

    public void setMobileNumber(long mobileNumber) {
        this.mobileNumber = mobileNumber;
    }

    public String getEmailId() {
        return emailId;
    }

    public void setEmailId(String emailId) {
        this.emailId = emailId;
    }

    public InsurancePolicies(String clientName, String policyId, int age, long mobileNumber, String emailId) {
        super();
        this.clientName = clientName;
        this.policyId = policyId;
        this.age = age;
        this.mobileNumber = mobileNumber;
        this.emailId = emailId;
    }
}

```

### **SeniorCitizenPolicy.java\***

```

public class SeniorCitizenPolicy extends InsurancePolicies{

    public SeniorCitizenPolicy(String clientName, String policyId, int age, long mobileNumber, String emailId) {
        super(clientName, policyId, age, mobileNumber, emailId);
    }

    public boolean validatePolicyId()
    {
        int count=0;
        if(policyId.contains("SENIOR"));

```

```

        count++;

        char ch[]=policyId.toCharArray();

        for(int i=6;i<9;i++)
        {

            if(ch[i]>='0' && ch[i]<='9')

                count++;

        }

        if(count==4)

            return true;

        else

            return false;

    }

}

public double calculateInsuranceAmount(int months, int no_of_members)

{

    double amount=0;

    if(age>=5 && age<60)

        amount=0;

    else if (age>=60)

        amount=10000*months*no_of_members;

    return amount;

}

}

```

### **UserInterface.java\***

```

import java.util.Scanner;

public class UserInterface {

    public static void main(String[] args)
    {

```

```

Scanner sc=new Scanner(System.in);

System.out.println("Enter Client name");

String name=sc.next();

System.out.println("Enter Policy Id");

String id=sc.next();

System.out.println("Enter Client age");

int age=sc.nextInt();

System.out.println("Enter mobile number");

long mnum=sc.nextLong();

System.out.println("Enter Email Id");

String email=sc.next();

InsurancePolicies policy=new InsurancePolicies(name,id,age,mnum,email);

System.out.println("Enter the months");

int month=sc.nextInt();

double amount=0;

if(id.contains("SINGLE"))

{

    IndividualInsurancePolicy g=new IndividualInsurancePolicy(name,id,age,mnum,email);

    if(g.validatePolicyId())

    {

        //System.out.println(g.validatePolicyId());

        amount=g.calculateInsuranceAmount(month);

        System.out.println("Name :" +name);

        System.out.println("Email Id :" +email);

        System.out.println("Amount to be paid :" +amount);

    }

    else

    {

        System.out.println("Provide valid Policy Id");

    }

}

```

```

else if(id.contains("FAMILY"))

{
    FamilyInsurancePolicy g=new FamilyInsurancePolicy(name,id,age,mnum,email);

    if(g.validatePolicyId())

    {

        System.out.println("Enter number of members");

        int num=sc.nextInt();

        amount=g.calculateInsuranceAmount(month,num);

        System.out.println("Name :" +name);

        System.out.println("Email Id :" +email);

        System.out.println("Amount to be paid :" +amount);

    }

    else

    {

        System.out.println("Provide valid Policy Id");

    }

}

else if(id.contains("SENIOR"))

{

    SeniorCitizenPolicy g=new SeniorCitizenPolicy(name,id,age,mnum,email);

    if(g.validatePolicyId())

    {

        System.out.println("Enter number of members");

        int num=sc.nextInt();

        amount=g.calculateInsuranceAmount(month,num);

        System.out.println("Name :" +name);

        System.out.println("Email Id :" +email);

        System.out.println("Amount to be paid :" +amount);

    }

    else

    {

        System.out.println("Provide valid Policy Id");

    }

}

```

```
else
    System.out.println("Provide valid Policy Id");
}

}
```

## Travel Request System

### TraveRequestDao.java\*

```
package com.cts.travelrequest.dao;

import java.io.FileInputStream;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;

import java.sql.*;
import java.util.*;
//import java.util.Properties;

import com.cts.travelrequest.vo.TravelRequest;

class DB{

    private static Connection con = null;
    private static Properties props = new Properties();

    //ENSURE YOU DON'T CHANGE THE BELOW CODE WHEN YOU SUBMIT

    public static Connection getConnection() throws ClassNotFoundException, SQLException {
        try{

            FileInputStream fis = null;
            fis = new FileInputStream("resource/database.properties");
            props.load(fis);

            // load the Driver Class
            Class.forName(props.getProperty("DB_DRIVER_CLASS"));

            // create the connection now
            con = DriverManager.getConnection(props.getProperty("URL"),props.getProperty("username"),props.getProperty("password"));
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }

    public static void closeConnection()
    {
        try{
            if(con != null)
                con.close();
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

```

        con =
DriverManager.getConnection(props.getProperty("DB_URL"),props.getProperty("DB_USERNAME"),props.getProperty("DB_PASSWORD"));

    }

    catch(IOException e){
        e.printStackTrace();
    }

    return con;
}

}

```

```

/**
 * Method to get travel details based on source and destination city      *
 * @return list
 */

public class TravelRequestDao{

    // public PreparedStatement prepareStatement(String query) throws SQLException{}

    public static List<TravelRequest> getTravelDetails(String sourceCity, String destinationCity) {

        List<TravelRequest> travel=new ArrayList<>();

        try{
            Connection con=DB.getConnection();

            String query="Select * from t_travelrequest where sourceCity=? and destinationCity=?;";

            PreparedStatement ps=con.prepareStatement(query);

            ps.setString(1,sourceCity);
            ps.setString(2,destinationCity);

            ResultSet rs=ps.executeQuery();

            while(rs.next()){

                String tid=rs.getString("travelReqId");
                java.sql.Date date=rs.getDate("travelDate");
                String apstat=rs.getString("approvalStatus");
                String sour=rs.getString("sourceCity");
                String des=rs.getString("destinationCity");
                double cost=rs.getDouble("travelCost");

                TravelRequest tr=new TravelRequest(tid,date,apstat,sour,des,cost);

```

```

        travel.add(tr);
    }
}

catch(ClassNotFoundException e){
    e.printStackTrace();
}

catch(SQLException e ){
    e.printStackTrace();
}

return travel; //TODO change this return value
}

/** 
 * Method to calculate total travel cost based approvalStatus
 * @return list
 */
public static double calculateTotalTravelCost(String approvalStatus) {
    double amount=0;
    try{
        Connection con=DB.getConnection();
        String query="select travelCost from t_travelrequest where approvalStatus=?;";
        PreparedStatement ps1=con.prepareStatement(query);
        ps1.setString(1,approvalStatus);
        ResultSet rs1=ps1.executeQuery();
        while(rs1.next()){
            amount+=rs1.getDouble("travelCost");
        }
    }
    catch(ClassNotFoundException e){
        e.printStackTrace();
    }
}

```

```
        catch(SQLException e){
            e.printStackTrace();
        }

        return amount; //TODO change this return value
    }
}
```

### **TravelRequestService.java\***

```
package com.cts.travelrequest.service;

import java.util.List;

import com.cts.travelrequest.dao.TravelRequestDao;
import com.cts.travelrequest.vo.TravelRequest;

public class TravelRequestService {

    /**
     * Method to validate approval status
     *
     * @return status
     */
    public String validateApprovalStatus(String approvalStatus) {
        if(approvalStatus.equalsIgnoreCase("Approved") | |approvalStatus.equalsIgnoreCase("Pending")){
            return "valid";
        }
        return "invalid"; //TODO change this return value
    }

    /**
     * Method to validate source and destination city
     *
     * @return status
     */
}
```

```

*/
public String validateSourceAndDestination(String sourceCity, String destinationCity) {
    if(!sourceCity.equalsIgnoreCase(destinationCity)){
        if(sourceCity.equalsIgnoreCase("Pune")|| sourceCity.equalsIgnoreCase("Mumbai")|||
        sourceCity.equalsIgnoreCase("Chennai")|| sourceCity.equalsIgnoreCase("Bangalore")|||
        sourceCity.equalsIgnoreCase("Hydrabad")){
            if(destinationCity.equalsIgnoreCase("Pune")|||
            destinationCity.equalsIgnoreCase("Mumbai")|||destinationCity.equalsIgnoreCase("Chennai")|||
            destinationCity.equalsIgnoreCase("Bangalore")||| destinationCity.equalsIgnoreCase("Hydrabad")){
                return "valid";
            }
        } else{
            return "invalid";
        }
    } else{
        return "invalid";
    }
}

/**
 * Method to invoke getTravelDetails method of TravelRequestDao class
 *
 * @return listOfTravelRequest
 */
public List<TravelRequest> getTravelDetails(String sourceCity, String destinationCity) {
    if(this.validateSourceAndDestination(sourceCity,destinationCity).contentEquals("valid")){
        return TravelRequestDao.getTravelDetails(sourceCity,destinationCity);
    }
    else{
        return null;
    }
}

```

```

}

/**
 * Method to invoke calculateTotalTravelCost method of TravelRequestDao class
 *
 * @return totalCost
 */
public double calculateTotalTravelCost(String approvalStatus) {
    if(this.validateApprovalStatus(approvalStatus).equals("valid")){
        return TravelRequestDao.calculateTotalTravelCost(approvalStatus);
    }
    else{
        return -1;
    }
}

```

### **SkeletonValidator.java\***

```

package com.cts.travelrequest.skeletonvalidator;

import java.lang.reflect.Method;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * @author t-aarti3
 *      This class is used to verify if the Code Skeleton is intact and not
 *      modified by participants thereby ensuring smooth auto evaluation
 */

public class SkeletonValidator {
    public SkeletonValidator() {
        validateClassName("com.cts.travelrequest.service.TravelRequestService");
        validateClassName("com.cts.travelrequest.vo.TravelRequest");
    }
}

```

```

validateMethodSignature()

"validateApprovalStatus:java.lang.String,validateSourceAndDestination:java.lang.String,getTravelDetails:java
.util.List,calculateTotalTravelCost:double",
"com.cts.travelrequest.service.TravelRequestService");

}

private static final Logger LOG = Logger.getLogger("SkeletonValidator");

protected final boolean validateClassName(String className) {

    boolean iscorrect = false;

    try {

        Class.forName(className);

        iscorrect = true;

        LOG.info("Class Name " + className + " is correct");

    } catch (ClassNotFoundException e) {

        LOG.log(Level.SEVERE, "You have changed either the " + "class name/package. Use the
correct package "
                + "and class name as provided in the skeleton");

    } catch (Exception e) {

        LOG.log(Level.SEVERE,
                "There is an error in validating the " + "Class Name. Please manually verify
that the "
                + "Class name is same as skeleton before uploading");

    }

    return iscorrect;
}

protected final void validateMethodSignature(String methodWithExcptn, String className) {

    Class cls = null;

    try {

        String[] actualmethods = methodWithExcptn.split(",");

```

```

boolean errorFlag = false;
String[] methodSignature;
String methodName = null;
String returnType = null;

for (String singleMethod : actualmethods) {
    boolean foundMethod = false;
    methodSignature = singleMethod.split(":");
    methodName = methodSignature[0];
    returnType = methodSignature[1];
    cls = Class.forName(className);
    Method[] methods = cls.getMethods();
    for (Method findMethod : methods) {
        if (methodName.equals(findMethod.getName())) {
            foundMethod = true;
            if (!(findMethod.getReturnType().getName().equals(returnType))) {
                errorFlag = true;
                LOG.log(Level.SEVERE, " You have changed the " + "return
type in " + methodName
                                         + " method. Please stick to the " + "skeleton
provided");
            }
        } else {
            LOG.info("Method signature of " + methodName + " is
valid");
        }
    }
    if (!foundMethod) {
        errorFlag = true;
        LOG.log(Level.SEVERE, " Unable to find the given public method " +
methodName
                                         + ". Do not change the " + "given public method name. " +
"Verify it with the skeleton");
    }
}

```

```

        }

    }

    if (!errorFlag) {
        LOG.info("Method signature is valid");
    }

}

} catch (Exception e) {
    LOG.log(Level.SEVERE,
        " There is an error in validating the " + "method structure. Please manually
verify that the "
        + "Method signature is same as the skeleton before
uploading");
}

}

}

```

### **TravelRequest.java\***

```

package com.cts.travelrequest.vo;

import java.util.Date;

public class TravelRequest {
    // member variables
    private String travelReqId;
    private Date travelDate;
    private String approvalStatus;
    private String sourceCity;
    private String destinationCity;
    private double travelCost;

    public TravelRequest() {
        super();
        // TODO Auto-generated constructor stub
    }
}

```

```
}

// parameterized constructor

public TravelRequest(String travelReqId, Date travelDate, String approvalStatus, String sourceCity,
                     String destinationCity, double travelCost) {

    super();

    this.travelReqId = travelReqId;

    this.travelDate = travelDate;

    this.approvalStatus = approvalStatus;

    this.sourceCity = sourceCity;

    this.destinationCity = destinationCity;

    this.travelCost = travelCost;

}

// setter, getter


/***
 * @return the travelReqId
 */
public String getTravelReqId() {

    return travelReqId;

}

/***
 * @param travelReqId
 *      the travelReqId to set
 */
public void setTravelReqId(String travelReqId) {

    this.travelReqId = travelReqId;

}

/***
 * @return the travelDate
 */
public Date getTravelDate() {

    return travelDate;

}

/***
```

```
* @param travelDate
*      the travelDate to set
*/
public void setTravelDate(Date travelDate) {
    this.travelDate = travelDate;
}

/**
* @return the approvalStatus
*/
public String getApprovalStatus() {
    return approvalStatus;
}

/**
* @param approvalStatus
*      the approvalStatus to set
*/
public void setApprovalStatus(String approvalStatus) {
    this.approvalStatus = approvalStatus;
}

/**
* @return the sourceCity
*/
public String getSourceCity() {
    return sourceCity;
}

/**
* @param sourceCity
*      the sourceCity to set
*/
public void setSourceCity(String sourceCity) {
    this.sourceCity = sourceCity;
}

/**
* @return the sourceCity
```

```

        */
    public String getDestinationCity() {
        return destinationCity;
    }

    /**
     * @param destinationCity
     *      the destinationCity to set
     */

    public void setDestinationCity(String destinationCity) {
        this.destinationCity = destinationCity;
    }

    /**
     * @return the travelCost
     */

    public double getTravelCost() {
        return travelCost;
    }

    /**
     * @param travelCost
     *      the travelCost to set
     */

    public void setTravelCost(double travelCost) {
        this.travelCost = travelCost;
    }

}

```

### Main.java\*

```

package com.cts.travelrequest.main;

import java.sql.*;
import java.util.*;
import java.text.SimpleDateFormat;
import com.cts.travelrequest.service.TravelRequestService;

```

```

import com.cts.travelrequest.skeletonvalidator.SkeletonValidator;
import com.cts.travelrequest.vo.TravelRequest;
public class Main {

    public static void main(String[] args) throws SQLException {
        // CODE SKELETON - VALIDATION STARTS
        // DO NOT CHANGE THIS CODE
        new SkeletonValidator();
        // CODE SKELETON - VALIDATION ENDS
        //TravelRequest tr=new TravelRequest();
        //List<TravelRequest> ltr=new ArrayList<>();
        TravelRequestService service = new TravelRequestService();
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter source city:");
        String sourceCity=sc.next();
        System.out.println("Enter destination city:");
        String destinationCity=sc.next();
        System.out.println("Enter approval status to find total travel cost:");
        String status=sc.next();

        if(service.validateSourceAndDestination(sourceCity,destinationCity).equals("valid")){
            List<TravelRequest> ltr=service.getTravelDetails(sourceCity, destinationCity);
            if(ltr.isEmpty()){
                System.out.println("No travel request raised for given source and destination cities");
            }
            else{
                for(TravelRequest t:ltr){
                    SimpleDateFormat sd= new SimpleDateFormat("dd-MMM-YYYY");
                    String d=sd.format(t.getTravelDate());
                    System.out.println(t.getTravelReqId()+"\t| "+d+"\t| "+t.getApprovalStatus()+"\t| "+t.getSourceCity()+"\t| "+t.getDestinationCity()+"\t| "+t.getTravelCost());
                }
            }
        }
    }
}

```

```
else{
    System.out.println("Provide correct source and destination city");
}

if(service.validateApprovalStatus(status).contentEquals("valid")){
    System.out.println(service.calculateTotalTravelCost(status));
}

else{
    System.out.println("Provide valid approval status");
}

}

}
```

# **JAVA ICT CODES**

## **Rainfall Report Automation**

### **AnnualRainfall**

```
public class AnnualRainfall {  
  
    private int cityPincode;  
    private String cityName;  
    private double averageAnnualRainfall;  
  
    public int getCityPincode() {  
        return cityPincode;  
    }  
  
    public void setCityPincode(int cityPincode) {  
        this.cityPincode = cityPincode;  
    }  
  
    public String getCityName(){  
        return cityName;  
    }  
  
    public void setCityName(String cityName){  
        this.cityName = cityName;  
    }  
}
```

```
public double getAverageAnnualRainfall(){
    return averageAnnualRainfall;
}

public void
setAverageAnnualRainfall(double
averageAnnualRainfall){
    this.averageAnnualRainfall =
averageAnnualRainfall;
}

public void calculateAverageAnnualRainfall
(double monthlyRainfall [ ]){
    double average=0;
    for(int i=0;i<monthlyRainfall.length;i++)
    {
        average+=monthlyRainfall[i];
    }
    average/=12;
    this.averageAnnualRainfall=average;
}

}
```

## DBHandler

```
import java.io.FileInputStream;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;

public class DBHandler {

    private static Connection con = null;
    private static Properties props = new Properties();
    //Write the required business logic as
    //expected in the question description
    public Connection establishConnection()
        throws ClassNotFoundException,
        SQLException {
        try{
            FileInputStream fis
                = null;
            fis = new
            FileInputStream("db.properties");
            props.load(fis);

            // load the Driver
            Class
                Class.forName(props.getProperty("d
b.classname"));

            // create the
            // connection now
            con =
            DriverManager.getConnection(props.getPro
perty("db.url"),props.getProperty("db.usern
ame"),props.getProperty("db.password"));
        }
    }
}
```

```
        }

        catch(IOException e){

            e.printStackTrace();

        }

        return con;

    }

}
```

### **InvalidCityPincodeException**

```
@SuppressWarnings("serial")

public class InvalidCityPincodeException extends Exception {

    public
    InvalidCityPincodeException(String s)

    {

        super(s);

    }

}
```

### **Main**

```
import java.io.IOException;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

public class Main {

    public static void main(String[] args)

        throws IOException,

        InvalidCityPincodeException,

        ClassNotFoundException, SQLException {
```

```

        RainfallReport rf = new
        RainfallReport();

        List<AnnualRainfall> avgli = new
        ArrayList<AnnualRainfall>();

        avgli =
        rf.generateRainfallReport("AllCityMonthlyR
ainfall.txt");

        List<AnnualRainfall> maxli = new
        ArrayList<AnnualRainfall>();

        maxli =
        rf.findMaximumRainfallCities();

        for (int i = 0; i < maxli.size(); i++) {

            AnnualRainfall ob =
            maxli.get(i);

            System.out.println("City
Pincode:" + ob.getCityPincode());

            System.out.println("City
Name:" + ob.getCityName());

            System.out.println("Average
RainFall:" + ob.getAverageAnnualRainfall());

        }

    }

}

```

## **RainfallReport**

```

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.sql.Connection;

```

```
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

public class RainfallReport {

    public List<AnnualRainfall>
    generateRainfallReport(String filePath)
    throws IOException {

        List<AnnualRainfall> avgList=new
        ArrayList<>();

        FileReader fr = new FileReader(new
        File(filePath));

        BufferedReader br = new
        BufferedReader(fr);

        String l;
        while((l=br.readLine())!=null)
        {
            String[] a=l.split(",");
            String pincode=a[0];
            try
            {
                if(validate(pincode))
                {
                    double[]
                    monthlyRainFall=new double[12];
                    for(int
i=2;i<=13;i++)
                    {

```

```
        monthlyRainFall[i-  
2]=Double.parseDouble(a[i]);  
    }  
  
    AnnualRainfall ar=new  
    AnnualRainfall();  
  
    ar.calculateAverageAnnualRainfall(  
monthlyRainFall);  
  
    ar.setCityName(a[1]);  
  
    ar.setCityPincode(Integer.parseInt(p  
incode));  
  
    avgList.add(ar);  
}  
}  
  
}  
  
catch(InvalidCityPincodeException  
e)  
{  
  
System.out.println(e.getMessage());  
}  
}  
  
br.close();  
  
return avgList;  
}  
  
}  
  
public List<AnnualRainfall>  
findMaximumRainfallCities() throws  
SQLException, ClassNotFoundException,  
IOException {  
  
DBHandler d=new DBHandler();
```

```
        List<AnnualRainfall> finalList=new
        ArrayList<>();

        Connection
c=d.establishConnection();

        Statement s=c.createStatement();

        String sql = "SELECT * FROM
ANNUALRAINFALL WHERE
AVERAGE_ANNUAL_RAINFALL IN (SELECT
MAX(AVERAGE_ANNUAL_RAINFALL) FROM
ANNUALRAINFALL)";

        ResultSet rs=s.executeQuery(sql);

        while(rs.next())
        {

            AnnualRainfall ar=new
AnnualRainfall();

            ar.setCityName(rs.getString(2));

            ar.setCityPincode(Integer.parseInt(r
s.getString(1)));

            ar.setAverageAnnualRainfall(Double
.parseDouble(rs.getString(3)));

            finalList.add(ar);
        }

        return finalList;
    }
}
```

```
public boolean validate(String cityPincode)
throws InvalidCityPincodeException {

    if(cityPincode.length()==5)

    {

        return true;
    }

    else
```

```
        {
            throw new
            InvalidCityPincodeException("Invalid
            sCityPincode Exception");
        }
    }
```

## Electricity Bill Automation

### DBHandler

```
import java.sql.Connection;
import java.io.FileInputStream;
import java.io.IOException;
import java.sql.*;
import java.util.Properties;
import java.io.FileNotFoundException;
```

```
public class DBHandler {
```

```
    public Connection establishConnection()
        throws ClassNotFoundException,
        SQLException, FileNotFoundException {
```

```
        Connection con = null;
```

```
Properties props = new Properties();

// this try block reads the db
Properties file and establishConnection.

try{

    FileInputStream fis = new
    FileInputStream("src/db.properties");

    props.load(fis);

    Class.forName(props.getProperty("db.cl
assname"));

    con =
    DriverManager.getConnection(props.get
Property("db.url"),props.getProperty("d
b.username"),props.getProperty("db.pa
ssword"));

}

catch(IOException e){

    e.printStackTrace();

}

return con;

}

//fill code here

}
```

## ElectricityBill

```
//This is the POJO/model class

public class ElectricityBill {

    private String consumerNumber;
    private String consumerName;
    private String consumerAddress;
    private int unitsConsumed;
    private double billAmount;

    public String getConsumerNumber() {
        return consumerNumber;
    }

    public void setConsumerNumber(String consumerNumber) {
        this.consumerNumber = consumerNumber;
    }

    public String getConsumerName() {
        return consumerName;
    }

    public void setConsumerName(String consumerName) {
        this.consumerName = consumerName;
    }
}
```

```
public String getConsumerAddress() {  
    return consumerAddress;  
}
```

```
public void setConsumerAddress(String  
consumerAddress) {  
    this.consumerAddress =  
consumerAddress;  
}
```

```
public int getUnitsConsumed() {  
    return unitsConsumed;  
}
```

```
public void setUnitsConsumed(int  
unitsConsumed) {  
    this.unitsConsumed =  
unitsConsumed;  
}
```

```
public double getBillAmount() {  
    return billAmount;  
}
```

```
public void setBillAmount(double  
billAmount) {  
    this.billAmount = billAmount;  
}
```

```
//Write the required business logic as  
expected in the question description  
  
public void calculateBillAmount() {  
  
    // method for calculating the bill  
    amount.  
  
    int units = unitsConsumed;  
  
    double bill = 0;  
  
  
    if(units <= 100){  
  
        bill = 0;  
  
    }  
  
    if(units > 100 && units <= 300){  
  
        bill = (units-100) * 1.5;  
  
    }  
  
    if(units > 300 && units <= 600){  
  
        bill = 200 * 1.5 + (units-300) * 3.5;  
  
    }  
  
    if(units > 600 && units <= 1000){  
  
        bill = 200 * 1.5 + 300 * 3.5 + (units-  
        600) * 5.5;  
  
    }  
  
    if(units > 1000){  
  
        bill = 200 * 1.5 + 300 * 3.5 + 400 *  
        5.5 + (units-1000) * 7.5;  
  
    }  
  
  
    setBillAmount(bill);  
  
  
    //fill the code
```

```
}
```

```
}
```

## ElectricityBoard

```
import java.util.List;
import java.util.*;
import java.io.FileReader;
import java.io.File;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.regex.Pattern;
import java.sql.SQLException;
import java.sql.Connection;
import java.sql.PreparedStatement;

public class ElectricityBoard {

    //write the required business logic methods as expected in the question description

    public void addBill(List<ElectricityBill> billList) {

        DBHandler db = new DBHandler();

        try(Connection con = db.establisConnection()){

            PreparedStatement stmt = con.prepareStatement("insert into ElectricityBill
values(?,?,?,?,?);");

```

```
// for loop to insert the values into the table  
  
for(ElectricityBill obj : billList){  
  
    stmt.setString(1,obj.getConsumerNumber());  
  
    stmt.setString(2,obj.getConsumerName());  
  
    stmt.setString(3,obj.getConsumerAddress());  
  
    stmt.setInt(4,obj.getUnitsConsumed());  
  
    stmt.setDouble(5,obj.getBillAmount());  
  
  
    stmt.execute();  
  
}  
  
}  
  
catch(ClassNotFoundException e){  
  
    e.printStackTrace();  
  
}  
  
  
catch(FileNotFoundException e){  
  
    e.printStackTrace();  
  
}  
  
  
catch(SQLException e){  
  
    e.printStackTrace();  
  
}  
  
  
//fill the code
```

```
}
```

```
public List<ElectricityBill> generateBill(String filePath) {  
  
    List <ElectricityBill> list = new ArrayList<>();  
    File f = new File (filePath);  
  
    // this try block is for opening and reading the file  
    try(BufferedReader br = new BufferedReader(new FileReader(f)))  
    {  
        String line = null;  
  
        while((line = br.readLine())!= null)  
        {  
            String records[] = null;  
            String consumerNumber = "";  
            String consumerName = "";  
            String consumerAddress = "";  
            int unitsConsumed = 0;  
  
            records = line.split(",");  
            consumerNumber = records[0];  
            consumerName = records[1];  
            consumerAddress = records[2];  
            unitsConsumed = Integer.parseInt(records[3]);  
  
            //this try block checks for the validated consumerNumber  
            try{  
                  
            }  
            catch(Exception e){  
                System.out.println("Consumer Number "+ consumerNumber + " is invalid");  
            }  
            ElectricityBill bill = new ElectricityBill(consumerNumber, consumerName, consumerAddress, unitsConsumed);  
            list.add(bill);  
        }  
    }  
}
```

```
if(validate(consumerNumber)){

    ElectricityBill obj = new ElectricityBill();

    obj.setConsumerNumber(consumerNumber);

    obj.setConsumerName(consumerName);

    obj.setConsumerAddress(consumerAddress);

    obj.setUnitsConsumed(unitsConsumed);

    obj.calculateBillAmount();

    list.add(obj);

}

}

catch(InvalidConsumerNumberException e){

    System.out.println(e.getMessage());

}

}

}

catch(FileNotFoundException e){

    e.printStackTrace();

}

catch(IOException e){

    e.printStackTrace();

}

}

return list;

//fill the code

}

public boolean validate(String consumerNumber) throws
    InvalidConsumerNumberException {
```

```

// method for validating the consumerNumber

boolean isValid = Pattern.matches("^[0][0-9]{9}" , consumerNumber);

if(!isValid){
    throw new InvalidConsumerNumberException("Invalid Consumer Number");
}

return true;

//fill the code

}

}

```

## InvalidConsumerNumberException

```

//make the required changes to this class so that InvalidConsumerNumberException is of
//type exception.

public class InvalidConsumerNumberException extends Exception{

    public InvalidConsumerNumberException(String message)
    {
        super(message);
    }
}
```

```
//fill the code
```

```
}
```

## Main

```
import java.util.*;  
import java.util.List;  
import java.util.ArrayList;  
public class Main {  
  
    public static void main(String[] args) {  
        Scanner sc= new Scanner(System.in);  
  
        String filePath = "src/ElectricityBill.txt";  
  
        List<ElectricityBill> list = new ArrayList<>();  
  
        ElectricityBoard eb = new ElectricityBoard();  
        list = eb.generateBill(filePath);  
  
        for(ElectricityBill obj: list){  
            System.out.println(obj.getConsumerNumber() + " " + obj.getConsumerName() + " " +  
                obj.getBillAmount());  
        }  
  
        eb.addBill(list);  
    }  
}
```

```
        System.out.println("Successfully Inserted");

        sc.close();
        //fill your code here

    }

}
```

## **CreditCardAdminSystem**

### **CreditCardDAO**

```
/****************************************************************************
 * This class CreditCardDAO is used to persist or retrieve data from database.
 *
 * DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS,
 * EXCEPTION CLAUSES, RETURN TYPES
 *
 * YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
 *
 * DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS
 *
 * DO TEST YOUR CODE USING MAIN METHOD
 *
 * CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC
 *
 * DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF
 * NEED BE,
 *
 * YOU CAN CATCH THEM AND THROW ONLY
 *
 * THE APPLICATION SPECIFIC EXCEPTION AS PER EXCEPTION CLAUSE
 *
 *
```

```
*****
*****
/
package com.cts.creditcard.dao;

import java.sql.Connection;
import java.util.List;

import com.cts.creditcard.exception.CreditCardAdminSystemException;
import com.cts.creditcard.vo.CreditCard;

public class CreditCardDAO {

    private static Connection conn = null;

    public Boolean
        addCreditCardDetails(List<CreditCard>
    cards) throws
        CreditCardAdminSystemException {
        //TODO add your code here
        return false; //TODO CHANGE
        THIS RETURN TYPE
    }

}
```

## CreditCardAdminSystemException

```
*****
*****
```

- \* This class CreditCardAdminSystemException is as exception class.
- \*
- \* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS,  
EXCEPTION CLAUSES, RETURN TYPES
- \* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
- \* DO NOT SUBMIT THE CODE WITH COMPILE ERRORS
- \* DO TEST YOUR CODE USING MAIN METHOD
- \* CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC
- \* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF  
NEED BE,
- \* YOU CAN CATCH THEM AND THROW ONLY
- \* THE APPLICATION SPECIFIC EXCEPTION AS PER EXCEPTION CLAUSE

\*

```
*****  
*****  
/
```

package com.cts.creditcard.exception;

```
public class CreditCardAdminSystemException extends Exception {
```

```
    private static final long serialVersionUID  
    = -6349759544203601561L;
```

```
    //TODO add your constructors here
```

```
}
```

## MainApp

```
*****  
*****
```

\* This class MainApp is used to run the service methods and to test the database.

```

*
* YOU CAN INVOKE THE METHODS AS REQUIRED FROM HERE TO TEST THE APP
* DO NOT USE VIA COMMAND LINE ARGUMENTS FROM MAIN METHOD, UNLESS SPECIFIED
*
*****
***** /
package com.cts.creditcard.main;

public class MainApp {
    public static void main(String ag[]) {
        //TODO add your code here
    }
}

```

## CreditCardAdminService

```

/*****
*****
* This class CreditCardAdminService is used to handle business logic for the proposed
system.
*
* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS,
EXCEPTION CLAUSES, RETURN TYPES
*
* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
*
* DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS
*
* DO TEST YOUR CODE USING MAIN METHOD
*
* CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC
*
* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF
NEED BE,

```

\* YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER  
EXCEPTION CLAUSE

\*

\*\*\*\*\*

\*\*\*\*\*

/

```
package com.cts.creditcard.service;
```

```
import java.util.List;
```

```
import com.cts.creditcard.exception.CreditCardAdminSystemException;
```

```
import com.cts.creditcard.vo.CreditCard;
```

```
public class CreditCardAdminService {
```

```
/**
```

```
* @param records
```

```
* @return List<Customer>
```

```
*/
```

```
public static List<CreditCard>
```

```
buildMasterCreditCardList(List<String>
```

```
records) {
```

```
// TODO Add your logic here
```

```
return null; // TODO change this  
return value
```

```
}
```

```
/**
```

```
* @param billAmount
```

```
* @return Double
```

```

        */

    public static Double
    getBillAmountWithLatePaymentCharges
    (Double billAmount) {

        // TODO add your logic here

        return 0.00; // TODO change this
        return value

    }

    /**
     * @param inputFeed
     * @return Boolean
     * @throws
     CreditCardAdminSystemException
     */

    public Boolean
    addCreditCardDetails(String inputFeed)
    throws
    CreditCardAdminSystemException {

        // TODO add your logic here

        return null; //TODO change this
        return value

    }

}

```

## ApplicationUtil

```

*****
*****
```

\* This class ApplicationUtil is used for any utility methods needed for service or dao classes

\*

\* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS,  
EXCEPTION CLAUSES, RETURN TYPES

\* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE

\* DO NOT SUBMIT THE CODE WITH COMPILE ERRORS

\* DO TEST YOUR CODE USING MAIN METHOD

\* CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC

\* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF  
NEED BE,

\* YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER  
EXCEPTION CLAUSE

\*

\*\*\*\*\*

\*\*\*\*\*

/

```
package com.cts.creditcard.util;
```

```
import java.util.Date;
```

```
import java.util.List;
```

```
import com.cts.creditcard.exception.CreditCardAdminSystemException;
```

```
public class ApplicationUtil {
```

```
/**
```

```
 * @param fileName
```

```
 * @return List<String>
```

```
 * @throws
```

```
CreditCardAdminSystemException
```

```
 */
```

```
public static List<String> readFile(String  
fileName) throws  
CreditCardAdminSystemException {  
  
    // TODO Add your logic here  
  
    return null; // TODO change this  
    return value  
}  
  
public static Date  
geDateWithoutTime(Date date) {  
  
    // TODO Add your logic here  
  
    return null; // TODO change this  
    return value  
}  
  
}  
  
/**  
 * @param util  
 *      Date  
 * @return sql Date  
 */  
public static java.sql.Date  
convertUtilToSqlDate(java.util.Date  
uDate) {  
  
    // TODO Add your logic here  
  
    return null; // TODO change this  
    return value  
}  
  
/**  
 * @param inDate  
 * @return Date
```

```
        */  
  
        public static Date  
        convertStringToDate(String inDate) {  
  
            // TODO Add your logic here  
  
            return null; // TODO change this  
            return value  
  
        }  
  
    }  
  
}
```

## DBConnectionManager

```
/*****************************************************************************  
*****  
* This class DBConnectionManager is used acquire database connection  
*  
* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS,  
* EXCEPTION CLAUSES, RETURN TYPES  
* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE  
* DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS  
* DO TEST YOUR CODE USING MAIN METHOD  
* CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC  
* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF  
* NEED BE,  
* YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER  
* EXCEPTION CLAUSE  
*  
*****  
*****  
/
```

```
package com.cts.creditcard.util;
```

```
import com.cts.creditcard.exception.CreditCardAdminSystemException;

public class DBConnectionManager {

    /**
     * @throws
     CreditCardAdminSystemException
     */

    private DBConnectionManager() throws
    CreditCardAdminSystemException {

    }

    /**
     * @return DBConnectionManager
     * @throws
     CreditCardAdminSystemException
     */

    public static DBConnectionManager
    getInstance() throws
    CreditCardAdminSystemException {
        // TODO Add your logic here

        return null; // TODO change this
        return value
    }

}

CreditCard
```

```
*****  
*****
```

\* This class CreditCard is a value object for data transfer between Service and DAO layers

\*

\* DO NOT CHANGE THE NAMES OR DATA TYPES OR VISIBILITY OF THE BELOW MEMBER  
VARIABLES

\* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS,  
EXCEPTION CLAUSES, RETURN TYPES

\* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE

\* DO NOT SUBMIT THE CODE WITH COMPILE ERRORS

\* DO TEST YOUR CODE USING MAIN METHOD

\* CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC

\* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF  
NEED BE,

\* YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER  
EXCEPTION CLAUSE

\*

```
*****  
*****
```

/

```
package com.cts.creditcard.vo;
```

```
import java.util.Date;
```

```
public class CreditCard {
```

```
//DO NOT CHANGE THE NAMES OR  
DATA TYPES OR VISIBILITY OF THE  
BELOW MEMBER VARIABLES
```

```
    public Long creditCardNum;
```

```
    public String customerName;
```

```
        public String customerEmail;  
        public Long customerPhone;  
        public Double billAmount;  
        public Date dueDate;  
        public Date paymentDate;  
  
        //TODO add your code here  
  
    }
```

## **CreditCardAdminSystem**

### **CreditCardDAO**

```
/*****************************************************************************  
 * This class CreditCardDAO is used to persist or retrieve data from database.  
 *  
 * DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS,  
 * EXCEPTION CLAUSES, RETURN TYPES  
 *  
 * YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE  
 *  
 * DO NOT SUBMIT THE CODE WITH COMPILED ERRORS  
 *  
 * DO TEST YOUR CODE USING MAIN METHOD  
 *  
 * CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC  
 *  
 * DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF  
 * NEED BE,  
 *  
 * YOU CAN CATCH THEM AND THROW ONLY  
 *  
 * THE APPLICATION SPECIFIC EXCEPTION AS PER EXCEPTION CLAUSE  
 *
```

```
*****
*****
/
package com.cts.creditcard.dao;

import java.sql.Connection;
import java.util.List;

import com.cts.creditcard.exception.CreditCardAdminSystemException;
import com.cts.creditcard.vo.CreditCard;

public class CreditCardDAO {

    private static Connection conn = null;

    public Boolean
        addCreditCardDetails(List<CreditCard>
    cards) throws
        CreditCardAdminSystemException {
        //TODO add your code here
        return false; //TODO CHANGE
        THIS RETURN TYPE
    }

}
```

## CreditCardAdminSystemException

```
*****
*****
```

- \* This class CreditCardAdminSystemException is as exception class.
- \*
- \* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS,  
EXCEPTION CLAUSES, RETURN TYPES
- \* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
- \* DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS
- \* DO TEST YOUR CODE USING MAIN METHOD
- \* CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC
- \* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF  
NEED BE,

- \* YOU CAN CATCH THEM AND THROW ONLY
- \* THE APPLICATION SPECIFIC EXCEPTION AS PER EXCEPTION CLAUSE

- \*

```
*****  
*****  
/
```

```
package com.cts.creditcard.exception;
```

```
public class CreditCardAdminSystemException extends Exception {
```

```
    private static final long serialVersionUID  
    = -6349759544203601561L;
```

```
    //TODO add your constructors here
```

```
}
```

## MainApp

```
*****  
*****
```

- \* This class MainApp is used to run the service methods and to test the database.

```
*  
* YOU CAN INVOKE THE METHODS AS REQUIRED FROM HERE TO TEST THE APP  
* DO NOT USE VIA COMMAND LINE ARGUMENTS FROM MAIN METHOD, UNLESS SPECIFIED  
*  
*****  
*****  
/  
package com.cts.creditcard.main;  
  
public class MainApp {  
    public static void main(String ag[]) {  
        //TODO add your code here  
    }  
}
```

### CreditCardAdminService

```
*****  
*****  
* This class CreditCardAdminService is used to handle business logic for the proposed  
* system.  
*  
* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS,  
* EXCEPTION CLAUSES, RETURN TYPES  
* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE  
* DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS  
* DO TEST YOUR CODE USING MAIN METHOD  
* CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC  
* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF  
* NEED BE,  
* YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER  
* EXCEPTION CLAUSE
```

```
*  
*****  
*****  
/  
  
package com.cts.creditcard.service;  
  
import java.util.List;  
  
import com.cts.creditcard.exception.CreditCardAdminSystemException;  
import com.cts.creditcard.vo.CreditCard;  
  
public class CreditCardAdminService {  
  
    /**  
     * @param records  
     * @return List<Customer>  
     */  
    public static List<CreditCard>  
    buildMasterCreditCardList(List<String>  
    records) {  
        // TODO Add your logic here  
  
        return null; // TODO change this  
        return value  
    }  
  
    /**  
     * @param billAmount  
     * @return Double  
     */
```

```

        public static Double
        getBillAmountWithLatePaymentCharges
        (Double billAmount) {

            // TODO add your logic here

            return 0.00; // TODO change this
            return value

        }

        /**
         * @param inputFeed
         * @return Boolean
         * @throws
         CreditCardAdminSystemException
         */

        public Boolean
        addCreditCardDetails(String inputFeed)
        throws
        CreditCardAdminSystemException {

            // TODO add your logic here

            return null; //TODO change this
            return value

        }

    }

```

## ApplicationUtil

```

*****
*****  

* This class ApplicationUtil is used for any utility methods needed for service or dao classes
*

```

- \* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS,  
EXCEPTION CLAUSES, RETURN TYPES
- \* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
- \* DO NOT SUBMIT THE CODE WITH COMPILED ERRORS
- \* DO TEST YOUR CODE USING MAIN METHOD
- \* CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC
- \* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF  
NEED BE,
- \* YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER  
EXCEPTION CLAUSE

\*

```
*****
***** /
```

```
package com.cts.creditcard.util;

import java.util.Date;
import java.util.List;

import com.cts.creditcard.exception.CreditCardAdminSystemException;

public class ApplicationUtil {

    /**
     * @param fileName
     * @return List<String>
     * @throws CreditCardAdminSystemException
     */
    public static List<String> readFile(String
fileName) throws
CreditCardAdminSystemException {
```

```
// TODO Add your logic here

        return null; // TODO change this
return value
}

public static Date
geDateWithoutTime(Date date) {
    // TODO Add your logic here
    return null; // TODO change this
return value
}

}

/***
 * @param util
 *      Date
 * @return sql Date
 */
public static java.sql.Date
convertUtilToSqlDate(java.util.Date
uDate) {
    // TODO Add your logic here
    return null; // TODO change this
return value
}

/***
 * @param inDate
 * @return Date
 */
public static Date
convertStringToDate(String inDate) {
```

```
// TODO Add your logic here
        return null; // TODO change this
        return value
    }

}

/*****
*****
* This class DBConnectionManager is used acquire database connection
*
* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS,
* EXCEPTION CLAUSES, RETURN TYPES
*
* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
*
* DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS
*
* DO TEST YOUR CODE USING MAIN METHOD
*
* CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC
*
* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF
* NEED BE,
*
* YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER
* EXCEPTION CLAUSE
*
*****
*/
package com.cts.creditcard.util;

import com.cts.creditcard.exception.CreditCardAdminSystemException;

public class DBConnectionManager {
```

```

    /**
     * @throws
     CreditCardAdminSystemException
    */

    private DBConnectionManager() throws
    CreditCardAdminSystemException {

}

    /**
     * @return DBConnectionManager
     * @throws
     CreditCardAdminSystemException
    */

    public static DBConnectionManager
getInstance() throws
CreditCardAdminSystemException {

    // TODO Add your logic here

        return null; // TODO change this
        return value
    }

}

```

## CreditCard

```

*****
*****  

* This class CreditCard is a value object for data transfer between Service and DAO layers
*

```

- \* DO NOT CHANGE THE NAMES OR DATA TYPES OR VISIBILITY OF THE BELOW MEMBER VARIABLES
- \* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS, EXCEPTION CLAUSES, RETURN TYPES
- \* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
- \* DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS
- \* DO TEST YOUR CODE USING MAIN METHOD
- \* CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC
- \* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF NEED BE,
- \* YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER EXCEPTION CLAUSE

\*

```
*****  
*****  
/
```

```
package com.cts.creditcard.vo;
```

```
import java.util.Date;
```

```
public class CreditCard {
```

```
//DO NOT CHANGE THE NAMES OR  
DATA TYPES OR VISIBILITY OF THE  
BELOW MEMBER VARIABLES
```

```
    public Long creditCardNum;  
    public String customerName;  
    public String customerEmail;  
    public Long customerPhone;  
    public Double billAmount;  
    public Date dueDate;  
    public Date paymentDate;
```

```
//TODO add your code here  
}
```

UNOAdmission\_

StudentAdmissionDAO

```
package com.cts.unoadm.dao;
```

```
import java.util.ArrayList;
```

```
import java.util.List;
```

```
import com.cts.unoadm.exception.StudentAdmissionException;
```

```
import com.cts.unoadm.vo.StudentAdmission;
```

```
public class StudentAdmissionDAO {
```

```
    public boolean  
    addStudentAdmissionDetails(List<Stude  
ntAdmission> stdAdmissions) throws  
StudentAdmissionException {
```

```
        boolean recordsAdded = false;
```

```
        //code here
```

```
        return recordsAdded;
```

```
}
```

```
public List<StudentAdmission>
getAllStudentAdmissionDetails() throws
StudentAdmissionException {

    List<StudentAdmission>
    stdAdmissions = new
    ArrayList<StudentAdmission>();

    //code here

    return stdAdmissions;

}

}
```

## **StudentAdmissionException**

```
package com.cts.unoadm.exception;

public class StudentAdmissionException extends Exception {

    private static final long serialVersionUID
    = -1105431869622052445L;

    /**
     * @param message
     * @param cause
     */
    public
    StudentAdmissionException(String
    message, Throwable cause) {
```

```
        super(message, cause);

    }

}

MainApp

package com.cts.unoadm.main;

import com.cts.unoadm.skeletonvalidator.SkeletonValidator;

public class MainApp {

    public static void main(String[] args) {
        //Don't delete this code
        //Skeletonvalidaton starts
        new SkeletonValidator();
        //Skeletonvalidation ends

        //Write your code here..

    }
}
```

## **StudentAdmissionService**

```
package com.cts.unoadm.service;
```

```
import java.util.ArrayList;
import java.util.List;

import com.cts.unoadm.exception.StudentAdmissionException;
import com.cts.unoadm.vo.StudentAdmission;

public class StudentAdmissionService {

    /**
     * @return List<StudentAdmission>
     */
    public static List<StudentAdmission>
    buildStudentAdmissionsList(List<String>
    studentAdmissionRecords) {
        List<StudentAdmission>
        studentAdmissionList = new
        ArrayList<StudentAdmission>();

        //Code here

        return studentAdmissionList;
    }

    public boolean
    addStudentAdmissionDetails(String
    inputFeed) throws
    StudentAdmissionException {
        //Code here

        return false;
    }
}
```

```

        }

public static double[]
calculateTotalCollegeFee(String
preferCollegeHostel, String
firstGraduate, String departmentName)
{
    double[] studentAdmissionCosts
= new double[4];

//Code here..

return studentAdmissionCosts;
}

public boolean
searchStudentAdmission(String
admissionId) throws
StudentAdmissionException {
    boolean status = false;

//Code here..

return status;
}
}

```

## SkeletonValidator

```
package com.cts.unoadm.skeletonvalidator;
```

```
import java.lang.reflect.Array;
import java.lang.reflect.Method;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * @author t-aarti3
 *
 *      This class is used to verify if the Code Skeleton is intact and not
 * modified by participants thereby ensuring smooth auto evaluation
 */

public class SkeletonValidator {
    public SkeletonValidator() {
        validateClassName("com.cts.uno
adm.util.DBConnectionManager");

        validateClassName("com.cts.uno
adm.util.ApplicationUtil");

        validateClassName("com.cts.uno
adm.service.StudentAdmissionService");

        validateClassName("com.cts.uno
adm.dao.StudentAdmissionDAO");

        validateClassName("com.cts.uno
adm.vo.StudentAdmission");

        validateClassName("com.cts.uno
adm.exception.StudentAdmissionExcept
ion");
    }
}
```

```
 validateMethodSignature(
```

```
    "addStudentAdmissionDetails:boolean,getAllStudentAdmissionDetails>List",
```

```
    "com.cts.unoadm.dao.StudentAdmissionDAO");
```

```
 validateMethodSignature(
```

```
    "buildStudentAdmissionsList>List,addStudentAdmissionDetails:boolean,calculateTotalCollegeFee:double[],searchStudentAdmission:boolean",
```

```
    "com.cts.unoadm.service.StudentAdmissionService");
```

```
 validateMethodSignature(
```

```
    "readFile>List,convertUtilToSqlDate:Date,convertStringToDate:Date,checkIfValidAdmission:boolean",
```

```
    "com.cts.unoadm.util.ApplicationUtil");
```

```
 validateMethodSignature(
```

```
    "getConnection:Connection,getInstance:DBConnectionManager",
```

```
    "com.cts.unoadm.util.DBConnectionManager");
```

```
}
```

```
private static final Logger LOG =
Logger.getLogger("SkeletonValidator");

protected final boolean
validateClassName(String className) {

    boolean iscorrect = false;

    try {

        Class.forName(className);

        iscorrect = true;

        LOG.info("Class Name " +
className + " is correct");

    } catch (ClassNotFoundException e) {

        LOG.log(Level.SEVERE,
"You have changed either the " + "class
name/package. Use the correct package
"

        + "and

class name as provided in the
skeleton");

    }

    } catch (Exception e) {

        LOG.log(Level.SEVERE,
"There is
an error in validating the " + "Class
Name. Please manually verify that the "

        + "Class name is same as
skeleton before uploading");

    }

    return iscorrect;
}
```

```
protected final void
validateMethodSignature(String
methodWithExcptn, String className) {

    Class cls = null;

    try {

        String[] actualmethods =
methodWithExcptn.split(",");
        boolean errorFlag = false;
        String[] methodSignature;
        String methodName =
null;
        String returnType = null;

        for (String singleMethod :
actualmethods) {
            boolean
foundMethod = false;
            methodSignature
= singleMethod.split(":");
            methodName =
methodSignature[0];
            returnType =
methodSignature[1];
            cls =
Class.forName(className);
            Method[]
methods = cls.getMethods();
            for (Method
findMethod : methods) {
```

```
        if  
(methodName.equals(findMethod.getN  
ame())) {
```

```
            foundMethod = true;
```

```
        if  
(!(findMethod.getReturnType().getSimpl  
eName().equals(returnType))) {
```

```
            errorFlag = true;
```

```
            LOG.log(Level.SEVERE, " You  
have changed the " + "return type in " +  
methodName
```

```
            + " method.  
Please stick to the " + "skeleton  
provided");
```

```
    }  
else {
```

```
    LOG.info("Method signature of "  
+ methodName + " is valid");
```

```
}
```

```
}
```

```
if (!foundMethod)  
{
```

```
    errorFlag =  
true;
```

```
    LOG.log(Level.SEVERE, " Unable  
to find the given public method " +  
methodName
```

```
        + ". Do not change the " + "given  
        public method name. " + "Verify it with  
        the skeleton");  
  
    }  
  
}  
  
if (!errorFlag) {  
  
    LOG.info("Method  
signature is valid");  
  
}  
  
}  
  
} catch (Exception e) {  
  
    LOG.log(Level.SEVERE,  
  
            " There is  
            an error in validating the " + "method  
            structure. Please manually verify that  
            the "  
  
            + "Method signature is same as  
            the skeleton before uploading");  
  
}  
  
}  
  
}
```

## ApplicationUtil

```
package com.cts.unoadm.util;
```

```
import java.util.ArrayList;
```

```
import java.util.Date;
import java.util.List;

import com.cts.unoadm.exception.StudentAdmissionException;

public class ApplicationUtil {

    /**
     * @param fileName
     * @return List<String>
     * @throws StudentAdmissionException
     */
    public static List<String> readFile(String
fileName) throws
StudentAdmissionException {
        List<String>
studentAdmissionList = new
ArrayList<String>();
        //Code here..

        return studentAdmissionList;
    }

    /**
     * @param util
     *      Date
     * @return sql Date
     */
    public static java.sql.Date
convertUtilToSqlDate(java.util.Date
uDate) {
```

```
java.sql.Date sDate = null;
```

```
//Code here..
```

```
return sDate;
```

```
}
```

```
/**
```

```
* @param inDate
```

```
* @return Date
```

```
*/
```

```
public static Date
```

```
convertStringToDate(String inDate) {
```

```
//Code here..
```

```
return new Date();//TODO
```

```
change this return value
```

```
}
```

```
public static boolean
```

```
checkIfValidAdmission(Date
```

```
dtOfCounseling, Date dtOfAdmission,
```

```
String manager) {
```

```
boolean admissionValidity =
```

```
false;
```

```
//Code here..
```

```
return admissionValidity;
```

```
    }  
}
```

## DBConnectionManager

```
/**  
 * Don't change this code  
 */  
  
package com.cts.unoadm.util;  
  
import java.io.FileInputStream;  
  
import java.io.FileNotFoundException;  
  
import java.io.IOException;  
  
import java.sql.Connection;  
  
import java.sql.DriverManager;  
  
import java.sql.SQLException;  
  
import java.util.Properties;  
  
  
import com.cts.unoadm.exception.StudentAdmissionException;  
  
  
  
  
  
  
public class DBConnectionManager {  
  
    public static final String PROPERTY_FILE  
        = "database.properties";  
  
    public static final String DRIVER =  
        "drivername";  
  
    public static final String URL = "url";  
  
    public static final String USER_NAME =  
        "username";
```

```
public static final String PASSWORD =
"password";

private static Connection connection =
null;

private static Properties props = null;

/** 
 * @throws StudentAdmissionException
 */
private DBConnectionManager() throws
StudentAdmissionException {
    loadProperties();
    try {

        Class.forName(props.getProperty
(DRIVER));
        this.connection =
DriverManager.getConnection(props.get
Property(URL),
props.getProperty(USER_NAME),

        props.getProperty(PASSWORD));
    } catch (ClassNotFoundException
ex) {

        throw new
StudentAdmissionException("Could not
find Driver class ", ex.getCause());
    } catch (SQLException e) {

        throw new
StudentAdmissionException("Database
Connection Creation Failed",
e.getCause());
    }
}
```

```
}
```

```
/**
```

```
* @return Connection
```

```
*/
```

```
public Connection getConnection() {
```

```
    return connection;
```

```
}
```

```
/**
```

```
* @return DBConnectionManager
```

```
* @throws StudentAdmissionException
```

```
*/
```

```
public static DBConnectionManager
```

```
getInstance() throws
```

```
StudentAdmissionException {
```

```
// Code here
```

```
    return null;
```

```
}
```

```
/**
```

```
* @throws StudentAdmissionException
```

```
*/
```

```
private void loadProperties() throws
```

```
StudentAdmissionException {
```

```
    FileInputStream inputStream =  
    null;
```

```
    try {
```

```

        inputStream = new
InputStream(PROPERTY_FILE);

        props = new Properties();
        props.load(inputStream);

    } catch (FileNotFoundException
e) {

        throw new
StudentAdmissionException("Database
Property File Not Found", e.getCause());

    } catch (IOException e) {

        throw new
StudentAdmissionException("Exception
during property file I/O", e.getCause());

    } finally {

        if (inputStream != null) {

            try {

                inputStream.close();

            } catch
(IOException e) {

                throw new
StudentAdmissionException("Exception
during property file I/O", e.getCause());

            }
        }
    }
}

```

## StudentAdmission

```
/*
 * Don't change this code
 */

package com.cts.unoadm.vo;

import java.util.Date;

public class StudentAdmission {

    String admissionId;
    String studentCode;
    Date dateOfCounseling;
    String departmentName;
    Date dateOfAdmission;
    String preferCollegeHostel;
    String firstGraduate;
    String managerApproval;
    double admissionFee;
    double tuitionFee;
    double hostelFee;
    double totalCollegeFee;
    String finalStatusOfAdmission;

    public StudentAdmission() {
        super();
    }

    public StudentAdmission(String
        admissionId, String studentCode, Date
```

```
        dateOfCounseling, String  
        departmentName,  
  
        Date dateOfAdmission,  
        String preferCollegeHostel, String  
        firstGraduate, String managerApproval,  
  
        double admissionFee,  
        double tuitionFee, double hostelFee,  
        double totalCollegeFee,  
  
        String  
        finalStatusOfAdmission) {  
  
        super();  
  
        this.admissionId = admissionId;  
  
        this.studentCode = studentCode;  
  
        this.dateOfCounseling =  
        dateOfCounseling;  
  
        this.departmentName =  
        departmentName;  
  
        this.dateOfAdmission =  
        dateOfAdmission;  
  
        this.preferCollegeHostel =  
        preferCollegeHostel;  
  
        this.firstGraduate =  
        firstGraduate;  
  
        this.managerApproval =  
        managerApproval;  
  
        this.admissionFee =  
        admissionFee;  
  
        this.tuitionFee = tuitionFee;  
  
        this.hostelFee = hostelFee;  
  
        this.totalCollegeFee =  
        totalCollegeFee;  
  
        this.finalStatusOfAdmission =  
        finalStatusOfAdmission;  
    }
```

```
public String getAdmissionId() {  
    return admissionId;  
}  
  
public void setAdmissionId(String admissionId) {  
    this.admissionId = admissionId;  
}  
  
public String getStudentCode() {  
    return studentCode;  
}  
  
public void setStudentCode(String studentCode) {  
    this.studentCode = studentCode;  
}  
  
public Date getDateOfCounseling() {  
    return dateOfCounseling;  
}  
  
public void setDateOfCounseling(Date dateOfCounseling) {  
    this.dateOfCounseling = dateOfCounseling;  
}  
  
public String getDepartmentName() {  
    return departmentName;
```

```
}
```

```
public void setDepartmentName(String  
departmentName) {
```

```
    this.departmentName =  
    departmentName;
```

```
}
```

```
public Date getDateOfAdmission() {
```

```
    return dateOfAdmission;
```

```
}
```

```
public void setDateOfAdmission(Date  
dateOfAdmission) {
```

```
    this.dateOfAdmission =  
    dateOfAdmission;
```

```
}
```

```
public String getPreferCollegeHostel() {
```

```
    return preferCollegeHostel;
```

```
}
```

```
public void  
setPreferCollegeHostel(String  
preferCollegeHostel) {
```

```
    this.preferCollegeHostel =  
    preferCollegeHostel;
```

```
}
```

```
public String getFirstGraduate() {
```

```
    return firstGraduate;
```

```
}
```

```
public void setFirstGraduate(String  
firstGraduate) {
```

```
    this.firstGraduate =  
    firstGraduate;  
}
```

```
public String getManagerApproval() {  
  
    return managerApproval;  
}
```

```
public void setManagerApproval(String  
managerApproval) {  
  
    this.managerApproval =  
    managerApproval;  
}
```

```
public double getAdmissionFee() {  
  
    return admissionFee;  
}
```

```
public void setAdmissionFee(double  
admissionFee) {  
  
    this.admissionFee =  
    admissionFee;  
}
```

```
public double getTuitionFee() {  
  
    return tuitionFee;  
}
```

```
public void setTuitionFee(double  
tuitionFee) {
```

```
    this.tuitionFee = tuitionFee;
```

```
}
```

```
public double getHostelFee() {
```

```
    return hostelFee;
```

```
}
```

```
public void setHostelFee(double  
hostelFee) {
```

```
    this.hostelFee = hostelFee;
```

```
}
```

```
public double getTotalCollegeFee() {
```

```
    return totalCollegeFee;
```

```
}
```

```
public void setTotalCollegeFee(double  
totalCollegeFee) {
```

```
    this.totalCollegeFee =  
    totalCollegeFee;
```

```
}
```

```
public String
```

```
getFinalStatusOfAdmission() {
```

```
    return finalStatusOfAdmission;
```

```
}
```

```
public void
```

```
setFinalStatusOfAdmission(String  
finalStatusOfAdmission) {
```

```

        this.finalStatusOfAdmission =
finalStatusOfAdmission;

    }

@Override
public String toString() {
    return "Student Admission
Details: [admissionId=" + admissionId +
", studentCode=" + studentCode + ",
dateOfCounseling="

+
dateOfCounseling +",
departmentName=" + departmentName +
", dateOfAdmission=" +
dateOfAdmission +",
preferCollegeHostel="

+
preferCollegeHostel +", firstGraduate=" +
firstGraduate +", managerApproval=" +
managerApproval

+",
admissionFee=" + admissionFee +",
tuitionFee=" + tuitionFee +",
hostelFee=" + hostelFee +",
totalCollegeFee=" + totalCollegeFee

+",
finalStatusOfAdmission=" +
finalStatusOfAdmission +"]";
}

}

```

## **ConstructionCostTimeEstimate**

## **CostAndTimeEstimation**

```
package com.cts.conctes.client;

import com.cts.conctes.service.ConstructionProjectEstimationService;

public class CostAndTimeEstimation {

    public static void main(String[] args)
    {
        ConstructionProjectEstimationService cpeService = new
        ConstructionProjectEstimationService();
        //WRITE YOUR CODE HERE

    }
}
```

## **CostAndTimeEstDAO**

```
package com.cts.conctes.dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Date;
```

```
import com.cts.conctes.exception.ConstructionEstimationException;
import com.cts.conctes.model.ConstructionProject;
import com.cts.conctes.util.ApplicationUtil;

public class CostAndTimeEstDAO {

    public static Connection connection =
        null;

    public boolean
    insertConstructionProject(ArrayList
    <ConstructionProject> constProjects)
    throws
    ConstructionEstimationException {
        boolean recordsAdded = false;

        //WRITE YOUR CODE HERE

        return recordsAdded;
    }

    public ArrayList <ConstructionProject>
    getConstructionProjectsData()
    {

        ArrayList <ConstructionProject>
        consApplicants = new
        ArrayList<ConstructionProject>();

        //WRITE YOUR CODE HERE

        return consApplicants;
    }
}
```

```
    }
```

```
}
```

## **DBConnectionManager**

```
package com.cts.conctes.dao;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;

import com.cts.conctes.exception.ConstructionEstimationException;

public class DBConnectionManager {

    private static Connection con = null;
    private static DBConnectionManager
    instance;

    public DBConnectionManager() throws
    ConstructionEstimationException
    {
```

```
//WRITE YOUR CODE HERE  
//return con;  
}  
  
public static DBConnectionManager  
getInstance() throws  
ConstructionEstimationException  
{  
  
//WRITE YOUR CODE HERE  
return instance;  
}  
  
public Connection getConnection()  
{  
  
//WRITE YOUR CODE HERE  
return con;  
}  
}
```

## ConstructionEstimationException

```
package com.cts.conctes.exception;  
  
public class ConstructionEstimationException extends Exception{  
  
String strMsg1;  
Throwable strMsg2;
```

```
        public  
        ConstructionEstimationException() {  
            super();  
        }  
  
    }  
  
}
```

## ConstructionProject

```
package com.cts.conctes.model;  
  
import java.util.Date;  
  
public class ConstructionProject {  
  
    String projectId;  
    Date plannedDOStart;  
    String typeOfProject;  
    String structure;  
    double areaInSqFt;  
    double estimatedCostInlac;  
    double estimatedTimeInMonths;  
  
    public ConstructionProject() {  
        super();  
    }  
}
```

```
public ConstructionProject(String  
projectId, Date plannedDOStart, String  
typeOfProject, String structure,
```

```
        double areaInSqFt,  
        double estimatedCostInLac, double  
        estimatedTimeInMonths) {
```

```
    super();
```

```
    this.projectId = projectId;
```

```
    this.plannedDOStart =  
    plannedDOStart;
```

```
    this.typeOfProject =  
    typeOfProject;
```

```
    this.structure = structure;
```

```
    this.areaInSqFt = areaInSqFt;
```

```
    this.estimatedCostInLac =  
    estimatedCostInLac;
```

```
    this.estimatedTimeInMonths =  
    estimatedTimeInMonths;
```

```
}
```

```
public String getProjectId() {
```

```
    return projectId;
```

```
}
```

```
public void setProjectId(String projectId)  
{
```

```
    this.projectId = projectId;
```

```
}
```

```
public Date getPlannedDOStart() {
```

```
    return plannedDOStart;
```

```
}
```

```
public void setPlannedDOStart(Date  
plannedDOStart) {
```

```
    this.plannedDOStart =
```

```
    plannedDOStart;
```

```
}
```

```
public String getTypeOfProject() {
```

```
    return typeOfProject;
```

```
}
```

```
public void setTypeOfProject(String  
typeOfProject) {
```

```
    this.typeOfProject =
```

```
    typeOfProject;
```

```
}
```

```
public String getStructure() {
```

```
    return structure;
```

```
}
```

```
public void setStructure(String structure)
```

```
{
```

```
    this.structure = structure;
```

```
}
```

```
public double getAreaInSqFt() {
```

```
    return areaInSqFt;
```

```
}
```

```
public void setAreaInSqFt(double  
areaInSqFt) {  
    this.areaInSqFt = areaInSqFt;  
}  
  
public double getEstimatedCostInLac() {  
    return estimatedCostInLac;  
}  
  
public void  
setEstimatedCostInLac(double  
estimatedCostInLac) {  
    this.estimatedCostInLac =  
estimatedCostInLac;  
}  
  
public double  
getEstimatedTimeInMonths() {  
    return estimatedTimeInMonths;  
}  
  
public void  
setEstimatedTimeInMonths(double  
estimatedTimeInMonths) {  
    this.estimatedTimeInMonths =  
estimatedTimeInMonths;  
}  
  
@Override  
public String toString() {  
    return "ConstructionProject  
[projectId=" + projectId + ",  

```

```

        plannedDOStart=" + plannedDOStart + ",
        typeOfProject="

                + typeOfProject +
        ", structure=" + structure + ",
        areaInSqFt=" + areaInSqFt + ",
        estimatedCostInLac="

                +
        estimatedCostInLac + ",
        estimatedTimeInMonths=" +
        estimatedTimeInMonths + "]";

    }

}

```

## **ConstructionProjectEstimationService**

```

package com.cts.conctes.service;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;

import com.cts.conctes.dao.CostAndTimeEstDAO;
import com.cts.conctes.exception.ConstructionEstimationException;
import com.cts.conctes.model.ConstructionProject;
import com.cts.conctes.util.ApplicationUtil;

```

```
public class ConstructionProjectEstimationService {  
  
    public static ArrayList  
    <ConstructionProject>  
    buildConstructionProjectList(List  
    <String> consProjectRecords) {  
  
        final String COMMADELIMITER =  
        ",";  
        ArrayList <ConstructionProject>  
        consProjectRecordList = new  
        ArrayList<ConstructionProject>();  
  
        //WRITE YOUR CODE HERE  
  
        return consProjectRecordList;  
    }  
  
    public boolean  
    addConstructionProjectDetails(String  
    inputFeed) throws  
    ConstructionEstimationException {  
  
        //WRITE YOUR CODE HERE  
        return false;  
    }  
  
    public static double[]  
    estimateTimeAndCostForConstruction(S
```

```
tring projectType,String  
structure,double areaInSqFt)  
{  
  
    double  
costEstimateInRs=0.0,timeEstimateInMo  
nths=0.0;  
  
    double costs[] =  
{costEstimateInRs,timeEstimateInMonth  
s};  
  
/*  
 * The Cost Estimate and  
 *  
 */
```

Based on the type of the Project & the Structure , according to the required

area of Construction, the cost & time have to be calculated based on the base

data available in the table provided in the use case document:

For eg. If the Project Type is “Commercial” and the structure

is “Shopping Complex” the cost incurred for the construction of

per sq. ft is Rs.2600 and the time taken for the construction of

the 1000 sq ft of the same project is 0.23 Months,

calculation has to be performed on the similar basis

i.e Pro rata basis depending upon the type and the area of construction.

\*/

```
//WRITE YOUR CODE HERE

        return costs;

    }

}
```

## **SkeletonValidator**

```
package com.cts.conctes.skeleton;

import java.lang.reflect.Method;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

import com.cts.conctes.model.ConstructionProject;

/**
 * @author 222805
 *
 * This class is used to verify if the Code Skeleton is intact and not modified by participants
 * thereby ensuring smooth auto
 * evaluation

```

```
*  
*/  
public class SkeletonValidator {  
  
    public SkeletonValidator() {  
  
        validateClassName("com.cts.con  
ctes.model.ConstructionProject");  
  
        validateClassName("com.cts.con  
ctes.dao.CostAndTimeEstDAO");  
  
        validateClassName("com.cts.con  
ctes.dao.DBConnectionManager");  
  
        validateClassName("com.cts.con  
ctes.exception.ConstructionEstimationE  
xception");  
  
        validateClassName("com.cts.con  
ctes.service.ConstructionProjectEstimati  
onService");  
  
        validateClassName("com.cts.con  
ctes.util.ApplicationUtil");  
  
        validateMethodSignature("insert  
ConstructionProject:boolean","com.cts.c  
ontctes.dao.CostAndTimeEstDAO");  
  
        validateMethodSignature("getIns  
tance:DBConnectionManager","com.cts.  
conctes.dao.DBConnectionManager");  
  
        validateMethodSignature("getCo
```

```
nnection:Connection","com.cts.conctes.  
dao.DBConnectionManager");
```

```
validateMethodSignature("build  
ConstructionProjectList:ArrayList,addCo  
nstructionProjectDetails:boolean,estima  
teTimeAndCostForConstruction:double[]]  
","com.cts.conctes.service.Construction  
ProjectEstimationService");
```

```
}
```

```
private static final Logger LOG =  
Logger.getLogger("SkeletonValidator");
```

```
protected final boolean  
validateClassName(String className) {
```

```
boolean iscorrect = false;
```

```
try {
```

```
Class.forName(className);
```

```
incorrect = true;
```

```
LOG.info("Class Name " +  
className + " is correct");
```

```
} catch (ClassNotFoundException  
e) {
```

```
LOG.log(Level.SEVERE,  
"You have changed either the " + "class  
name/package. Use the correct package  
"
```

```
+ "and  
class name as provided in the  
skeleton");  
  
} catch (Exception e) {  
  
    LOG.log(Level.SEVERE,  
            "There is  
an error in validating the " + "Class  
Name. Please manually verify that the "  
  
        + "Class name is same as  
skeleton before uploading");  
}  
  
return iscorrect;  
  
}  
  
protected final void  
validateMethodSignature(String  
methodWithExcptn, String className) {  
  
    Class cls = null;  
    try {  
  
        String[] actualmethods =  
methodWithExcptn.split(",");  
        boolean errorFlag = false;  
        String[] methodSignature;  
        String methodName =  
null;  
        String returnType = null;  
  
        for (String singleMethod :  
actualmethods) {
```

```
        boolean
foundMethod = false;

        methodSignature
= singleMethod.split ":";

        methodName =
methodSignature[0];

        returnType =
methodSignature[1];

        cls =
Class.forName(className);

        Method[]
methods = cls.getMethods();

        for (Method
findMethod : methods) {

            if
(methodName.equals(findMethod.getName())) {

                foundMethod = true;

                if
(!findMethod.getReturnType().getSimpleName().equals(returnType))) {

                    errorFlag = true;

                    LOG.log(Level.SEVERE, " You
have changed the " + "return type in " +
methodName

                    + "' method.
Please stick to the " + "skeleton
provided");



                }
            }
        }
    }
}
```

```
        LOG.info("Method signature of "  
+ methodName + " is valid");
```

```
}
```

```
}
```

```
}
```

```
if (!foundMethod)
```

```
{
```

```
    errorFlag =  
    true;
```

```
    LOG.log(Level.SEVERE, " Unable  
to find the given public method " +  
methodName
```

```
    + ". Do not change the " + "given  
public method name. " + "Verify it with  
the skeleton");
```

```
}
```

```
}
```

```
if (!errorFlag) {
```

```
    LOG.info("Method  
signature is valid");
```

```
}
```

```
} catch (Exception e) {
```

```
    LOG.log(Level.SEVERE,  
    " There is  
an error in validating the " + "method  
structure. Please manually verify that  
the "
```

```
        + "Method signature is same as  
the skeleton before uploading");  
    }  
}  
}
```

## ApplicationUtil

```
package com.cts.conctes.util;  
  
import java.io.BufferedReader;  
import java.io.FileInputStream;  
import java.io.FileNotFoundException;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
import java.util.StringTokenizer;  
  
import com.cts.conctes.exception.ConstructionEstimationException;  
  
public class ApplicationUtil {
```

```
public static List<String> readFile(String  
inputfeed) throws  
ConstructionEstimationException {  
  
    List<String> constructionProjects  
= new ArrayList<String>();
```

//WRITE YOUR CODE HERE

```
return constructionProjects;
```

```
}
```

```
public static java.sql.Date  
utilToSqlDateConverter(java.util.Date  
utDate) {
```

```
java.sql.Date sqlDate = null;
```

//WRITE YOUR CODE HERE

```
return sqlDate;
```

```
}
```

```
public static java.util.Date  
stringToDateConverter(String  
stringDate) {
```

```
Date strDate = new Date();
```

//WRITE YOUR CODE HERE

```
return strDate;
```

```
}
```

```
public static boolean  
checkIfCurrentFinYearProject(Date dos)
```

```
{  
    boolean flag = false;  
  
    int givenYear,givenMonth;  
  
    givenYear =  
(dos.getYear()+1900);  
  
    givenMonth = dos.getMonth();  
  
    Date curDate = new Date();  
  
    int curYear,curMonth;  
  
    curYear =  
(curDate.getYear()+1900);  
  
    curMonth = curDate.getMonth();  
  
    if( curYear == givenYear)  
    {  
  
        if(((curMonth  
        >=0)&&(curMonth <= 2)) &&  
        ((givenMonth >=0)&&(givenMonth <=  
        2)))  
        {  
  
            flag = true;  
        }  
  
        else if(((curMonth  
        >=3)&&(curMonth <= 11)) &&  
        ((givenMonth >=3)&&(givenMonth <=  
        11)))  
        {  
  
            flag = true;  
        }  
  
        else  
        {  
  
            flag = false;  
        }  
    }  
}
```

```
else if(curYear > givenYear)
```

```
{
```

```
    int dif = curYear -  
    givenYear;
```

```
    if(dif == 1)
```

```
{
```

```
    if(((curMonth  
    >=0)&&(curMonth <= 2)) &&  
    ((givenMonth >=3)&&(givenMonth <=  
    11)))
```

```
{
```

```
    flag = true;
```

```
}
```

```
    else if(((curMonth  
    >=3)&&(curMonth <= 11)) &&  
    ((givenMonth >=3)&&(givenMonth <=  
    11)))
```

```
{
```

```
    flag = false;
```

```
}
```

```
else
```

```
{
```

```
    flag = false;
```

```
}
```

```
}
```

```
else
```

```
{
```

```
    flag = false;
```

```
}
```

```
}
```

```
else if(curYear < givenYear)
```

```
{
```

```
int dif = givenYear-
curYear;

if(dif == 1)

{

    if(((curMonth
>=3)&&(curMonth <= 11)) &&
((givenMonth >=0)&&(givenMonth <=
2)))

{

    flag = true;

}

else if(((curMonth
>=3)&&(curMonth <= 11)) &&
((givenMonth >=3)&&(givenMonth <=
11)))

{

    flag = false;

}

else

{

    flag = false;

}
```

```
        return flag;  
  
    }  
  
}
```

## Insurance

CollectionAgency

```
import java.io.*;  
import java.sql.*;  
import java.util.*;
```

```
public class CollectionAgency
```

```
{
```

```
//write the required business logic  
methods as expected in the question  
description
```

```
public List<Payment>  
generatePaymentAmount(String  
filePath)
```

```
{
```

```
// fill your code here
```

```
}
```

```
public boolean validate(String policyId)
throws InvalidPolicyIdException

{

    // fill your code here


}

public void updatePolicyDetails(List
<Payment> paymentList)

{

    // fill your code here

}

}
```

## **DBHandler**

```
import java.io.*;
import java.sql.*;
import java.util.*;

public class DBHandler {

    //write the required business logic methods as expected in the question description

    public Connection establishConnection()

    {

        // fill your code here


    }

}
```

```
}
```

```
InvalidPolicyIdException
```

```
//make the required changes to this class so that InvalidPolicyIdException is of type  
exception.
```

```
public class InvalidPolicyIdException
```

```
{
```

```
//fill your code here
```

```
}
```

## Main

```
import java.util.*;
```

```
public class Main
```

```
{
```

```
    public static void main(String[] args)
```

```
{
```

```
    // fill your code here
```

```
}
```

```
}
```

```
Payment
```

```
public class Payment
{
    private String policyId;
    private double monthlyPremium;
    private int noOfMonths;
    private double paymentAmount;
    public String getPolicyId() {
        return policyId;
    }
    public void setPolicyId(String policyId) {
        this.policyId = policyId;
    }
    public double getMonthlyPremium() {
        return monthlyPremium;
    }
    public void setMonthlyPremium(double monthlyPremium) {
        this.monthlyPremium =
monthlyPremium;
    }
    public int getNoOfMonths() {
        return noOfMonths;
    }
    public void setNoOfMonths(int noOfMonths) {
        this.noOfMonths = noOfMonths;
    }
    public double getPaymentAmount() {
```

```
        return paymentAmount;  
    }  
  
    public void setPaymentAmount(double  
        paymentAmount) {  
        this.paymentAmount =  
            paymentAmount;  
    }  
  
    //Write the required business logic as  
    //expected in the question description  
    public void calculatePaymentAmount()  
    {  
        //fill your code here  
    }  
  
}
```

## 1. Check Sum of Odd Digits

Write a program to read a number , calculate the sum of odd digits (values) present in the given number.

Include a class **UserMainCode** with a static method **checkSum** which accepts a positive integer . The return type should be 1 if the sum is odd . In case the sum is even return -1 as output.

Create a class **Main** which would get the input as a positive integer and call the static method **checkSum** present in the UserMainCode.

### Input and Output Format:

Input consists of a positive integer n.

Refer sample output for formatting specifications.

### Sample Input 1:

56895

### Sample Output 1:

Sum of odd digits is odd.

### Sample Input 2:

84228

### Sample Output 2:

Sum of odd digits is even.

### MAIN:

```
import java.util.*;
public class Main {
    public static void main(String[] args)
    {
        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        int r=UserMainCode.checkSum(n);
        if(r==1)
        {
            System.out.println("The sum of odd digits are odd");
        }
        else
        {
            System.out.println("The sum of odd digits are even");
        }
        s.close();
    }
}
```

### USERMAINCODE:

```
public class UserMainCode {
    public static int checkSum(int n)
    {
        int n1;
        int sum=0;
        int r;
        while(n!=0)
        {
            n1=n%10;
            if(n1%2!=0)
            {
                sum=sum+n1;
```

```

        }
        n=n/10;
    }
    if(sum%2==0)
    {
        r=-1;
    }
    else
    {
        r=1;
    }
    return r;
}
}

```

## 2. Number Validation

Write a program to read a string of 10 digit number , check whether the string contains a 10 digit number in the format XXX-XXX-XXXX where 'X' is a digit.

Include a class **UserMainCode** with a static method **validateNumber** which accepts a string as input .

The return type of the output should be 1 if the string meets the above specified format . In case the number does not meet the specified format then return -1 as output.

Create a class **Main** which would get the input as a String of numbers and call the static method**validateNumber** present in the UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output is a string specifying the given string is valid or not .

Refer sample output for formatting specifications.

#### **Sample Input 1:**

123-456-7895

#### **Sample Output 1:**

Valid number format

#### **Sample Input 2:**

-123-12344322

#### **Sample Output 2:**

Invalid number format

### **MAIN:**

```

import java.util.*;
public class Main {
    public static void main(String[] args)
    {
        Scanner s=new Scanner(System.in);
        String number=s.next();
        int r=UserMainCode.validateNumber(number);
        if(r==1)
        {
            System.out.println("Valid number format");
        }
    }
}

```

```

        else
        {
            System.out.println("Invalid number format");
        }
        s.close();
    }

}

```

### USERMAINCODE:

```

import java.util.*;
public class UserMainCode {
    public static int validateNumber(String number)
    {
        int b;
        if(number.matches("[0-9]{3}[-]{1}[0-9]{3}[-]{1}[0-9]{4}"))
        {
            b=1;
        }
        else
        {
            b=0;
        }
        return b;
    }
}

```

### **3. Sum of Squares of Even Digits**

Write a program to read a number , calculate the sum of squares of even digits (values) present in the given number.

Include a class **UserMainCode** with a static method **sumOfSquaresOfEvenDigits**

which

accepts a positive integer . The return type (integer) should be the sum of squares of the even digits.

Create a class **Main** which would get the input as a positive integer and call the static method sumOfSquaresOfEvenDigits present in the UserMainCode.

#### **Input and Output Format:**

Input consists of a positive integer n.

Output is a single integer .

Refer sample output for formatting specifications.

#### **Sample Input 1:**

56895

#### **Sample Output 1:**

100

### MAIN:

```

import java.util.*;
public class Main {
    public static void main(String[] args)
    {

```

```

        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        System.out.println(UserMainCode.sumOfSquaresOfEvenDigits(n));
        s.close();
    }

}

```

### **USERMAINCODE:**

```

public class UserMainCode {
    public static int sumOfSquaresOfEvenDigits(int n)
    {
        int n1=0;
        int sum=0;
        while(n!=0)
        {
            n1=n%10;
            if(n1%2==0)
            {
                sum+=n1*n1;
            }
            n=n/10;
        }
        return sum;
    }
}

```

## **4. Fetching Middle Characters from String**

Write a program to read a string of even length and to fetch two middle most characters from the input string and return it as string output.

Include a class **UserMainCode** with a static method **getMiddleChars** which accepts a string

of even length as input . The return type is a string which should be the middle characters of the string.

Create a class **Main** which would get the input as a string and call the static method **getMiddleChars**present in the UserMainCode.

### **Input and Output Format:**

Input consists of a string of even length.

Output is a string .

Refer sample output for formatting specifications.

### **Sample Input 1:**

this

### **Sample Output 1:**

hi

### **Sample Input 1:**

Hell

### **Sample Output 1:**

el

### **MAIN:**

```
import java.util.*;
public class Main {
    public static void main(String[] args)
    {
        Scanner s=new Scanner(System.in);
        String str=s.nextLine();
        System.out.println(UserMainCode.getMiddleChars(str));
        s.close();
    }
}
```

### **USERMAINCODE:**

```
import java.util.*;
public class UserMainCode {
    public static String getMiddleChars(String str)
    {
        StringBuffer sb=new StringBuffer();
        if(str.length()%2==0)
        {
            sb.append(str.substring((str.length()/2)-1,(str.length()/2)+1));
        }
        return sb.toString();
    }
}
```

## **5. Check Characters in a String**

Write a program to read a string and to test whether first and last character are same. The string is said to be valid if the 1st and last character are the same. Else the string is said to be invalid.

Include a class **UserMainCode** with a static method **checkCharacters** which accepts a string as input .

The return type of this method is an int. Output should be 1 if the first character and last character are same . If they are different then return -1 as output.

Create a class **Main** which would get the input as a string and call the static method **checkCharacters** present in the UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output is a string saying characters are same or not .

Refer sample output for formatting specifications.

### **Sample Input 1:**

the picture was great

### **Sample Output 1:**

Valid

### **Sample Input 1:**

this

### **Sample Output 1:**

Invalid

#### **MAIN:**

```

import java.util.*;
public class main {
    public static void main(String[] args)
    {
        Scanner s=new Scanner(System.in);
        String input=s.nextLine();
        int r=UserMainCode.checkCharacters(input);
        if(r==1)
        {
            System.out.println("Valid");
        }
        else
        {
            System.out.println("Invalid");
        }
        s.close();
    }
}

```

#### **USERMAINCODE:**

```

import java.util.*;
public class UserMainCode {
    public static int checkCharacters(String input)
    {
        int r;
        StringTokenizer t = new StringTokenizer(input, " ");
        String s = t.nextToken();
        String s1 =s ;
        while(t.hasMoreTokens())
        {
            s1 = t.nextToken();
        }
        if(s.charAt(0) == s1.charAt(s1.length()-1))
            r=1;
        else
            r=0;
        return r;
    }
}

```

## **6. Forming New Word from a String**

Write a program to read a string and a positive integer n as input and construct a string with first n and last n characters in the given string.

Include a class **UserMainCode** with a static method **formNewWord** which accepts a string

and positive integer .

The return type of the output should be a string (value) of first n character and last n character.

Create a class **Main** which would get the input as a string and integer n and call the static method **formNewWord** present in the UserMainCode.

### **Input and Output Format:**

Input consists of a string of even length.

Output is a string .

Note: The given string length must be  $\geq 2n$ .

Refer sample output for formatting specifications.

### **Sample Input 1:**

California

3

### **Sample Output 1:**

Calnia

### **Sample Input 2:**

this

1

### **Sample Output 2:**

Ts

### **MAIN:**

```
import java.util.*;
public class Main {
    public static void main(String[] args)
    {
        Scanner s=new Scanner(System.in);
        String s1=s.nextLine();
        int n1=s.nextInt();
        System.out.println(UserMainCode.formNewWord(s1,n1));
        s.close();
    }
}
```

### **USERMAINCODE:**

```
import java.util.*;
public class UserMainCode {
    public static String formNewWord(String s1,int n)
    {
        String s = new String();
        if(s1.length()>n)
        {
            s = s1.substring(0,n) + s1.substring(s1.length()-n, s1.length());
            return s;
        }
        else
            return null;
    }
}
```

## 7. Reversing a Number

Write a program to read a positive number as input and to get the reverse of the given number and return it as output.

Include a class **UserMainCode** with a static method **reverseNumber** which accepts a positive integer .

The return type is an integer value which is the reverse of the given number.

Create a **Main** class which gets the input as a integer and call the static method **reverseNumber** present in the **UserMainCode**

### Input and Output Format:

Input consists of a positive integer.

Output is an integer .

Refer sample output for formatting specifications.

### Sample Input 1:

543

### Sample Output 1:

345

### Sample Input 1:

1111

### Sample Output 1:

1111

### MAIN:

```
import java.util.*;
public class Main {
    public static void main(String[] args)
    {
        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        System.out.println(UserMainCode.reverseNumber(n));
        s.close();
    }
}
```

### USERMAINCODE:

```
public class UserMainCode {
    public static int reverseNumber(int n)
    {
        int a,r=0;
        while(n!=0)
        {
            a=n%10;
            r=r*10+a;
            n=n/10;
        }
        return r;
    }
}
```

## 8. Array List Sorting and Merging

Write a code to read two int array lists of size 5 each as input and to merge the two arrayLists, sort the merged arrayList in ascending order and fetch the elements at 2nd, 6th and 8th index into a new arrayList and return the final ArrayList.

Include a class **UserMainCode** with a static method **sortMergedArrayList** which accepts 2 ArrayLists.

The return type is an ArrayList with elements from 2,6 and 8th index position .Array index starts from position 0.

Create a **Main** class which gets two array list of size 5 as input and call the static method **sortMergedArrayList** present in the **UserMainCode**.

**Input and Output Format:**

Input consists of two array lists of size 5.

Output is an array list .

Note - The first element is at index 0.

Refer sample output for formatting specifications.

**Sample Input 1:**

```
3  
1  
17  
11  
19  
5  
2  
7  
6  
20
```

**Sample Output 1:**

```
3  
11  
19
```

**Sample Input 2:**

```
1  
2  
3  
4  
5  
6  
7  
8  
9  
10
```

**Sample Output 2:**

```
3  
7  
9
```

**Main:**

```
import java.util.*;  
public class Main {  
    public static void main(String[] args)
```

```

{
    Scanner s=new Scanner(System.in);
    ArrayList<Integer> list1=new ArrayList<Integer>();
    ArrayList<Integer> list2=new ArrayList<Integer>();
    ArrayList<Integer> newlist=new ArrayList<Integer>();
    for (int i = 0; i < 5; i++)
    {
        list1.add(s.nextInt());
    }
    for (int i = 0; i < 5; i++)
    {
        list2.add(s.nextInt());
    }
    newlist=UserMainCode.sortMergedArrayList(list1,list2);
    for (int i = 0; i < 3; i++)
    {
        System.out.println(newlist.get(i));
    }
    s.close();
}
}

```

### UserMainCode:

```

import java.util.*;
public class UserMainCode {
    public static ArrayList<Integer> sortMergedArrayList(ArrayList<Integer>
list1,ArrayList<Integer> list2)
    {
        list1.addAll(list2);
        Collections.sort(list1);
        ArrayList<Integer> ans=new ArrayList<Integer>();
        ans.add(list1.get(2));
        ans.add(list1.get(6));
        ans.add(list1.get(8));
        return ans;
    }
}

```

## 9. Validating Date Format

Obtain a date string in the format dd/mm/yyyy. Write code to validate the given date against the given format.

Include a class **UserMainCode** with a static method **validateDate** which accepts a string . The return type of the validateDate method is 1 if the given date format matches the specified format , If the validation fails return the output as -1.

Create a **Main** class which gets date string as an input and call the static method **validateDate** present in the **UserMainCode**.

### **Input and Output Format:**

Input is a string .

Refer sample output for formatting specifications

### **Sample Input 1:**

12/06/1987

### **Sample Output 1:**

Valid date format

### **Sample Input 2:**

03/1/1987

### **Sample Output 2:**

Invalid date format

#### **Main:**

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        String s1=sc.nextLine();
        int b=UserMainCode.ValidateDate(s1);
        if(b==1){
            System.out.println("Valid date format");
        }
        else{
            System.out.println("Invalid date format");
        }
        sc.close();
    }
}
```

#### **UserMainCode:**

```
import java.util.*;
import java.text.*;
public class UserMainCode{
    public static int ValidateDate(String s1) {
        if(s1.matches("[0-9]{2}[/]{1}[0-9]{2}[/]{1}[0-9]{4}"))
        {
            SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
            sdf.setLenient(false);
            try {
                Date d1=sdf.parse(s1);
                return 1;
            } catch (ParseException e) {
                return -1;
            }
        }
        else{
            return -1;}}}
```

## **10. Validate Time**

Obtain a time string as input in the following format 'hh:mm am' or 'hh:mm pm'. Write code to validate it using the following rules:

- It should be a valid time in 12 hrs format
- It should have case insensitive AM or PM

Include a class **UserMainCode** with a static method **validateTime** which accepts a string. If the given time is as per the given rules then return 1 else return -1. If the value returned is 1 then print as valid time else print as Invalid time.

Create a **Main** class which gets time(string value) as an input and call the static method **validateTime** present in the **UserMainCode**.

**Input and Output Format:**

Input is a string .

Output is a string .

**Sample Input 1:**

09:59 pm

**Sample Output 1:**

Valid time

**Sample Input 2:**

10:70 AM

**Sample Output 2:**

Invalid time

**Main:**

```
import java.util.*;
public class Main{
public static void main(String []args){
Scanner sc=new Scanner(System.in);
String str=sc.nextLine();
int b=UserMainCode.ValidateTime(str);
if(b==1){
    System.out.println("Valid time");
}
else{
    System.out.println("Invalid time");
}
sc.close();
}}
```

**UserMainCode:**

```
import java.text.*;
import java.util.*;
public class UserMainCode{
public static int ValidateTime(String str){
StringTokenizer st=new StringTokenizer(str,":");
if(st.countTokens()==3)
{
SimpleDateFormat sdf1 = new SimpleDateFormat("h:mm:ss a");
sdf1.setLenient(false);
try
{
Date d2=sdf1.parse(str);
return 1;
}
catch(Exception e)
{
    return -1;
}}
else
{
SimpleDateFormat sdf = new SimpleDateFormat("h:mm a");
sdf.setLenient(false);
try
{
Date d1=sdf.parse(str);
return 1;
}
}}
```

```

        catch(Exception e){
            return -1;
       }}}
    
```

## 11. String Encryption

Given an input as string and write code to encrypt the given string using following rules and return the encrypted string:

1. Replace the characters at odd positions by next character in alphabet.
2. Leave the characters at even positions unchanged.

Note:

- If an odd position character is 'z' replace it by 'a'.
- Assume the first character in the string is at position 1.

Include a class **UserMainCode** with a static method **encrypt** which accepts a string.

The return type of the output is the encrypted string.

Create a **Main** class which gets string as an input and call the static method **encrypt** present

in the **UserMainCode**.

### **Input and Output Format:**

Input is a string .

Output is a string.

#### **Sample Input 1:**

curiosity

#### **Sample Output 1:**

dusipsjtz

#### **Sample Input 2:**

zzzz

#### **Sample Output 2:**

Azaz

### **Main:**

```

import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.next();
        System.out.println(UserMainCode.encrypt(s1));
        s.close();
    }
}
    
```

### **UserMainCode:**

```

public class UserMainCode{
    public static String encrypt(String s1) {
        StringBuffer sb=new StringBuffer();
        for(int i=0;i<s1.length();i++){
            char c=s1.charAt(i);
            if(i%2==0){
                if(c==122)
                    if((c==122)&&(i==0)){
                        c='A';
                    }else
                        c=(char) (c-25);
            }else{
                c=c+1;
            }
            sb.append(c);
        }
        return sb.toString();
    }
}
    
```

```

c=(char) (c+1);
sb.append(c);
else
sb.append(c);
return sb.toString();
}}

```

## 12. Password Validation

Given a method with a password in string format as input. Write code to validate the password using following rules:

- Must contain at least one digit
- Must contain at least one of the following special characters @, #, \$
- # Length should be between 6 to 20 characters.

Include a class **UserMainCode** with a static method **validatePassword** which accepts a password string as input.

If the password is as per the given rules return 1 else return -1. If the return value is 1 then print valid password else print as invalid password.

Create a **Main** class which gets string as an input and call the static method **validatePassword** present in the **UserMainCode**.

### **Input and Output Format:**

Input is a string .

Output is a string .

### **Sample Input 1:**

%Dhoom%

### **Sample Output 1:**

Invalid password

### **Sample Input 2:**

#@6Don

### **Sample Output 2:**

Valid password

### **Main:**

```

import java.util.*;
public class Main {
public static void main(String[] args){
    Scanner s=new Scanner(System.in);
    String password=s.next();
    int b=UserMainCode.ValidatePassword(password);
    if(b==1){
        System.out.println("Valid Password");
    }
    else{
        System.out.println("Invalid Password");
    }
    s.close();
}}

```

### **UserMainCode:**

```

public class UserMainCode{
public static int ValidatePassword(String password){
if(password.matches("[0-9]{1,}.*") && password.matches(".*[@$]{1,}.*"))
&& password.length()>=6 && password.length()<=20)
{
}

```

```

    return 1;
}
else
{
return -1;
}}}

```

### 13. Removing vowels from String

Given a method with string input. Write code to remove vowels from even position in the string.

Include a class **UserMainCode** with a static method **removeEvenVowels** which accepts a string as input.

The return type of the output is string after removing all the vowels.

Create a **Main** class which gets string as an input and call the static method **removeEvenVowels** present in the **UserMainCode**.

#### **Input and Output Format:**

Input is a string .

Output is a string .

Assume the first character is at position 1 in the given string.

#### **Sample Input 1:**

commitment

#### **Sample Output 1:**

cmmitmmt

#### **Sample Input 2:**

capacity

#### **Sample Output 2:**

Cpcty

#### **Main:**

```

import java.util.*;
public class Main {
public static void main(String[] args) {
Scanner s=new Scanner(System.in);
String s1=s.nextLine();
System.out.println(UserMainCode.removeEvenVowels(s1));
s.close();
}}

```

#### **UserMainCode:**

```

public class UserMainCode{
public static String removeEvenVowels(String s1) {
StringBuffer sb1=new StringBuffer();
for(int i=0;i<s1.length();i++)
if((i%2)==0)
sb1.append(s1.charAt(i));
else if((i%2)!=0)
if(s1.charAt(i)!='a' && s1.charAt(i)!='e' &&
s1.charAt(i)!='i' && s1.charAt(i)!='o' && s1.charAt(i)!='u')
if(s1.charAt(i)!='A' && s1.charAt(i)!='E' &&
s1.charAt(i)!='I' && s1.charAt(i)!='O' && s1.charAt(i)!='U')
sb1.append(s1.charAt(i));
return sb1.toString();
}}

```

## 14. Sum of Powers of elements in an array

Given a method with an int array. Write code to find the power of each individual element according to its position index, add them up and return as output.

Include a class **UserMainCode** with a static method **getSumOfPower** which accepts an integer array as input.

The return type of the output is an integer which is the sum powers of each element in the array.

Create a **Main** class which gets integer array as an input and call the static method **getSumOfPower** present in the **UserMainCode**.

### Input and Output Format:

Input is an integer array. First element corresponds to the number(n) of elements in an array. The next inputs corresponds to each element in an array.

Output is an integer .

### Sample Input 1:

```
4  
3  
6  
2  
1
```

### Sample Output 1:

```
12
```

### Sample Input 2:

```
4  
5  
3  
7  
2
```

### Sample Output 2:

```
61
```

#### Main:

```
import java.util.Scanner;  
public class Main{  
    public static void main(String args[]){  
        Scanner sc=new Scanner(System.in);  
        int n=sc.nextInt();  
        int a[]={};  
        for(int i=0;i<n;i++)  
        {  
            a[i]=sc.nextInt();  
        }  
        System.out.println(UserMainCode.getSumOfPower(n,a));  
        sc.close();  
    }}
```

#### UserMainCode:

```
public class UserMainCode{  
    public static int getSumOfPower(int n,int[]a)  
    {{  
        int sum=0;  
        for(int i=0;i<n;i++)
```

```

sum=(int)(sum+Math.pow(a[i], i));
return sum;
}}}

```

## 15.Difference between largest and smallest elements in an array

Given a method taking an int array having size more than or equal to 1 as input. Write code to return the difference between the largest and smallest elements in the array. If there is only one element in the array return the same element as output.

Include a class **UserMainCode** with a static method **getBigDiff** which accepts a integer array as input.

The return type of the output is an integer which is the difference between the largest and smallest elements in the array.

Create a **Main** class which gets integer array as an input and call the static method **getBigDiff** present in the **UserMainCode**.

### **Input and Output Format:**

Input is an integer array. First element in the input represents the number of elements in an array.

Size of the array must be  $\geq 1$

Output is an integer which is the difference between the largest and smallest element in an array.

### **Sample Input 1:**

```

4
3
6
2
1

```

### **Sample Output 1:**

```
5
```

### **Sample Input 2:**

```

4
5
3
7
2

```

### **Sample Output 2:**

```
5
```

### **Main:**

```

import java.util.*;
public class Main {
    public static void main(String args[]){
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        int a[]={};
        for(int i=0;i<n;i++)
        {
            a[i]=sc.nextInt();
        }
        System.out.println(UserMainCode.getBigDiff(a,n));
        sc.close();
    }
}

```

```

        }}
UserMainCode:
import java.util.*;
public class UserMainCode{
public static int getBigDiff(int [] a,int n)
{
    Arrays.sort(a);
    int n1=a[a.length-1]-a[0];
    return n1;
}
}

```

## **16.Find the element position in a reversed string array**

Given a method with an array of strings and one string variable as input. Write code to sort the given array in reverse alphabetical order and return the postion of the given string in the array.

Include a class **UserMainCode** with a static method **getElementPosition** which accepts an

array of strings and a string variable as input.

The return type of the output is an integer which is the position of given string value from the array.

Create a **Main** class which gets string array and a string variable as an input and call the static method **getElementPosition** present in the **UserMainCode**.

### **Input and Output Format:**

Input is an string array. First element in the input represents the size the array

Assume the position of first element is 1.

Output is an integer which is the position of the string variable

### **Sample Input 1:**

```

4
red
green
blue
ivory
ivory

```

### **Sample Output 1:**

```

2

```

### **Sample Input 2:**

```

3
grape
mango
apple
apple

```

### **Sample Output 2:**

```

3

```

### **Main:**

```

import java.util.*;
public class Main {
public static void main(String[] args) {
Scanner sc=new Scanner(System.in);
int fr=sc.nextInt();

```

```

String a[] = new String[fr];
for(int i=0;i<fr;i++)
{
a[i]=sc.next();
}
String ba=sc.next();
UserMainCode.getElementPosition(a,ba);
sc.close();
}

UserMainCode:

import java.util.*;
public class UserMainCode{
public static void getElementPosition(String[] a, String b) {
ArrayList<String>al=new ArrayList<String>();
for(int i=0;i<a.length;i++)
{
al.add(a[i]);
}
Collections.sort(al);
Collections.reverse(al);
for(int i=0;i<al.size();i++)
{
if(b.equals(al.get(i)))
{
System.out.println(i+1);
}}}
}

```

## 17.generate the series

Given a method taking an odd positive Integer number as input. Write code to evaluate the following series:

1+3-5+7-9...+/-n.

Include a class **UserMainCode** with a static method **addSeries** which accepts a positive integer .

The return type of the output should be an integer .

Create a class **Main** which would get the input as a positive integer and call the static method **addSeries** present in the UserMainCode.

### **Input and Output Format:**

Input consists of a positive integer n.

Output is a single integer .

Refer sample output for formatting specifications.

**Sample Input 1:**

9

**Sample Output 1:**

-3

**Sample Input 2:**

11

**Sample Output 2:**

8

**Main**

```
import java.util.*;

public class Main {

    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        System.out.println(UserMainCode.addSeries(n));
        s.close();
    }
}
```

**UserMainCode**

```
import java.util.ArrayList;
import java.util.List;

public class UserMainCode {
```

```

public static int addSeries(int n){

    List<Integer> l1=new ArrayList<Integer>();

    for(int i=1;i<=n;i++)

        if(i%2!=0)

            l1.add(i);

        int n1=l1.get(0);

        for(int i=1;i<l1.size();i++)

            if(i%2!=0)

                n1=n1+l1.get(i);

            else

                n1=n1-l1.get(i);

        return n1;

    }

}

```

## **18.Calculate Electricity Bill**

Given a method calculateElectricityBill() with three inputs. Write code to calculate the current bill.

Include a class **UserMainCode** with a static method **calculateElectricityBill** which accepts 3

inputs .The return type of the output should be an integer .

Create a class **Main** which would get the inputs and call the static method **calculateElectricityBill** present in the UserMainCode.

### **Input and Output Format:**

Input consist of 3 integers.

First input is previous reading, second input is current reading and last input is per unit charge.

Reading Format - XXXXXAAAAA where XXXXX is consumer number and AAAAA is meter

reading.

Output is a single integer corresponding to the current bill.

Refer sample output for formatting specifications.

**Sample Input 1:**

ABC2012345

ABC2012660

4

**Sample Output 1:**

**1260**

**Sample Input 2:**

ABCDE11111

ABCDE11222

3

**Sample Output 2:**

333

**Main**

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);

        String input1=s.next();
        String input2=s.next();
        int input3=s.nextInt();
```

```

        System.out.println(UserMainCode.calculateElectricityBill(input1,input2,input3));

        s.close();

    }

}

```

### **UserMainCode**

```

public class UserMainCode {

    public static int calculateElectricityBill(String input1, String input2,
int input3)

    {

        int n1=Integer.parseInt(input1.substring(5, input1.length()));

        int n2=Integer.parseInt(input2.substring(5, input2.length()));

        int n=Math.abs((n2-n1)*input3);

        return n;

    }

}

```

### **19.Sum of Digits in a String**

Write code to get the sum of all the digits present in the given string.

Include a class **UserMainCode** with a static method **sumOfDigits** which accepts string input.

Return the sum as output. If there is no digit in the given string return -1 as output.

Create a class **Main** which would get the input and call the static

method **sumOfDigits** present in the UserMainCode.

#### **Input and Output Format:**

Input consists of a string.

Output is a single integer which is the sum of digits in a given string.

Refer sample output for formatting specifications.

**Sample Input 1:**

good23bad4

**Sample Output 1:**

9

**Sample Input 2:**

good

**Sample Output 2:**

-1

**Main**

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.next();
        UserMainCode.sumOfDigits(s1);
        s.close();
    }
}
```

**UserMainCode**

```
public class UserMainCode {
    public static void sumOfDigits(String s1) {
        int sum=0;
        for(int i=0;i<s1.length();i++)
        {

```

```

        char a=s1.charAt(i);

        if(Character.isDigit(a))

        {

            int b=Integer.parseInt(String.valueOf(a));

            sum=sum+b;

        }

    }

    if(sum==0)

    {

        System.out.println(-1);

    }

    else

        System.out.println(sum);

}

}

```

## 20.String Concatenation

Write code to get two strings as input and If strings are of same length simply append them together and return the final string. If given strings are of different length, remove starting characters from the longer string so that both strings are of same length then append them together and return the final string.

Include a class **UserMainCode** with a static method **concatstring** which accepts two string input.

The return type of the output is a string which is the concatenated string.

Create a class **Main** which would get the input and call the static

method **concatstring** present in the UserMainCode.

### **Input and Output Format:**

Input consists of two strings.

Output is a string.

Refer sample output for formatting specifications.

### **Sample Input 1:**

Hello

hi

### **Sample Output 1:**

lohi

### **Sample Input 2:**

Hello

Delhi

### **Sample Output 2:**

HelloDelhi

## **Main**

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);

        String s1=s.next();
        String s2=s.next();

        UserMainCode.concatstring(s1,s2);

        s.close();
    }
}
```

```
}
```

### **} UserMainCode**

```
public class UserMainCode {  
  
    public static void concatstring(String s1, String s2) {  
  
        StringBuffer sb=new StringBuffer();  
  
        int l1=s1.length();  
  
        int l2=s2.length();  
  
        if(l1==l2)  
        {  
            sb.append(s1).append(s2);  
        }  
  
        else if(l1>l2)  
        {  
            sb.append(s1.substring(s1.length()-  
s2.length(),s1.length())).append(s2);  
        }  
  
        else if(l1<l2)  
        {  
            sb.append(s1).append(s2.substring(s2.length()-  
s1.length(),s2.length()));  
        }  
  
        System.out.println(sb);  
    }  
  
}
```

---

### **21. Color Code**

Write a program to read a string and validate whether the given string is a valid color code based on the following rules:

- Must start with "#" symbol
- Must contain six characters after #
- It may contain alphabets from A-F or digits from 0-9

Include a class **UserMainCode** with a static method **validateColorCode** which accepts a string. The return type (integer) should return 1 if the color is as per the rules else return -1.

Create a Class Main which would be used to accept a String and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of a string (Valid or Invalid).

Refer sample output for formatting specifications.

### **Sample Input 1:**

#FF9922

### **Sample Output 1:**

Valid

### **Sample Input 2:**

#FF9(22

### **Sample Output 2:**

Invalid

### **Main**

```
import java.util.*;  
  
public class Main {
```

```

public static void main(String[] args) {
    Scanner s=new Scanner(System.in);
    String s1=s.next();
    int b=UserMainCode.validateColorCode(s1);
    if(b==1)
        System.out.println("Valid");
    else
        System.out.println("Invalid");
    s.close();
}
}

```

## UserMainCode

```

public class UserMainCode {
    public static int validateColorCode(String s1) {
        int b=0,b1=0;
        String s2=s1.substring(1,s1.length());
        if(s1.length()==7)
            if(s1.charAt(0)=='#')
                b1=1;
        if(b1==1){
            /*for(int i=0;i<s2.length();i++){
                char c=s2.charAt(i);
                if(c!='#')
                    }*/
            if(s2.matches("[A-F0-9]{1,}"))
                b=1;
        }
    }
}

```

```
        else  
        b=-1;  
        //break;  
    }  
    return b;  
}  
}
```

---

## 22.Three Digits

Write a program to read a string and check if the given string is in the format "CTS-XXX" where XXX is a three digit number.

Include a class **UserMainCode** with a static method **validatestrings** which accepts a string.

The return type (integer) should return 1 if the string format is correct else return -1.

Create a Class Main which would be used to accept a String and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of a string (Valid or Invalid).

Refer sample output for formatting specifications.

### **Sample Input 1:**

CTS-215

### **Sample Output 1:**

Valid

### **Sample Input 2:**

CTS-2L5

## Sample Output 2:

Invalid

### Main

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner s=new Scanner(System.in);

        String s1=s.next();

        int b=UserMainCode.validatestrings(s1);

        if(b==1){

            System.out.println("Valid");

        }

        else

            System.out.println("Invalid");

        s.close();

    }

}
```

### UserMainCode

```
public class UserMainCode {

    public static int validatestrings(String s1) {

        int res=0;

        if(s1.matches("(CTS)[-]{1}[0-9]{3}"))

        {

            res=1;

        }

        else
```

```
    res=-1;

    return res;

}

}
```

## 23.Removing Keys from HashMap

Given a method with a `HashMap<Integer,string>` as input. Write code to remove all the entries having keys multiple of 4 and return the size of the final hashmap.

Include a class **UserMainCode** with a static method **sizeOfResultandHashMap** which accepts hashmap as input.

The return type of the output is an integer which is the size of the resultant hashmap.

Create a class **Main** which would get the input and call the static method **sizeOfResultandHashMap** present in the UserMainCode.

### **Input and Output Format:**

First input corresponds to the size of the hashmap.

Input consists of a `hashmap<integer,string>`.

Output is an integer which is the size of the hashmap.

Refer sample output for formatting specifications.

### **Sample Input 1:**

3

2

hi

4

hello

12

hello world

**Sample Output 1:**

1

**Sample Input 2:**

3

2

hi

4

sdfsdf

3

asdf

**Sample Output 2:**

2

**Main**

```
import java.util.*;

public class Main {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        int s=sc.nextInt();

        HashMap<Integer, String>hm=new HashMap<Integer, String>();

        for(int i=0;i<s;i++){

            hm.put((sc.nextInt()),(sc.next()));

        }

        System.out.println(UserMainCode.sizeOfResultandHashMap(hm));
    }
}
```

```
sc.close();
```

} }

## UserMainCode

```
import java.util.HashMap;  
import java.util.Iterator;
```

```
public class UserMainCode {  
  
    public static int sizeOfResultandHashMap(HashMap<Integer, String> hm) {  
  
        int count=0;  
  
        Iterator<Integer>itr=hm.keySet().iterator();  
  
        while(itr.hasNext())  
  
        {  
  
            int n=itr.next();  
  
            if(n%4!=0)  
  
            {  
  
                count++;  
  
            }  
  
        }  
  
        return count;  
  
    }  
  
}
```

## 24. Largest Element

Write a program to read an int array of odd length, compare the first, middle and the last

elements in the array and return the largest. If there is only one element in the array return the same element.

Include a class **UserMainCode** with a static method **checkLargestAmongCorner** which

accepts an int arrayThe return type (integer) should return the largest element among the first, middle and the last elements.

Create a Class Main which would be used to accept Input array and call the static method present in UserMainCode.

Assume maximum length of array is 20.

### **Input and Output Format:**

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

Output consists of a single Integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

5  
2  
3  
8  
4  
5

### **Sample Output 1:**

8

### **Main**

```
import java.util.*;  
  
public class Main {
```

```

public static void main(String[] args) {

    Scanner s=new Scanner(System.in);

    int n=s.nextInt();

    int a[]={new int[n];

    for(int i=0;i<n;i++){

        a[i]=s.nextInt();

    }

    System.out.println(UserMainCode.checkLargestAmongCorner(a));

    s.close();

}

}

```

## UserMainCode

```

public class UserMainCode {

    public static int checkLargestAmongCorner(int []a)

    {

        int max=0;

        int x,y,z;

        x=a[0];

        y=a[a.length/2];

        z=a[a.length-1];

        if(x>y && x>z)

            max=x;

        else if(y>x && y>z)

            max=y;

```

```

else if(z>x && z>y)
    max=z;
return max;
}
}

```

## 25. nCr

Write a program to calculate the ways in which r elements can be selected from n population, using nCr formula  $nCr = n!/r!(n-r)!$  where first input being n and second input being r.

**Note1 :** n! factorial can be achieved using given formula  $n!=nx(n-1)x(n-2)x..3x2x1$ .

**Note2 :**  $0! = 1$ .

Example  $5!=5x4x3x2x1=120$

Include a class **UserMainCode** with a static method **calculateNcr** which accepts two integers. The return type (integer) should return the value of nCr.

Create a Class Main which would be used to accept Input elements and call the static method present in UserMainCode.

### Input and Output Format:

Input consists of 2 integers. The first integer corresponds to n, the second integer corresponds to r.

Output consists of a single Integer.

Refer sample output for formatting specifications.

### Sample Input 1:

4

3

## **Sample Output 1:**

4

### **Main**

```
import java.util.*;

public class Main {

    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);

        int n=s.nextInt();
        int r=s.nextInt();

        System.out.println(UserMainCode.calculateNcr(n,r));
    }
}
```

### **UserMainCode**

```
public class UserMainCode {

    public static int calculateNcr(int n, int r) {
        int fact=1,fact1=1,fact2=1;

        for(int i=1;i<=n;i++)
        {
            fact=fact*i;
        }

        //System.out.println(fact);

        for(int i=1;i<=r;i++)
        {
            fact1=fact1*i;
        }
    }
}
```

```

        }

        //System.out.println(fact1);

        for(int i=1;i<=(n-r);i++)

        {

            fact2=fact2*i;

        }

        //System.out.println(fact2);

        int res=fact/(fact1*fact2);

        return res;

    }

}

```

## 26.Sum of Common Elements

Write a program to find out sum of common elements in given two arrays. If no common elements are found print - “No common elements”.

Include a class **UserMainCode** with a static method **getSumOfIntersection** which accepts

two integer arrays and their sizes. The return type (integer) should return the sum of common elements.

Create a Class Main which would be used to accept 2 Input arrays and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of  $2+m+n$  integers. The first integer corresponds to  $m$  (Size of the 1st array), the second integer corresponds to  $n$  (Size of the 2nd array), followed by  $m+n$  integers corresponding to the array elements.

Output consists of a single Integer corresponds to the sum of common elements or a string

“No common elements”.

Refer sample output for formatting specifications.

Assume the common element appears only once in each array.

**Sample Input 1:**

4

3

2

3

5

1

1

3

9

**Sample Output 1:**

4

**Sample Input 2:**

4

3

2

3

5

1

12

31

9

## **Sample Output 2:**

No common elements

-

### **Main**

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        int n=sc.nextInt();

        int m=sc.nextInt();

        int[] a=new int[n];

        int[] b=new int[m];

        for(int i=0;i<n;i++){
            a[i]=sc.nextInt();
        }

        for(int i=0;i<m;i++){
            b[i]=sc.nextInt();
        }

        int u=UserMainCode.getSumOfIntersection (a,b,n,m);

        if(u== -1)

            System.out.println("No common elements");

        else

            System.out.println(u);

        sc.close();
    }
}
```

### **UserMainCode**

```
public class UserMainCode {
```

```

public static int getSumOfIntersection(int a[],int b[],int n,int m)

{
    int sum=0;

    for(int i=0;i<a.length;i++)

    {
        for(int j=0;j<b.length;j++)

        {if(a[i]==b[j])

        sum=sum+a[i];

        }}

        if(sum==0)

        return -1;

        else

        return sum;

    }

}

```

## 27.Validating Input Password

102. Write a code get a password as string input and validate using the rules specified below.

Apply following validations:

1. Minimum length should be 8 characters
2. Must contain any one of these three special characters @ or \_ or #
3. May contain numbers or alphabets.
4. Should not start with special character or number
5. Should not end with special character

Include a class **UserMainCode** with a static method **validatePassword** which accepts password string as input and returns an integer. The method returns 1 if the password is

valid. Else it returns -1.

Create a class **Main** which would get the input and call the static method **validatePassword** present in the UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output is a string Valid or Invalid.

Refer sample output for formatting specifications.

### **Sample Input 1:**

ashok\_23

### **Sample Output 1:**

Valid

### **Sample Input 2:**

1980\_200

### **Sample Output 2:**

Invalid

## **Main**

```
import java.util.*;

public class Main{

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

String a=sc.next();

int e=UserMainCode.validatePassword(a);

if(e==1){

System.out.println("Valid");

}
```

```

else
{
    System.out.println("Invalid");

}
sc.close();
}}

```

## UserMainCode

```

public class UserMainCode {

    public static int validatePassword(String a){

        int d=0;

        if(a.length()>=8){

            if(a.contains("#") || a.contains("@") || a.contains("_"))

            {

                char c= a.charAt(0);

                //System.out.println(c);

                if(Character.isAlphabetic(c))

                {

                    char dd=a.charAt(a.length()-1);

                    //System.out.println(dd);

                    if((Character.isAlphabetic(dd))||(Character.isDigit(dd)))

                    {

                        if(a.matches(".*[0-9]{1,}.*")||a.matches(".*[a-zA-Z]{1,}.*")){

                            d=1;

                        }

                    }

                }

            }

        }

    }

}

```

```
    }  
}  
  
    else  
        d=-1;  
  
    return d;  
  
}  
}
```

## 28.iD Validation

Write a program to get two string inputs and validate the ID as per the specified format.

Include a class **UserMainCode** with a static method **validateIDLocations** which accepts two

strings as input.

The return type of the output is a string Valid Id or Invalid Id.

Create a class **Main** which would get the input and call the static  
method **validateIDLocations** present in the UserMainCode.

### **Input and Output Format:**

Input consists of two strings.

First string is ID and second string is location. ID is in the format CTS-LLL-XXXX where  
LLL is

the first three letters of given location and XXXX is a four digit number.

Output is a string Valid id or Invalid id.

Refer sample output for formatting specifications.

### **Sample Input 1:**

CTS-hyd-1234

hyderabad

### **Sample Output 1:**

Valid id

### **Sample Input 2:**

CTS-hyd-123

hyderabad

### **Sample Output 2:**

Invalid id

## **Main**

```
import java.util.*;

public class Main3 {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        String s1=sc.next();

        String s2=sc.next();

        boolean b=UserMainCode3.validateIDLocations(s1,s2);

        if(b==true)

            System.out.println("Valid id");

        else

            System.out.println("Invalid id");

        sc.close();
    }
}
```

```
}
```

```
}
```

---

### UserMainCode

```
import java.util.StringTokenizer;

public class UserMainCode3 {

    public static boolean validateIDLocations(String s1, String s2) {

        String s3=s2.substring(0, 3);

        boolean b=false;

        StringTokenizer t=new StringTokenizer(s1,"-");

        String s4=t.nextToken();

        String s5=t.nextToken();

        String s6=t.nextToken();

        if(s4.equals("CTS") && s5.equals(s3) && s6.matches("[0-9]{4}"))

            b=true;

        else{

            b=false;
        }

        return b;
    }
}
```

### **29.Remove Elements**

Write a program to remove all the elements of the given length and return the size of the final array as output. If there is no element of the given length, return the size of the same

array as output.

Include a class **UserMainCode** with a static method **removeElements** which accepts a string

array, the number of elements in the array and an integer. The return type (integer) should return the size of the final array as output.

Create a Class Main which would be used to accept Input String array and a number and call the static method present in UserMainCode.

Assume maximum length of array is 20.

### **Input and Output Format:**

Input consists of a integers that corresponds to n, followed by n strings and finally m which corresponds to the length value.

Output consists of a single Integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

5

a

bb

b

ccc

ddd

2

### **Sample Output 1:**

4

## **Main**

```
import java.util.*;
```

```

public class Main
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        int n=Integer.parseInt(sc.nextLine());

        String[] a=new String[n];

        for(int i=0;i<n;i++)
        {
            a[i]=sc.nextLine();

            int m=Integer.parseInt(sc.nextLine());

            System.out.println(UserMainCode.removeElements(a,m));
        }

        sc.close();
    }
}

```

### UserMainCode

```

public class UserMainCode {

    public static int removeElements(String[] a,int m){

        int u=a.length;

        for(int i=0;i<a.length;i++)
        {
            if(a[i].length()==m)

                u--;
        }

        return u;
    }
}

```

### **30.Find the difference between Dates in months**

Given a method with two date strings in yyyy-mm-dd format as input. Write code to find the difference between two dates in months.

Include a class **UserMainCode** with a static method **getMonthDifference** which accepts two date strings as input.

The return type of the output is an integer which returns the difference between two dates in months.

Create a class **Main** which would get the input and call the static method **getMonthDifference** present in the UserMainCode.

#### **Input and Output Format:**

Input consists of two date strings.

Format of date : yyyy-mm-dd.

Output is an integer.

Refer sample output for formatting specifications.

#### **Sample Input 1:**

2012-03-01

2012-04-16

#### **Sample Output 1:**

1

#### **Sample Input 2:**

2011-03-01

2012-04-16

#### **Sample Output 2:**

13

## Main

```
import java.text.*;
import java.util.*;

public class Main {

    public static void main(String[] args) throws ParseException {
        Scanner sc=new Scanner(System.in);
        String s1=sc.next();
        String s2=sc.next();
        System.out.println(UserMainCode.getMonthDifference(s1,s2));
        sc.close();
    }
}
```

## UserMainCode

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

public class UserMainCode {

    public static int getMonthDifference(String s1, String s2) throws ParseException {
        SimpleDateFormat sdf=new SimpleDateFormat("yyyy-MM-dd");
        Date d1=sdf.parse(s1);
        Date d2=sdf.parse(s2);
        Calendar cal=Calendar.getInstance();
    }
}
```

```

    cal.setTime(d1);

    int months1=cal.get(Calendar.MONTH);

    int year1=cal.get(Calendar.YEAR);

    cal.setTime(d2);

    int months2=cal.get(Calendar.MONTH);

    int year2=cal.get(Calendar.YEAR);

    int n=((year2-year1)*12)+(months2-months1);

    return n;

}

}

```

### **31.Sum of cubes and squares of elements in an array**

Write a program to get an int array as input and identify even and odd numbers. If number is odd get cube of it, if number is even get square of it. Finally add all cubes and squares together and return it as output.

Include a class **UserMainCode** with a static method **addEvenOdd** which accepts integer array as input.

The return type of the output is an integer which is the sum of cubes and squares of elements in the array.

Create a class **Main** which would get the input and call the static method **addEvenOdd** present in the UserMainCode.

#### **Input and Output Format:**

Input consists of integer array.

Output is an integer sum.

Refer sample output for formatting specifications.

**Sample Input 1:**

5  
2  
6  
3  
4  
5

**Sample Output 1:**

208

**Main**

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        int a[]={};
        for(int i=0;i<n;i++){
            a[i]=sc.nextInt();
        }
        System.out.println(UserMainCode.addEvenOdd(a));
        sc.close();
    }
}
```

```
}
```

### **UserMainCode**

```
public class UserMainCode6 {  
  
    public static int addEvenOdd(int[] a) {  
  
        int n1=0,n2=0;  
  
        for(int i=0;i<a.length;i++)  
  
            if(a[i]%2==0)  
  
                n1+=(a[i]*a[i]);  
  
            else  
  
                n2+=(a[i]*a[i]*a[i]);  
  
        return n1+n2;  
  
    }  
  
}
```

### **32.IP Validator**

Write a program to read a string and validate the IP address. Print “Valid” if the IP address is valid, else print “Invalid”.

Include a class **UserMainCode** with a static method **ipValidator** which accepts a string. The

return type (integer) should return 1 if it is a valid IP address else return 2.

Create a Class Main which would be used to accept Input String and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of a string that corresponds to an IP.

Output consists of a string (“Valid” or “Invalid”).

Refer sample output for formatting specifications.

**Note:** An IP address has the format a.b.c.d where a,b,c,d are numbers between 0-255.

### **Sample Input 1:**

132.145.184.210

### **Sample Output 1:**

Valid

### **Sample Input 2:**

132.145.184.290

### **Sample Output 2:**

Invalid

## **Main**

```
import java.util.*;

public class Main7 {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        String ipAddress=sc.next();

        boolean b=UserMainCode7.validateIpAddress(ipAddress);

        if(b==true)

            System.out.println("Valid");

        else

            System.out.println("Invalid");

        sc.close();

    }
}
```

```
}
```

### **UserMainCode**

```
import java.util.StringTokenizer;

public class UserMainCode7 {

    public static boolean validateIpAddress(String ipAddress) {

        boolean b1=false;

        StringTokenizer t=new StringTokenizer(ipAddress,".");

        int a=Integer.parseInt(t.nextToken());

        int b=Integer.parseInt(t.nextToken());

        int c=Integer.parseInt(t.nextToken());

        int d=Integer.parseInt(t.nextToken());

        if((a>=0 && a<=255)&&(b>=0 && b<=255)&&(c>=0 && c<=255)&&(d>=0
        && d<=255))

            b1=true;

        return b1;
    }
}
```

### **33.Difference between two dates in days**

Get two date strings as input and write code to find difference between two dates in days.

Include a class **UserMainCode** with a static method **getDateDifference** which accepts two

date strings as input.

The return type of the output is an integer which returns the difference between two dates in days.

Create a class **Main** which would get the input and call the static method **getDateDifference** present in the UserMainCode.

### **Input and Output Format:**

Input consists of two date strings.

Format of date : yyyy-mm-dd.

Output is an integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

2012-03-12

2012-03-14

### **Sample Output 1:**

2

### **Sample Input 2:**

2012-04-25

2012-04-28

### **Sample Output 2:**

3

## **Main**

```
import java.text.ParseException;
import java.util.*;
public class Main {
    public static void main(String[] args) throws ParseException
```

```
{  
  
Scanner s=new Scanner(System.in);  
  
String s1=s.nextLine();  
  
String s2=s.nextLine();  
  
int output=UserMainCode.getDateDifference(s1,s2);  
  
System.out.println(output);  
  
s.close();  
  
}  
  
}
```

### UserMainCode

```
import java.text.ParseException;  
  
import java.text.SimpleDateFormat;  
  
import java.util.*;  
  
public class UserMainCode {  
  
public static int getDateDifference(String s1,String s2) throws ParseException  
{  
  
SimpleDateFormat sd=new SimpleDateFormat("yyyy-MM-dd");  
  
Date d=sd.parse(s1);  
  
Calendar c=Calendar.getInstance();  
  
c.setTime(d);  
  
long d1=c.getTimeInMillis();  
  
d=sd.parse(s2);  
  
c.setTime(d);  
  
long d2=c.getTimeInMillis();  
  
int n=Math.abs((int) ((d1-d2)/(1000*3600*24)));  
  
return n;  
}
```

```
}
```

### 34.File Extension

Write a program to read a file name as a string and find out the file extension and return it as output. For example, the file sun.gif has the extension gif.

Include a class **UserMainCode** with a static method **fileIdentifier** which accepts a string. The return type (string) should return the extension of the input string (filename).

Create a Class Main which would be used to accept Input String and call the static method present in UserMainCode.

#### **Input and Output Format:**

Input consists of a string that corresponds to a file name.

Output consists of a string(extension of the input string (filename)).

Refer sample output for formatting specifications.

#### **Sample Input 1:**

sun.gif

#### **Sample Output 1:**

Gif

#### **Main**

```
import java.util.*;  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        Scanner s=new Scanner(System.in);  
  
        System.out.println("enter the string");  
  
        String s1=s.nextLine();  
  
        String output=UserMainCode.fileIdentifier(s1);  
  
        System.out.println(output);  
    }  
}
```

```

s.close();

}

}

UserMainCode

import java.util.*;

public class UserMainCode {

public static String fileIdentifier(String s1)

{

StringTokenizer t=new StringTokenizer(s1,".");

t.nextToken();

String s2=t.nextToken();

return s2;

}

}

```

### **35.Find common characters and unique characters in string**

Given a method with two strings as input. Write code to count the common and unique letters in the two strings.

Note:

- Space should not be counted as a letter.
- Consider letters to be case sensitive. ie, "a" is not equal to "A".

Include a class **UserMainCode** with a static method **commonChars** which accepts two strings as input.

The return type of the output is the count of all common and unique characters in the two strings.

Create a class **Main** which would get the inputs and call the static method **commonChars** present in the UserMainCode.

## **Input and Output Format:**

Input consists of two strings.

Output is an integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

a black cow

battle ship

### **Sample Output 1:**

2

**[Explanation :** b, l and a are the common letters between the 2 input strings. But 'a' appears

more than once in the 1st string. So 'a' should not be considered while computing the count value.]

### **Sample Input 2:**

australia

sri lanka

### **Sample Output 2:**

4

## **Main**

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);

        String s1=sc.nextLine();

        String s2=sc.nextLine();

        StringBuffer sb1=new StringBuffer(s1.replace(" ",""));
    }
}
```

```
StringBuffer sb2=new StringBuffer(s2.replace(" ",""));

int output=UserMainCode.commonChars(s1,s2,sb1,sb2);

System.out.println(output);

sc.close();

}

}
```

### UserMainCode

```
import java.util.*;

public class UserMainCode {

public static int commonChars(String s1,String s2,StringBuffer sb1,StringBuffer
sb2) {

for(int i=0;i<sb1.length();i++){

int c=0;

for(int j=i+1;j<sb1.length();j++){

if(sb1.charAt(i)==sb1.charAt(j)){

sb1.deleteCharAt(j);

c++;

}

}

if(c>=1){

sb1.deleteCharAt(i);

}

}

for(int i=0;i<sb2.length();i++){

int c=0;

for(int j=i+1;j<sb2.length();j++){

if(sb2.charAt(i)==sb2.charAt(j)){

sb2.deleteCharAt(j);

}

}

}

}
```

```

c++;
}

}

if(c>=1){

sb2.deleteCharAt(i);

}

}

int count=0;

for(int i=0;i<sb1.length();i++){

for(int j=0;j<sb2.length();j++){

if(sb1.charAt(i)==sb2.charAt(j)){

count++;

}

}

}

return (count);

}
}

```

### **36.Initial Format**

Write a program to input a person's name in the format "FirstName LastName" and return the person name in the following format - "LastName, InitialOfFirstName".

Include a class **UserMainCode** with a static method **nameFormatter** which accepts a string.

The return type (string) should return the expected format.

Create a Class Main which would be used to accept Input String and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string that corresponds to a Person's name.

Output consists of a string(person's name in expected format).

Refer sample output for formatting specifications.

**Sample Input :**

Jessica Miller

**Sample Output:**

Miller, J

**Main**

```
import java.util.*;  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        Scanner s=new Scanner(System.in);  
  
        String s1=s.nextLine();  
  
        String output=UserMainCode.nameFormatter(s1);  
  
        System.out.println(output);  
  
        s.close();  
    }  
}
```

**UserMainCode**

```
import java.util.*;  
  
public class UserMainCode {  
  
    public static String nameFormatter(String s1) {  
  
        StringBuffer sb=new StringBuffer();  
  
        StringTokenizer st=new StringTokenizer(s1, " ");  
  
        String s2=st.nextToken();  
  
        String s3=st.nextToken();
```

```
sb.append(s3).append(",");
sb.append(s2.substring(0,1).toUpperCase());
return sb.toString();
}
}
```

### 37.Character cleaning

Write a program to input a String and a character, and remove that character from the given String. Print the final string.

Include a class **UserMainCode** with a static method **removeCharacter** which accepts a string

and a character. The return type (string) should return the character cleaned string.

Create a Class Main which would be used to accept Input String and call the static method present in UserMainCode.

#### **Input and Output Format:**

Input consists of a string and a character.

Output consists of a string(the character cleaned string).

Refer sample output for formatting specifications.

#### **Sample Input :**

elephant

e

#### **Sample Output:**

Lphant

#### **Main**

```
import java.util.*;
public class Main {
```

```
public static void main(String[] args) {  
  
    Scanner s=new Scanner(System.in);  
  
    String s1=s.nextLine();  
  
    String c=s.nextLine();  
  
    String output=UserMainCode.removeCharacter(s1,c);  
  
    System.out.println(output);  
  
}  
  
}
```

### **UserMainCode**

```
import java.util.*;  
  
public class UserMainCode {  
  
    public static String removeCharacter(String s1,String c)  
  
    {  
  
        String d=s1.replace(c,"");  
  
        return d;  
  
    }  
  
}
```

### **38.Vowel Check**

Write a program to read a String and check if that String contains all the vowels. Print "yes" if the string contains all vowels else print "no".

Include a class **UserMainCode** with a static method **getVowels** which accepts a string. The return type (integer) should return 1 if the String contains all vowels else return -1.

Create a Class Main which would be used to accept Input String and call the static method present in UserMainCode.

#### **Input and Output Format:**

Input consists of a string.

Output consists of a string("yes" or "no").

Refer sample output for formatting specifications.

**Sample Input 1:**

abceiduosp

**Sample Output 1:**

yes

**Sample Input 2:**

bceiduosp

**Sample Output 2:**

No

**Main**

```
import java.util.*;  
  
public class User {  
  
    public static void main(String[] args) {  
  
        Scanner s=new Scanner(System.in);  
  
        String s1=s.nextLine();  
  
        String s2=s1.toLowerCase();  
  
        int output=UserMainCode.getVowels(s2);  
  
        //System.out.println(output);  
  
        if(output==1)  
        {  
            System.out.println("yes");  
        }  
        else  
            System.out.println("no");  
    }  
}
```

```

s.close();

}

}

UserMainCode

import java.util.*;

public class UserMainCode {

public static int getVowels(String s2) {

if(s2.contains("a") && s2.contains("e") && s2.contains("i") && s2.contains("o") &&
s2.contains("u") )

{

return 1;

}

else

return -1;

}

}

```

### **39.Swap Characters**

Write a program to input a String and swap the every 2 characters in the string. If size is an odd number then keep the last letter as it is. Print the final swapped string.

Include a class **UserMainCode** with a static method **swapCharacter** which accepts a string.

The return type (String) should return the character swapped string.

Create a Class Main which would be used to accept Input String and call the static method present in UserMainCode.

#### **Input and Output Format:**

Input consists of a string.

Output consists of a string.

Refer sample output for formatting specifications.

**Sample Input 1:**

TRAINER

**Sample Output 1:**

RTIAENR

**Sample Input 2:**

TOM ANDJERRY

**Sample output 2:**

OT MNAJDREYR

## Main

```
import java.util.*;

public class Main {

    public static void main(String[] args) {

        Scanner s=new Scanner(System.in);

        String s1=s.nextLine();

        String output=UserMainCode.swapCharacter(s1);

        System.out.println(output);

        s.close();
    }
}
```

## UserMainCode

```
import java.util.*;

public class UserMainCode {

    public static String swapCharacter(String s1)

    {
```

```

StringBuffer sb=new StringBuffer();

int l=s1.length();

if(l%2==0)

{

for(int i=0;i<s1.length()-1;i=i+2)

{

char a=s1.charAt(i);

char b=s1.charAt(i+1);

sb.append(b).append(a);

}

return sb.toString();

}

else

{

for(int i = 0;i<s1.length()-1;i=i+2)

{

char a=s1.charAt(i);

char b=s1.charAt(i+1);

sb.append(b).append(a);

}

sb.append(s1.charAt(l-1));

return sb.toString();

}
}
}
}

```

#### **40.Average of Elements in Hashmap**

Given a method with a `HashMap<int, float>` as input. Write code to find out avg of all

values whose keys are even numbers. Round the average to two decimal places and return as output.

[Hint : If the average is 5.901, the rounded average value is 5.9 . If the average is 6.333, the rounded average value is 6.33 . ]

Include a class **UserMainCode** with a static method **avgOfEven** which accepts a `HashMap<int, float>` as input.

The return type of the output is a floating point value which is the average of all values whose key elements are even numbers.

Create a class **Main** which would get the input and call the static method **avgOfEven** present in the UserMainCode.

### **Input and Output Format:**

Input consists of the number of elements in the `HashMap` and the `HashMap<int, float>`.

Output is a floating point value that corresponds to the average.

Refer sample output for formatting specifications.

### **Sample Input 1:**

3

1

2.3

2

4.1

6

6.2

### **Sample Output 1:**

5.15

**Sample Input 2:**

3

9

3.1

4

6.3

1

2.6

**Sample Output 2:**

6.3

**Main**

```
import java.util.HashMap;
import java.util.Scanner;
public class Main {
    public static void main(String []args){
        Scanner sc=new Scanner(System.in);
        int s=sc.nextInt();
        HashMap<Integer,Float>hm=new HashMap<Integer,Float>();
        for(int i=0;i<s;i++)
        {
            int r=sc.nextInt();
            float j=sc.nextFloat();
            hm.put(r,j);
        }
        System.out.println(UserMainCode.display(hm));
        sc.close();
    }
}
```

```
}

}

UserMainCode

import java.text.DecimalFormat;

import java.util.*;

public class UserMainCode

{

public static String display(HashMap<Integer,Float>hm)

{

float sum=0;

int count=0;

DecimalFormat df=new DecimalFormat("#.00");

Iterator<Integer> it=hm.keySet().iterator();

while(it.hasNext())

{

int y=it.next();

if(y%2==0)

{

sum=(float) (sum+hm.get(y));

count++;

}}
```

**float** d=sum/count;

**return** df.format(d);

}

}

## 41.Calculate Average – Hash Map

Write a method that accepts the input data as a hash map and finds out the avg of all values whose keys are odd numbers.

Include a class **UserMainCode** with a static method **calculateAverage** which accepts aHashMap<Integer,Double> and the size of the HashMap. The return type (Double) should return the calculated average. Round the average to two decimal places and return it.

Create a Class Main which would be used to accept Input values and store it as a hash map, and call the static method present in UserMainCode.

### Input and Output Format:

Input consists of an integer n corresponds to number of hash map values, followed by 2n values. (index followed by value).

Output consists of a Double.

Refer sample input and output for formatting specifications.

### Sample Input :

```
4
1
3.41
2
4.1
3
1.61
4
2.5
```

### Sample Output :

```
2.51
```

## Main

```
import java.util.*;
import java.text.*;
public class Main {
public static void main(String[] args)
{
HashMap<Integer,Double> hm=new HashMap<Integer,Double>();
Scanner sc=new Scanner(System.in);
int n=sc.nextInt();
for(int i=0;i<n;i++)
{
int a=sc.nextInt();
double s=sc.nextDouble();
hm.put(a,s);
}
System.out.println(UserMaincode.dis(hm));}}
```

## UserMainCode

```
class UserMaincode
{
public static double dis(HashMap<Integer,Double> h1)
```

```

{
double avg=0.0,sum=0.0;
int k=0;
for(Map.Entry m:h1.entrySet())
{
int a=(Integer)m.getKey();
if(a%2!=0)
{
Double d=(Double) m.getValue();
sum=sum+d;
k++;
}
avg = (double)sum/k;
DecimalFormat df = new DecimalFormat(".##");
String b1 = df.format(avg);
double b = Double.parseDouble(b1);
return b;
}

```

## 42.Count Sequential Characters

109.Get a string as input and write code to count the number of characters which gets repeated 3 times consecutively and return that count (ignore case). If no character gets repeated 3 times consecutively return -1.

Include a class **UserMainCode** with a static method **countSequentialChars** which accepts a string as input.

The return type of the output is the repeat count.

Create a class **Main** which would get the input and call the static method **countSequentialChars** present in the UserMainCode.

### **Input and Output Format:**

Input consists a string.

Output is an integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

abcXXXabc

### **Sample Output 1:**

1

### **Sample Input 2:**

aaxxyzAAAx

### **Sample Output 2:**

2

## **Main**

```

import java.util.*;
import java.text.*;

public class Main {
public static void main(String[] args) {
Scanner sc=new Scanner (System.in);
String input1=sc.next();

System.out.println(UserMainCode.consecutiveRepeatingChar(input1));
}

```

```
}
```

### UserMainCode

```
class UserMainCode
{
    public static int consecutiveRepeatingChar(String input1)
    {
        int c=0;
        int n=0;
        for(int i=0;i<input1.length()-1;i++){
            if(input1.charAt(i)==input1.charAt(i+1))
                n++;
            else
                n=0;
            if(n==2)
                c++;
        }
        return c;
    }
}
```

### 43.Length of the Largest Chunk

Write a program to read a string and find the length of the largest chunk in the string. If there are no chunks print “No chunks” else print the length.

NOTE: chunk is the letter which is repeating 2 or more than 2 times.

Include a class **UserMainCode** with a static method **largestChunk** which accepts a string.

The return type (Integer) should return the length of the largest chunk if the chunk is present, else return -1.

Create a Class Main which would be used to accept Input String and call the static method present in UserMainCode.

#### Input and Output Format:

Input consists of a string.

Refer sample output for formatting specifications.

#### Sample Input 1:

You are toooo good

#### Sample Output 1:

4

(Because the largest chunk is letter 'o' which is repeating 4 times)

#### Sample Input 2:

who are u

#### Sample Output 2:

No chunks

#### Main

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        String s1=sc.nextLine();
        System.out.println(UserMainCode.largestChunk(s1));
```

```

}

}

UserMaincode
class UserMainCode
{
public static int largestChunk(String s1) {
int max=1;
int b=0;
StringTokenizer t=new StringTokenizer(s1, " ");
while(t.hasMoreTokens()){
String s2=t.nextToken();
int n=0;
for(int i=0;i<s2.length()-1;i++)
if(s2.charAt(i)==s2.charAt(i+1))
n++;
if(n>max)
{
max=n;
b=max+1;
}
}
return b;
}
}

```

#### **44.Unique Characters in a string**

Write a program that takes a string and returns the number of unique characters in the string. If the given string does not contain any unique characters return -1

Include a class **UserMainCode** with a static method **uniqueCounter** which accepts a string as input.

The return type of the output is the count of all unique characters in the strings.

Create a class **Main** which would get the input and call the static method **uniqueCounter** present in the UserMainCode.

#### **Input and Output Format:**

Input consists a string.

Output is an integer.

Refer sample output for formatting specifications.

#### **Sample Input 1:**

HelloWorld

#### **Sample Output 1:**

5

#### **Sample Input 2:**

coco

#### **Sample Output 2:**

-1

#### **Main**

```

import java.util.*;
import java.text.*;

```

```

public class Main {

```

```

public static void main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);
String s1 = sc.nextLine();
System.out.println(UserMaincode.uniqueCounter(s1));
}}

```

#### UserMainCode

```

class UserMaincode
{
    public static int uniqueCounter(String s1)
    {

StringBuffer sb = new StringBuffer(s1);
for (int i = 0; i < sb.length(); i++) {
    int count = 0;
    for (int j = i + 1; j < sb.length(); j++) {
        if (sb.charAt(i) == sb.charAt(j)) {
            sb.deleteCharAt(j);
            j--;
            count++;
        }
    }
    if (count >= 1) {
        sb.deleteCharAt(i);
        i--;
    }
}
return sb.length();
}
}

```

## 45.Name Shrinking

Write a program that accepts a string as input and converts the first two names into dotseparated initials and printa the output.

Input string format is 'fn mn ln'. Output string format is 'ln [mn's 1st character].[fn's 1st character]'

Include a class **UserMainCode** with a static method **getFormattedString** which accepts a string. The return type (String) should return the shrinked name.

Create a Class Main which would be used to accept Input String and call the static method present in UserMainCode.

#### **Input and Output Format:**

Input consists of a string.

Output consists of a String.

Refer sample output for formatting specifications.

#### **Sample Input:**

Sachin Ramesh Tendulkar

#### **Sample Output:**

Tendulkar R.S

```

Main
import java.text.*;
import java.util.*;

public class Main {
public static void main(String[] args) {
Scanner sc=new Scanner(System.in);
String s1=sc.nextLine();
System.out.println(UserMainCode.getFormatedString(s1));
}
}

UserMainCode
class UserMainCode
{

public static String getFormatedString(String s1) {
StringBuffer sb=new StringBuffer();
StringTokenizer st=new StringTokenizer(s1, " ");
String s2=st.nextToken();
String s3=st.nextToken();
String s4=st.nextToken();
sb.append(s4).append(" ");
sb.append(s3.substring(0,1));
sb.append(".");
sb.append(s2.substring(0,1));
return sb.toString();
}
}

```

## 46.Odd Digit Sum

Write a program to input a String array. The input may contain digits and alphabets ("de5g4G7R"). Extract odd digits from each string and find the sum and print the output. For example, if the string is "AKj375A" then take 3+7+5=15 and not as 375 as digit. Include a class **UserMainCode** with a static method **oddDigitSum** which accepts a string array and the size of the array. The return type (Integer) should return the sum. Create a Class Main which would be used to accept Input Strings and call the static method present in UserMainCode.

Assume maximum length of array is 20.

### **Input and Output Format:**

Input consists of an integer n, corresponds to the number of strings, followed by n Strings.  
Output consists of an Integer.

Refer sample output for formatting specifications.

### **Sample Input :**

```

3
cog2nizant1
al33k
d2t4H3r5

```

### **Sample Output :**

```

15
(1+3+3+3+5)

```

```

Main
import java.util.Scanner;
public class Main {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
int s1=sc.nextInt();
String[] s2 = new String[s1];
for (int i = 0; i < s1; i++) {
s2[i] = sc.next();
}
System.out.println(UserMainCode.oddDigitSum(s2));
}}

```

### UserMainCode

```

public class UserMainCode {
public static int oddDigitSum (String[] s1) {
int sum=0;
for(int i=0;i<s1.length;i++)
for(int j=0;j<s1[i].length();j++){
char c=s1[i].charAt(j);
if(Character.isDigit(c)){
if(c%2!=0)
{
String t=String.valueOf(c);
int n=Integer.parseInt(t);
sum=sum+n; } }
return sum;
}
}

```

## 47.Unique Number

Write a program that accepts an Integer as input and finds whether the number is Unique or not. Print Unique if the number is “Unique”, else print “Not Unique”.

**Note:** A Unique number is a positive integer (without leading zeros) with no duplicate digits. For example 7, 135, 214 are all unique numbers whereas 33, 3121, 300 are not.

Include a class **UserMainCode** with a static method **getUnique** which accepts an integer. The return type (Integer) should return 1 if the number is unique else return -1.

Create a Class Main which would be used to accept Input Integer and call the static method present in UserMainCode.

### Input and Output Format:

Input consists of an integer .

Output consists of a String (“Unique” or “Not Unique”).

Refer sample output for formatting specifications.

### Sample Input 1:

123

### Sample Output 1:

Unique

### Sample Input 2:

33

### Sample Output 2:

Not Unique

```

import java.util.*;
import java.text.*;

public class Main{
public static void main(String[] args)
{int j=0;
Scanner sc=new Scanner(System.in);
int n=sc.nextInt();
j=UserMainCode.getUnique(n);
if(j>0)
{
System.out.println("Not Unique");
}
else if(j==0)
{
System.out.println("Unique");
}
}}

```

```

class UserMainCode
{
public static int getUnique(int n)
{
int []a=new int[100];
int i=0,count=0;
while(n!=0)
{
int num=n%10;
a[i]=num;
i++;
n=n/10;
}
for(int j=0;j<=i-1;j++)
{
for(int k=j+1;k<=i-1;k++)
{
if(a[j]==a[k]){
count++;
}
}
}
return count;
}
}

```

## 48.Sum of Lowest marks

Given input as HashMap, value consists of marks and rollno as key.Find the sum of the lowest three subject marks from the HashMap.

Include a class **UserMainCode** with a static method **getLowest** which accepts a Hashmap

with marks and rollno.

The return type of the output is the sum of lowest three subject marks.

Create a class **Main** which would get the input and call the static method **getLowest** present

in the UserMainCode.

### **Input and Output Format:**

First line of the input corresponds to the HashMap size.

Input consists a HashMap with marks and rollno.

Output is an integer which is the sum of lowest three subject marks.

Refer sample output for formatting specifications.

### **Sample Input 1:**

```
5  
1  
54  
2  
85  
3  
74  
4  
59  
5  
57
```

### **Sample Output 1:**

```
170
```

### **Sample Input 2:**

```
4  
10  
56  
20  
58  
30  
87  
40  
54
```

### **Sample Output 2:**

```
168
```

## **Main**

```
import java.util.*;  
  
public class Main {  
    public static void main(String args[]){  
        Scanner sc = new Scanner(System.in);  
        int n=Integer.parseInt(sc.nextLine());  
        HashMap<Integer, Integer> h1=new HashMap<Integer, Integer>();  
        for(int i=0;i<n;i++)  
        {  
            h1.put(sc.nextInt(), sc.nextInt());  
        }  
        System.out.println(UserMainCode.getLowest(h1));  
    }  
}
```

```

}
}

UserMainCode
class UserMainCode {
public static int getLowest(HashMap<Integer, Integer> h1)
{
ArrayList<Integer> a1=new ArrayList<Integer>();
int m=0;
Iterator<Integer> it=h1.keySet().iterator();
while(it.hasNext())
{
int x=it.next();
a1.add(h1.get(x));
}
Collections.sort(a1);
m=a1.get(0)+a1.get(1)+a1.get(2);
return m;
}}

```

## 49. Color Code Validation same as 21

Give a String as colour code as input and write code to validate whether the given string is a valid color code or not.

Validation Rule:

String should start with the Character '#'.

Length of String is 7.

It should contain 6 Characters after '#' Symbol.

It should contain Characters between 'A-F' and Digits '0-9'.

If String acceptable the return true otherwise false.

Include a class **UserMainCode** with a static method **validateColourCode** which accepts a string as input.

The return type of the output is a boolean which returns true if its is a valid color code else it returns false.

Create a class **Main** which would get the input and call the static method **validateColourCode** present in the UserMainCode.

**Input and Output Format:**

Input consists a string corresponding to the color code.

Output is a boolean which returns true or false

Refer sample output for formatting specifications.

**Sample Input 1:**

#99FF33

**Sample Output 1:**

true

**Sample Input 2:**

#CCCC99#

**Sample Output 2:**

False

**Main**

```

import java.util.*;
public class Add {
public static void main(String[] args) {
Scanner s=new Scanner(System.in);
String s1=s.next();
boolean b=userMainCode.validateColourCode (s1);
if(b==true)

```

```

System.out.println("valid color code");
else
System.out.println("invalid color code");
}

UserMainCode

static class userMainCode{
public static boolean validateColourCode (String s1)
boolean b=false;
    if(s1.length()==7&&s1.matches("#[A-F0-9]{1,}")){
        b=true;
    }
    return b;
}}}

```

## 50.Repeating set of characters in a string

Get a string and a positive integer n as input .The last n characters should repeat the number of times given as second input.Write code to repeat the set of character from the given string.

Include a class **UserMainCode** with a static method **getString** which accepts a string and an

integer n as input.

The return type of the output is a string with repeated n characters.

Create a class **Main** which would get the input and call the static method **getString** present

in the UserMainCode.

### **Input and Output Format:**

Input consists a string and a positive integer n.

Output is a string with repeated characters.

Refer sample output for formatting specifications.

### **Sample Input 1:**

Cognizant

3

### **Sample Output 1:**

Cognizantantant

### **Sample Input 2:**

myacademy

2

### **Sample Output 2:**

Myacademymymy

### **Main**

```

import java.util.*;
public class Main {

    public static void main(String[] args) {
        Scanner s= new Scanner(System.in);
        String input= s.next();
    }
}

```

```

        int n=s.nextInt();
        System.out.println(userMainCode.getString(input,n));
    }

}

UserMainCode
class userMainCode {

    public static String getString(String input, int n){
        StringBuffer sb=new StringBuffer();
        sb.append(input);
        for (int i=0;i<n;i++){
            sb.append(input.substring(input.length()-n,input.length()));
        }
        return sb.toString();
    }
}

```

### 51.Finding the day of birth

Given an input as date of birth of person, write a program to calculate on which day (MONDAY,TUESDAY....) he was born store and print the day in Upper Case letters.

Include a class **UserMainCode** with a static method **calculateBornDay** which accepts a string as input.

The return type of the output is a string which should be the day in which the person was born.

Create a class **Main** which would get the input and call the static method **calculateBornDay** present in the UserMainCode.

#### **Input and Output Format:**

NOTE: date format should be(dd-MM-yyyy)

Input consists a date string.

Output is a string which the day in which the person was born.

Refer sample output for formatting specifications.

#### **Sample Input 1:**

29-07-2013

#### **Sample Output 1:**

MONDAY

#### **Sample Input 2:**

14-12-1992

#### **Sample Output 2:**

MONDAY

#### **Main**

```

import java.util.*;
import java.text.*;
public class Main {

    public static void main(String[] args) throws ParseException {
        Scanner s= new Scanner(System.in);
        String input= s.next();
        System.out.println(userMainCode.calculateBornDay(input));
    }
}

```

```

    }
}

UserMainCode
class userMainCode{
    public static String calculateBornDay(String input) throws ParseException{
        SimpleDateFormat sdf= new SimpleDateFormat("dd-MM-yyyy");
        SimpleDateFormat sdf1= new SimpleDateFormat("EEEE");
        Date d= sdf.parse(input);
        String s1= sdf1.format(d);
        return s1;
    }
}

```

## 52.Removing elements from HashMap

Given a HashMap as input, write a program to perform the following operation : If the keys are divisible by 3 then remove that key and its values and print the number of remaining keys in the hashmap.

Include a class **UserMainCode** with a static method **afterDelete** which accepts a HashMap  
as input.

The return type of the output is an integer which represents the count of remaining elements in the hashmap.

Create a class **Main** which would get the input and call the static method **afterDelete** present in the UserMainCode.

### **Input and Output Format:**

First input corresponds to the size of hashmap

Input consists a HashMap

Output is an integer which is the count of remaining elements in the hashmap.

Refer sample output for formatting specifications.

### **Sample Input 1:**

```

4
339
RON
1010
JONS
3366
SMITH
2020
TIM

```

### **Sample Output 1:**

```

2

```

### **Sample Input 2:**

```

5
1010
C2WE
6252
XY4E
1212

```

M2ED  
7070  
S2M41ITH  
8585  
J410N

**Sample Output 2:**

3

**Main**

```
import java.util.*;
import java.text.*;
public class Main {

    public static void main(String[] args) {
        Scanner s= new Scanner(System.in);
        HashMap<Integer, String>hm=new HashMap<Integer, String>();
        int n= s.nextInt();
        for(int i=0;i<n;i++){
            hm.put(s.nextInt(),s.next());
        }

        System.out.println(UserMainCode.afterDelete(hm));
        s.close();
    }
}
```

**UserMainCode**

```
class UserMainCode{
    public static int afterDelete(HashMap<Integer, String> hm){
        int count=0;
        Iterator<Integer>itr=hm.keySet().iterator();
        while(itr.hasNext())
        {
            int n=itr.next();
            if(n%3!=0)
            {
                count++;
            }
        }
        return count;
    }
}
```

**53.Experience Calculator**

Write a program to read Date of Joining and current date as Strings and Experience as integer and validate whether the given experience and calculated experience are the same. Print “true” if same, else “false”.

Include a class **UserMainCode** with a static method **calculateExperience** which accepts 2

strings and an integer. The return type is boolean.

Create a Class Main which would be used to accept 2 string (dates) and an integer and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of 2 strings and an integer, where the 2 strings corresponds to the date of joining and current date, and the integer is the experience.

Output is either “true” or “false”.

Refer sample output for formatting specifications.

### **Sample Input 1:**

11/01/2010

01/09/2014

4

### **Sample Output 1:**

true

### **Sample Input 2:**

11/06/2009

01/09/2014

4

### **Sample Output 2:**

False

### **Main**

```
import java.util.*;
import java.text.*;
public class Main {

    public static void main(String[] args) throws ParseException {
        Scanner sc=new Scanner(System.in);
        String a=sc.next();
        String b=sc.next();
        int c=sc.nextInt();
        long res=(userMainCode.calculateExperience(a,b,c));
        if(res==c)
        {
            System.out.println("true");
        }
        else
            System.out.println("false");
    }
}
```

### **UserMainCode**

```
class userMainCode{
    public static long calculateExperience(String a, String b, int c) throws
ParseException{
    SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
    Date d=new Date();
    Date d1=new Date();
    d=sdf.parse(a);
    d1=sdf.parse(b);
    long t=d.getTime();
    long t1=d1.getTime();
    long t3=t1-t;
    long l1=(24 * 60 * 60 * 1000);
```

```

long l=l1*365;
long res=t3/l;
return res;
}
}

```

## 54.Flush Characters

Write a program to read a string from the user and remove all the alphabets and spaces from the String, and **only store special characters and digit** in the output String. Print

the output string.

Include a class **UserMainCode** with a static method **getSpecialChar** which accepts a string.

The return type (String) should return the character removed string.

Create a Class Main which would be used to accept a string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a strings.

Output consists of an String (character removed string).

Refer sample output for formatting specifications.

### **Sample Input :**

cogniz\$#45Ant

### **Sample Output :**

\$#45

#### **Main:**

```

import java.util.*;
import java.text.*;
public class Main {

    public static void main(String[] args) {
        Scanner s= new Scanner(System.in);
        String input=s.next();
        System.out.println(UserMainCode.getSpecialChar(input));

    }
}

```

#### **UserMainCode:**

```

class UserMainCode{
    public static String getSpecialChar(String input){
        int i;
        StringBuffer sb= new StringBuffer();
        for(i=0;i<input.length();i++)
        {
            char a=input.charAt(i);
            if (!Character.isAlphabetic(a))
                sb.append(a);
        }
        return sb.toString();
    }
}

```

```
}
```

## 55.String Repetition

Write a program to read a string and an integer and return a string based on the below rules.

If input2 is equal or greater than 3 then repeat the first three character of the String by given input2 times, separated by a space.

If input2 is 2 then repeat the first two character of String two times separated by a space,  
If input2 is 1 then return the first character of the String.

Include a class UserMainCode with a static method **repeatString** which takes a string & integer and returns a string based on the above rules.

Create a Class Main which would be used to accept Input string and call the static method present in UserMainCode.

### Input and Output Format:

Input consists of a string and integer.

Output consists of a string.

Refer sample output for formatting specifications.

### Sample Input 1:

```
COGNIZANT
```

```
4
```

### Sample Output 1:

```
COG COG COG COG
```

### Sample Input 2:

```
COGNIZANT
```

```
2
```

### Sample Output 2:

```
CO CO
```

```
class Main{
```

```
    public static void main(String[] args) {  
        Scanner s= new Scanner(System.in);  
        System.out.println("enter a string");  
        String input= s.next();  
        int n= s.nextInt();  
        System.out.println(UserMainCode.repeatString(input, n));  
    }
```

```
}
```

```
class UserMainCode{
```

```
    public static String repeatString(String input, int n){  
        StringBuffer sb= new StringBuffer();  
        String s1= new String();  
        if (n==1){  
            s1=input.substring(0,1);  
            sb.append(s1).append(" ");  
        }  
        if(n==2){  
            s1=input.substring(0,2);  
            for(int i=0;i<n;i++)  
                sb.append(s1).append(" ");  
        }  
        if(n>=3){
```

```

        s1=input.substring(0,3);
        for(int i=0;i<n;i++)
            sb.append(s1).append(" ");
    }
    return sb.toString();
}
}

```

## 56.Average of Prime Locations

Write a program to read an integer array and find the average of the numbers located on the Prime location(indexes).

Round the avarage to two decimal places.

Assume that the array starts with index 0.

Include a class UserMainCode with a static method **averageElements** which accepts a single

integer array. The return type (double) should be the average.

Create a Class Main which would be used to accept Input array and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of  $n+1$  integers. The first integer corresponds to  $n$ , the number of elements in the array. The next ' $n$ ' integers correspond to the elements in the array.

Output consists of a single Double value.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20.

### **Sample Input 1:**

```

8
4
1
7
6
5
8
6
9

```

### **Sample Output 1:**

7.5

#### **Main:**

```

import java.util.*;
import java.text.*;
public class Main {

    public static void main(String[] args) {
        Scanner s= new Scanner(System.in);
        int n,i;
        System.out.println("enter the array size");
        n=s.nextInt();
        int array[]=new int[n];
        for(i=0;i<n;i++){
            array[i]=s.nextInt();
        }
    }
}

```

```

        }
        System.out.println(UserMainCode.AverageElements(array));
        s.close();
    }

}

UserMainCode:
class UserMainCode{
    public static double AverageElements(int array[]){
        int n, sum=0, count=0, count1=0;
        double average;
        n=array.length;
        for(int i=0; i<=n; i++){
            for(int j=1; j<n; j++){
                if(i%j==0)
                    count++;
                if(count==2){
                    sum= sum+array[i];
                    count1++;
                }
            }
        }
        average= sum/count1;
        DecimalFormat df=new DecimalFormat("#.00");
        double ddd=Double.parseDouble(df.format(average));

        return ddd;
    }
}

```

## 57.Common Elements

Write a program to read two integer arrays and find the sum of common elements in both the arrays. If there are no common elements return -1 as output. Include a class UserMainCode with a static method sumCommonElements which accepts two single integer array. The return type (integer) should be the sum of common elements.

Create a Class Main which would be used to accept Input array and call the static method present in UserMainCode. Assume that all the elements will be distinct. Input and Output Format: Input consists of  $2n+1$  integers. The first integer corresponds to  $n$ , the number of elements in the array. The next ' $n$ ' integers correspond to the elements in the array, The last  $n$  elements correspond to the elements of the second array. Output consists of a single Integer value. Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20.

### Sample Input 1:

```

4
1
2
3
4
2

```

3  
6  
7

**Sample Output 1:**

5

**Main:**

```
import java.util.*;
public class Main {
private static Scanner s ;
;
public static void main(String[] args) {
s = new Scanner (System.in);
int n = s.nextInt();
int a[] = new int[n];
int b[] = new int[n];
for(int i=0;i<n;i++)
{
a[i] = s.nextInt();
}
for(int i=0;i<n;i++)
{
b[i] = s.nextInt();
}
System.out.println(UserMainCode.sumCommonElements(a, b));
}
}
```

**UserMainCode:**

```
public class UserMainCode {
public static int sumCommonElements(int a[],int b[]){
int sum = 0 ;
for(int i=0;i<a.length;i++)
{
for(int j=0;j<b.length;j++){
if(a[i]==b[j])
sum = sum + a[i];}
}
if(sum==0)
return -1;
else return sum;
}
}
```

**58. Middle of Array**

Write a program to read an integer array and return the middle element in the array. The size of the array would always be odd. Include a class UserMainCode with a static method getMiddleElement which accepts a single integer array. The return type (integer) should be the middle element in the array. Create a Class Main which would be used to accept Input

array and call the static method present in UserMainCode. Input and Output Format: Input consists of  $n+1$  integers. The first integer corresponds to  $n$ , the number of elements in the array. The next ' $n$ ' integers correspond to the elements in the array. Output consists of a single Integer value. Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 19.

Sample Input 1:

```
5  
1  
5  
23  
64  
9  
Sample Output 1:  
23
```

**Main:**

```
import java.util.*;  
  
public class Main {  
  
    private static Scanner s;  
  
    public static void main(String[] args) {  
  
        s = new Scanner(System.in);  
  
        int n = s.nextInt();  
  
        int[] a = new int[n];  
  
        for(int i=0;i<n;i++){  
  
            a[i] = s.nextInt();  
        }  
  
        System.out.println(UserMainCode.getMiddleElement(a));  
    }  
}
```

**UserMainCode:**

```
public class UserMainCode {  
  
    public static int getMiddleElement(int a[]){  
  
        int n = a.length;  
  
        return a[n/2];  
    }  
}
```

```
}
```

## 59. Simple String Manipulation

Write a program to read a string and return a modified string based on the following rules.

Return the String without the first 2 chars except when

1. keep the first char if it is 'j' 2. keep the second char if it is 'b'.

Include a class UserMainCode with a static method getString which accepts a string. The return type (string) should be the modified string based on the above rules. Consider all letters in the input to be small case. Create a Class Main which would be used to accept Input string and call the static method present in UserMainCode. Input and Output Format: Input consists of a string with maximum size of 100 characters. Output consists of a string. Refer sample output for formatting specifications.

Sample Input 1:

hello

Sample Output 1:

llo

Sample Input 2:

java

Sample Output 2:

Jva

### Main:

```
import java.util.*;  
  
public class Main {  
  
    private static Scanner s;  
  
    public static void main(String[] args) {  
  
        s = new Scanner(System.in);  
  
        String s1 = s.next();  
  
        System.out.println(UserMainCode.getString(s1));  
  
    }  
  
}
```

### UserMainCode:

```
public class UserMainCode {  
  
    public static String getString(String s1){  
  
        StringBuffer sb=new StringBuffer();  
  
        char a=s1.charAt(0);  
  
        char b=s1.charAt(1);  
  
        if(a=='j')  
            sb.append(a);  
        else if(b=='b')  
            sb.append(b);  
        else  
            sb.append(s1.substring(2));  
  
        return sb.toString();  
    }  
}
```

```

        if(a!='j'&& b!='b')
            sb.append(s1.substring(2));

        else if(a=='j' && b!='b')
            sb.append("j").append(s1.substring(2));

        else if(a!='j' && b=='b')
            sb.append(s1.substring(1));

        else
            sb.append(s1.substring(0));

        return sb.toString();
    }

}

```

## 60. Date Validation

Write a program to read a string representing a date. The date can be in any of the three formats 1:dd-MM-yyyy 2: dd/MM/yyyy 3: dd.MM.yyyy If the date is valid, print valid else print invalid. Include a class UserMainCode with a static method getValidDate which accepts a string. The return type (integer) should be based on the validity of the date. Create a Class Main which would be used to accept Input string and call the static method present in UserMainCode. Input and Output Format: Input consists of a string. Output consists of a string. Refer sample output for formatting specifications.

Sample Input 1:

03.12.2013

Sample Output 1:

valid

Sample Input 2:

03\$12\$2013

Sample Output 3:

Invalid

### Main:

```

import java.util.*;

public class Main {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        String s= sc.next();

        int b = UserMainCode.getvalues(s);

        if(b==1)
            System.out.println("Valid");
    }
}

```

```

    else
        System.out.println("Invalid");
    }
}

```

### UserMainCode:

```

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class UserMainCode {

    public static int getvalues(String s) {
        if(s.matches("[0-9]{2}[.]{1}[0-9]{2}[.]{1}[0-9]{4}"))
        {
            SimpleDateFormat sdf=new SimpleDateFormat("dd.MM.yyyy");
            sdf.setLenient(false);
            try
            {
                Date d1=sdf.parse(s);
                return 1;
            } catch (ParseException e) {
                return -1;
            }
        }
        else if(s.matches("[0-9]{2}[/]{1}[0-9]{2}[/][0-9]{4}"))
        {
            SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
            sdf.setLenient(false);
            try
            {
                Date d1=sdf.parse(s);

```

```

    return 1;

} catch (ParseException e) {

return -1;

}

}

else if(s.matches("[0-9]{2}[-]{1}[0-9]{2}[-][0-9]{4})) {

SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");

sdf.setLenient(false);

try

{

Date d1=sdf.parse(s);

return 1;

} catch (ParseException e) {

return -1;

}

}

else

return -1;

}

}

```

## 61. Boundary Average

Given an int array as input, write a program to compute the average of the maximum and minimum element in the array. Include a class UserMainCode with a static method “getBoundaryAverage” that accepts an integer array as argument and returns a float that corresponds to the average of the maximum and minimum element in the array. Create a class Main which would get the input array and call the static method getBoundaryAverage present in the UserMainCode. Input and Output Format: The first line of the input consists of an integer n, that corresponds to the size of the array. The next n lines consist of integers that correspond to the elements in the array. Assume that the maximum number of elements in the array is 10. Output consists of a single float value that corresponds to the average of the max and min element in the array.

Sample Input :

```
3  
6  
9  
4  
2  
5
```

Sample Output:

```
5.5
```

**Main:**

```
import java.util.*;  
  
import java.util.Arrays;  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        Scanner sc = new Scanner(System.in);  
  
        int s = sc.nextInt();  
  
        int a[] = new int[s];  
  
        for (int i = 0; i < s; i++)  
  
            a[i] = sc.nextInt();  
  
        System.out.println(UserMainCode.getBoundaryAverage(a));  
  
    }  
  
}
```

**UserMainCode**

```
import java.util.Arrays;  
  
public class UserMainCode {  
  
    public static float getBoundaryAverage(int a[] ){  
  
        Arrays.sort(a);  
  
        int sum = a[0] + a[a.length - 1];  
  
        float avg = (float) sum / 2;  
  
        return avg;  
    }  
  
}
```

## 62. Count Vowels

Given a string input, write a program to find the total number of vowels in the given string. Include a class UserMainCode with a static method “countVowels” that accepts a String argument and returns an int that corresponds to the total number of vowels in the given string. Create a class Main which would get the String as input and call the static method countVowels present in the UserMainCode. Input and Output Format: Input consists of a string. Output consists of an integer..

Sample Input:

avinash

Sample Output:

3

### **Main:**

```
import java.util.*;
public class Main {
private static Scanner s;

public static void main(String[] args) {
s = new Scanner(System.in);
String s1= s.next();
System.out.println(countVowels(s1));
}
```

### **UserMainCode**

```
public class UserMainCode{
public static int countVowels(String s1)
{
String s2=s1.toLowerCase();
String s3="aeiou";
int count=0;
for(int i=0;i<s2.length();i++)
{
    for(int j=0;j<s3.length();j++)
    {
        if(s2.charAt(i)==s3.charAt(j))
        {
            count++;
        }
    }
}
return count;
}
```

## 63. Month Name

Given a date as a string input in the format dd-mm-yy, write a program to extract the month and to print the month name in upper case. Include a class UserMainCode with a static method “getMonthName” that accepts a String argument and returns a String that corresponds to the month name. Create a class Main which would get the String as input and

call the static method getMonthName present in the UserMainCode. The month names are {JANUARY, FEBRUARY, MARCH, APRIL, MAY, JUNE, JULY, AUGUST, SEPTEMBER, OCTOBER, NOVEMBER, DECEMBER} Input and Output Format: Input consists of a String. Output consists of a String.

Sample Input:

01-06-82

Sample Output:

JUNE

**Main:**

```
import java.text.ParseException;
import java.util.Scanner;
public class Main {
    public static void main(String[] args) throws ParseException {

        Scanner sc=new Scanner(System.in);
        String s1=sc.nextLine();
        System.out.println(UserMainCode.calculateBornDay(s1));
        sc.close();
    }
}
```

**UserMainCode:**

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
public class UserMainCode {
    public static String calculateBornDay(String s1) throws ParseException
    {
        SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yy");
        SimpleDateFormat sdf1=new SimpleDateFormat("MMMM");
        Date d=sdf.parse(s1);
        String s=sdf1.format(d);
        return s.toUpperCase();
    }
}
```

#### 64. Reverse SubString

Given a string, startIndex and length, write a program to extract the substring from right to left. Assume the last character has index 0. Include a class UserMainCode with a static method “reverseSubstring” that accepts 3 arguments and returns a string. The 1st argument corresponds to the string, the second argument corresponds to the startIndex and the third argument corresponds to the length. Create a class Main which would get a String and 2 integers as input and call the static method reverseSubstring present in the UserMainCode. Input and Output Format: The first line of the input consists of a string. The second line of the input consists of an integer that corresponds to the startIndex. The third line of the input consists of an integer that corresponds to the length of the substring.

Sample Input:

rajasthan

2

3

Sample Output:

hts

**Main:**

```
import java.util.*;
public class Main {
private static Scanner s;
public static void main(String[] args) {
s=new Scanner(System.in);
String input1= s.next();
int input2=s.nextInt();int input3=s.nextInt();
System.out.println(UserMainCode.retrieveString(input1,input2,input3));
}
}
```

**UserMainCode**

```
class UserMainCode {
public static String retrieveString(String input1, int input2, int input3) {
StringBuffer sb=new StringBuffer(input1);
sb.reverse();
String output=sb.substring(input2, input2+input3);
return output;
}
}
```

## 65. String Finder

Given three strings say Searchstring, Str1 and Str2 as input, write a program to find out if Str2 comes after Str1 in the Searchstring.

Include a class **UserMainCode** with a static method “**stringFinder**” that accepts 3 String arguments and returns an integer. The 3 arguments correspond to SearchString, Str1 and Str2. The function returns 1 if Str2 appears after Str1 in the Searchstring. Else it returns 2.

Create a class **Main** which would get 3 Strings as input and call the static method **stringFinder** present in the UserMainCode.

**Input and Output Format:**

Input consists of 3 strings.

The first input corresponds to the SearchString.

The second input corresponds to Str1.

The third input corresponds to Str2.

Output consists of a string that is either “yes” or “no”

**Sample Input 1:**

geniousRajKumarDev

Raj

Dev

**Sample Output 1:**

Yes

Sample Input 2:  
geniousRajKumarDev  
Dev  
Raj

Sample Output 2:  
No

**USERMAINCODE:**

```
public class UserMainCode {  
    public static int stringFinder(String s1, String s2, String s3)  
    {  
        String a1=s1.toLowerCase();  
        String a2=s2.toLowerCase();  
        String a3=s3.toLowerCase();  
        if(a1.contains(a2)&&a1.contains(a3))  
        {  
            if(a1.indexOf(a2)<a1.indexOf(a3))  
            {  
                return 1;  
            }  
            else  
                return 2;  
        }  
        return 0;  
    }  
}
```

**MAIN:**

```
import java.util.*;  
public class Main {  
    public static void main(String[] args) {  
        Scanner s=new Scanner(System.in);  
        String s1=s.next();  
        String s2=s.next();  
        String s3=s.next();  
        int b=UserMainCode.stringFinder(s1, s2, s3);  
        if(b==1)  
        {  
            System.out.println("yes");  
        }  
        else  
            System.out.println("No");  
        s.close();  
    }  
}
```

**66. Phone Number Validator**

Given a phone number as a string input, write a program to verify whether the phone number is valid using the following business rules:

- It should contain only numbers or dashes (-)
- Dashes may appear at any position
- Should have exactly 10 digits

Include a class **UserMainCode** with a static method “**validatePhoneNumber**” that accepts a String input and returns an integer. The method returns 1 if the phone number is valid. Else it returns 2.

Create a class **Main** which would get a String as input and call the static method **validatePhoneNumber** present in the UserMainCode.

#### **Input and Output Format:**

Input consists of a string.

Output consists of a string that is either 'Valid' or 'Invalid'

#### **Sample Input 1:**

265-265-7777

#### **Sample Output 1:**

Valid

#### **Sample Input 2:**

265-65-7777

#### **Sample Output 2:**

Invalid

#### **USERMAINCODE:**

```
public class UserMainCode {  
    public static int validatePhoneNumber(String s1)  
  
    {  
        String s2 = s1.replaceAll("-", "");  
        if (s2.matches("[0-9]{10}"))  
        {  
            return 1;  
        }  
        else  
            return 2;  
    }  
}
```

### MAIN:

```
import java.util.*;  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        Scanner s=new Scanner(System.in);  
  
        String s1=s.nextLine();  
  
        int b=UserMainCode.validatePhoneNumber(s1);  
  
        if(b==1)  
        {  
            System.out.println("Valid");  
        }  
  
        else  
        {  
            System.out.println("Invalid");  
            s.close();  
        }  
  
    }  
}
```

## 67. Month : Number of Days

Given two inputs year and month (Month is coded as: Jan=0, Feb=1 ,Mar=2 ...), write a program to find out total number of days in the given month for the given year. Include a class **UserMainCode** with a static method “**getNumberOfDays**” that accepts 2 integers as arguments and returns an integer. The first argument corresponds to the year and the second argument corresponds to the month code. The method returns an integer corresponding to the number of days in the month.  
Create a class **Main** which would get 2 integers as input and call the static method **getNumberOfDays** present in the **UserMainCode**.

### Input and Output Format:

Input consists of 2 integers that correspond to the year and month code. Output consists of an integer that corresponds to the number of days in the month in the given year.

Sample Input:

2000

1

Sample Output:

29

USERMAINCODE:

```
import java.util.Calendar;
public class UserMainCode {
    public static int getNumberOfDays(int y,int c)
    {
        Calendar cal=Calendar.getInstance();
        cal.set(Calendar.YEAR, y);
        cal.set(Calendar.MONTH, c);
        int day=cal.getActualMaximum(Calendar.DAY_OF_MONTH);
        return day;
    }
}
```

MAIN:

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        int y=s.nextInt();
        int c=s.nextInt();
        System.out.println(UserMainCode.getNumberOfDays(y, c));
        s.close();
    }
}
```

## 68. Negative String

Given a string input, write a program to replace every appearance of the word "is" by "is not".

If the word "is" is immediately preceded or followed by a letter no change should be made to the string .

Include a class **UserMainCode** with a static method "**negativeString**" that accepts a String arguement and returns a String.

Create a class **Main** which would get a String as input and call the static method **negativeString** present in the UserMainCode.

Input and Output Format:

Input consists of a String.

Output consists of a String.

Sample Input 1:

This is just a misconception

Sample Output 1:

This is not just a misconception

**Sample Input 2:**

Today is misty

**Sample Output 2:**

Today is not misty

**USERMAINCODE:**

```
public class UserMainCode {
    public static String negativeString(String s1)
    {
        String str=s1.replace(" is ", " is not ");
        return str;
    }
}
```

**MAIN:**

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.nextLine();
        System.out.println(UserMainCode.negativeString(s1));
        s.close();
    }
}
```

## 69. Validate Number

Given a negative number as string input, write a program to validate the number and to print the corresponding positive number.

Include a class **UserMainCode** with a static method “**validateNumber**” that accepts a string argument and returns a string. If the argument string contains a valid negative number, the method returns the corresponding positive number as a string. Else the method returns -1. Create a class **Main** which would get a String as input and call the static method **validateNumber** present in the UserMainCode.

**Input and Output Format:**

Input consists of a String.

Output consists of a String.

**Sample Input 1:**

-94923

**Sample Output 1:**

94923

**Sample Input 2:**

-6t

**Sample Output 2:**

-1

**USERMAINCODE:**

```
public class UserMainCode {
    public static String validateNumber(String s1)
```

```

{
    String ss="-1";
    if (s1.matches("[ -]{1}[0-9]{1,}"))
    {
        String st=s1.replace(" -","");
        return st;
    }
    else
        return ss;
    }
}

```

### MAIN:

```

import java.util.*;
public class Main {

    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.next();
        System.out.println(UserMainCode.validateNumber(s1));
        s.close();
    }
}

```

## 70. Digits

Write a program to read a non-negative integer n that returns the count of the occurrences of 7 as digit.

Include a class UserMainCode with a static method **countSeven** which accepts the integer value. The return type is integer which is the count value.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

#### Input and Output Format:

Input consists of a integer.

Output consists of integer.

Refer sample output for formatting specifications.

#### Sample Input 1:

717

#### Sample Output 1:

2

#### Sample Input 2:

4534

#### Sample Output 2:

0

#### USERMAINCODE:

```

public class UserMainCode {
    public static int countSeven(int n)
    {
        int rem,sum=0;

```

```

while(n>0)
{
rem=n%10;
if(rem==7)
{
    sum++;
}
n=n/10;
}
return sum;
}
}

```

### MAIN:

```

import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        System.out.println(UserMainCode.countSeven(n));
        s.close();
    }
}

```

## 71. String Processing – III

Write a program to read a string where all the lowercase 'x' chars have been moved to the end of the string.

Include a class UserMainCode with a static method **moveX** which accepts the string. The return type is the modified string.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### Input and Output Format:

Input consists of a string.

Output consists of a string.

Refer sample output for formatting specifications.

#### Sample Input 1:

xxhixx

#### Sample Input 2:

XXxtest

#### Sample Output 1:

hixxxx

#### Sample Output 2:

XXtestxx

### USERMAINCODE:

```

public class UserMainCode {
    public static String moveX(String s1)
{
}

```

```

String s2="";
String s3="";
for(int i=0;i<s1.length();i++)
{
    char c=s1.charAt(i);
    if(c=='x')
    {
        s2=s2+s1.charAt(i);
    }
    else
    {
        s3=s3+s1.charAt(i)
    }
}
String st=s3.concat(s2);
return st;
}
}

```

#### MAIN:

```

import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.next();
        System.out.println(UserMainCode.moveX(s1));
        s.close();
    }
}

```

## 72. String Processing – IV

Write a program to read a string and also a number N. Form a new string starting with 1st character and with every Nth character of the given string. Ex - if N is 3, use chars 1, 3, 6,... and so on to form the new String. Assume  $N \geq 1$ .

Include a class UserMainCode with a static method `getStringUsingNthCharacter` which accepts the string and the number n. The return type is the string as per the problem statement.

Create a Class Main which would be used to accept the string and integer and call the static method present in UserMainCode.

#### Input and Output Format:

Input consists of a string and integer.

Output consists of a string.

Refer sample output for formatting specifications.

#### Sample Input 1:

#### Sample Output 1:

HelloWorld

HelWrd

**USERMAINCODE:**

```
public class UserMainCode {  
    public static String getStringUsingNthCharacter(String s1,int n)  
    {  
        StringBuffer sb=new StringBuffer();  
        sb.append(s1.charAt(0));  
        for(int i=n-1;i<s1.length();i+=n)  
        {  
            sb.append(s1.charAt(i));  
        }  
        return sb.toString();  
    }  
}
```

**MAIN:**

```
import java.util.*;  
public class Main {  
    public static void main(String[] args) {  
        Scanner s=new Scanner(System.in);  
        String s1=s.next();  
        int n=s.nextInt();  
        System.out.println(UserMainCode.getStringUsingNthCharacter(s1, n));  
        s.close();  
    }  
}
```

### 73. Digit Comparison

Write a program to read two integers and return true if they have the same last digit.  
Include a class UserMainCode with a static method **compareLastDigit** which accepts two integers and returns boolean. (true / false)  
Create a Class Main which would be used to accept two integers and call the static method present in UserMainCode.

#### Input and Output Format:

Input consists of two integer.

Output consists TRUE / FALSE.

Refer sample output for formatting specifications.

#### Sample Input 1:

59

29

### Sample Output 1:

TRUE

#### UserMainCode

```
public class UserMainCode {  
    public static boolean compareLastDigit(int c,int d)  
    {  
        int c1=c%10;  
        int d1=d%10;  
        boolean b=false;  
        if(c1==d1)  
        {  
            b=true;  
        }  
        return b;  
    }  
}
```

#### Main

```
import java.util.*;  
public class Main {  
  
    public static void main(String[] args) {  
        Scanner s=new Scanner(System.in);  
        int c=s.nextInt();  
        int d=s.nextInt();  
        boolean res=UserMainCode.compareLastDigit(c,d);  
        if(res==true)  
        {  
            System.out.println("TRUE");  
        }  
  
        else  
        {  
            System.out.println("FALSE");  
        }  
  
        s.close();  
    }  
}
```

### 74. Duplicates

Given three integers (a,b,c) find the sum. However, if one of the values is the same as another, both the numbers do not count towards the sum and the third number is returned as the sum.

Include a class UserMainCode with a static method **getDistinctSum** which accepts three integers and returns integer.

Create a Class Main which would be used to accept three integers and call the static method

present in UserMainCode.

### **Input and Output Format:**

Input consists of three integers.

Output consists of a integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

```
1  
2  
1
```

### **Sample Output 1:**

```
2
```

### **Sample Input 2:**

```
1  
2  
3
```

### **Sample Output 2:**

```
6
```

### **UserMainCode:**

```
public class UserMainCode {  
    public static int getDistinctSum(int a,int b,int c)  
    {  
        int sum;  
        if(a==b)  
        {  
            sum=c;  
        }  
  
        else if(b==c)  
        {  
            sum=a;  
        }  
        else if(c==a)  
        {  
            sum=b;  
        }  
        else  
        {  
            sum=a+b+c;  
        }  
        return sum;  
    }  
}
```

### **Main:**

```
import java.util.*;  
public class Main {  
  
    public static void main(String[] args) {  
        Scanner s=new Scanner(System.in);  
        int a=s.nextInt();
```

```

        int b=s.nextInt();
        int c=s.nextInt();
        int sum=UserMainCode.getDistinctSum(a, b, c);
        System.out.println(sum);
        s.close();
    }

}

```

## 75. String Processing - MixMania

Write a program to read a string and check if it starts with '\_ix' where '\_' is any one char(a-z, A-Z, 0-9).

If specified pattern is found return true else false.

Include a class UserMainCode with a static method **checkPattern** which accepts the string.

The return type is TRUE / FALSE.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of TRUE / FALSE.

Refer sample output for formatting specifications.

### **Sample Input 1:**

Mix Mania

### **Sample Output 1:**

TRUE

### **UserMainCode**

```

public class UserMainCode {
    public static boolean checkPattern(String str)
    {
        String str1=str.substring(0,3);
        int a=0,b=0,c=0;
        char c1=str1.charAt(0);
        char c2=str1.charAt(1);
        char c3=str1.charAt(2);
        boolean b1=false;
        if(Character.isDigit(c1)||Character.isLetter(c1))
        {
            a=1;
        }
        if(c2=='i')
        {
            b=1;
        }
        if(c3=='x')
        {
            c=1;
        }
        if(a==1&&b==1&&c==1)
        {
            b1=true;
        }
    }
}

```

```

        }
        return b1;
    }

}

Main:
import java.util.*;
public class Main {

    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String str=s.nextLine();
        boolean b2=UserMainCode.checkPattern(str);
        if(b2==true)
        {
            System.out.println("TRUE");
        }
        else
        {
            System.out.println("FALSE");
        }
    s.close();
    }
}

```

## 76. String Processing

Write a program to read a string and return a new string where the first and last chars have been interchanged.

Include a class UserMainCode with a static method **exchangeCharacters** which accepts the string. The return type is the modified string.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of string.

Refer sample output for formatting specifications.

### **Sample Input 1:**

HelloWorld

### **Sample Output 1:**

delloWorlH

### **UserMainCode**

```

public class UserMainCode {
public static String exchangeCharacters(String s1)
{
    String s2=s1.substring(1,s1.length()-1);
    StringBuffer sb=new StringBuffer();
    char c1=s1.charAt(0);
    char c2=s1.charAt(s1.length()-1);
    sb.append(c2).append(s2).append(c1);
    return sb.toString();
}

```

```
}
```

**Main:**

```
import java.util.*;
public class Main {

    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.next();

        System.out.println(UserMainCode.exchangeCharacters(s1));
        s.close();
    }
}
```

## 77. Regular Expression - II

Given a string (s) apply the following rules.

1. String consists of three characters only.
2. The characters should be alphabets only.

If all the conditions are satisfied then print TRUE else print FALSE.

Include a class UserMainCode with a static method **validateString** which accepts the string.

The return type is the boolean formed based on rules.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of a string.

Output consists of TRUE or FALSE .

Refer sample output for formatting specifications.

**Sample Input 1:**

AcB

**Sample Output 1:**

TRUE

**Sample Input 2:**

A2B

**Sample Output 2:**

FALSE

**UserMainCode:**

```
public class UserMainCode {
    public static boolean validateString(String s1)
    {
        boolean b=false;
        if(s1.length()==3)
        {
            if(s1.matches("[a-zA-Z]{3}"))
            {
                b=true;
            }
        }
    }
}
```

```

        }
    }
    return b;
}
}

Main:

import java.util.*;
public class Main {

    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.next();
        boolean b1=userMainCode.validateString(s1);
        if(b1==true)
        {
            System.out.println("TRUE");
        }
        else
        {
            System.out.println("FALSE");
        }
    }

    s.close();
}
}

```

## 78. Strings Processing - Replication

Write a program to read a string and also a number N. Return the replica of original string for n given time.

Include a class UserMainCode with a static method **repeatString** which accepts the the string and the number n. The return type is the string based on the problem statement.

Create a Class Main which would be used to accept the string and integer and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string and integer.

Output consists of a string.

Refer sample output for formatting specifications.

### **Sample Input 1:**

Lily

2

### **Sample Output 1:**

LilyLily

### **UserMainCode:**

```

public class UserMainCode {
    public static String repeatString(String s1,int n)
    {
        StringBuffer sb=new StringBuffer();
        for(int i=0;i<n;i++)

```

```

        {
            sb.append(s1);
        }
    return sb.toString();
}

}

```

### **Main:**

```

import java.util.*;
public class Main {

    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.next();
        int n=s.nextInt();
        System.out.println(UserMainCode.repeatString(s1, n));
    s.close();
    }

}

```

### **79. SumOdd**

Write a program to read an integer and find the sum of all odd numbers from 1 to the given number. [inclusive of the given number]

if N = 9 [ 1,3,5,7,9]. Sum = 25

Include a class UserMainCode with a static method **addOddNumbers** which accepts the number n. The return type is the integer based on the problem statement.

Create a Class Main which would be used to accept the integer and call the static method present in UserMainCode.

#### **Input and Output Format:**

Input consists of a integer.

Output consists of a integer.

Refer sample output for formatting specifications.

#### **Sample Input 1:**

6

#### **Sample Output 1:**

9

#### **UserMainCode:**

```

public class UserMainCode {
    public static int addOddNumbers(int n)
    {
        int sum=0;
        for(int i=1;i<=n;i+=2)
        {
            sum=sum+i;
        }
    return sum;
}

```

```
}
```

### **Main:**

```
import java.util.*;
public class Main {

    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        System.out.println(UserMainCode.addOddNumbers(n));
        s.close();
    }
}
```

## **80. String Processing - V**

Write a program to read a string array, concatenate the array elements one by one separated by comma and return the final string as output.

Include a class UserMainCode with a static method **concatString** which accepts the string array. The return type is the string.

Create a Class Main which would be used to accept the string array and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of an integer n which is the number of elements followed by n string values.

Output consists of the string.

Refer sample output for formatting specifications.

### **Sample Input 1:**

```
3
```

```
AAA
```

```
BBB
```

```
CCC
```

### **Sample Output 1:**

```
AAA,BBB,CCC
```

### **UserMainCode:**

```
public class UserMainCode {
    public static String concatString(int n, String[] s1)
    {
        StringBuffer sb=new StringBuffer();
        for(int i=0;i<s1.length;i++)
        {
            sb.append(s1[i]).append(",");
        }
        String s2=sb.toString();
        String s3=s2.substring(0,s2.length()-1);

        return s3;
    }
}
```

**Main:**

```
import java.util.*;
public class Main {

    public static void main(String[] args) {
Scanner s=new Scanner(System.in);
int n=s.nextInt();
String s1[]={};
for(int i=0;i<n;i++)
{
    s1[i]=s.next();
}
System.out.println(UserMainCode.concatString(n, s1));
s.close();
    }

}
```

**81.Unique Number**

Given three integers (a,b,c) , Write a program that returns the number of unique integers among the three.

Include a class UserMainCode with a static method **calculateUnique** which accepts three integers and returns the count as integer.

Create a Class Main which would be used to accept three integers and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of three integers.

Output consists of a integer.

Refer sample output for formatting specifications.

**Sample Input 1:**

```
12
4
3
```

**Sample Output 1:**

```
3
```

**Sample Input 2:**

```
4
-4
4
```

**Sample Output 2:**

```
2
```

**Main:**

```
import java.util.*;
public class Main {
public static void main(String[]args){
Scanner s=new Scanner(System.in);
int a=s.nextInt();
int b=s.nextInt();
int c=s.nextInt();
System.out.println(UserMainCode.calculateUnique(a, b, c));
```

```

        s.close();
    }
}

```

### UserMainCode:

```

public class UserMainCode {
    public static int calculateUnique(int a,int b,int c)
    {
        int d=0;
        if(a!=b&&a!=c&&b!=c)
        {
            d=3;
        }
        else if(a==b&&a==c&&b==c)
        {
            d=1;
        }
        else if((a!=b&&a==c&&b==c) || (a!=b&&a!=c&&b==c))
        {
            d=2;
        }
        else if((a==b&&a!=c&&b==c) || (a==b&&a!=c&&b!=c))
        {
            d=2;
        }
        else if((a==b&&a==c&&b!=c) || (a!=b&&a==c&&b!=c))
        {
            d=2;
        }

        return d;
    }
}

```

## 82. Math Calculator

Write a program that accepts three inputs, first two inputs are operands in int form and third one being one of the following five operators: +, -, \*, /, %. Implement calculator logic and return the result of the given inputs as per the operator provided. In case of division, Assume the result would be integer.

Include a class UserMainCode with a static method **calculator** which accepts two integers, one operand and returns the integer.

Create a Class Main which would be used to accept three integers and call the static method present in UserMainCode.

### Input and Output Format:

Input consists of two integers and a character.

Output consists of a integer.

Refer sample output for formatting specifications.

### Sample Input 1:

23

2

\*

## Sample Output 1:

46

### Main:

```
import java.util.*;
public class Main {
    public static void main(String[] args){
        Scanner s=new Scanner(System.in);
        int a=s.nextInt();
        int b=s.nextInt();
        char c = s.next().trim().charAt(0);
        System.out.println(UserMainCode.calculator(a, b, c));
        s.close();
    }
}
```

### UserMainCode:

```
public class UserMainCode {
    public static int calculator(int a,int b,char c)
    {
        int a1=0;

        if(c=='*')
        {
            a1=a*b;
        }
        else if(c=='+')
        {
            a1=a+b;
        }
        else if(c=='-')
        {
            a1=a-b;
        }
        else if(c=='/')
        {
            a1=a/b;
        }
        else if(c=='%')
        {
            a1=a%b;
        }
        return a1;
    }
}
```

## 83. Scores

Write a program to read a integer array of scores, if 100 appears at two consecutive locations return true else return false.

Include a class UserMainCode with a static method **checkScores** which accepts the integer

array. The return type is boolean.

Create a Class Main which would be used to accept the integer array and call the static

method present in UserMainCode.

### **Input and Output Format:**

Input consists of an integer n which is the number of elements followed by n integer values.

Output consists of a string that is either 'TRUE' or 'FALSE'.

Refer sample output for formatting specifications.

### **Sample Input 1:**

```
3  
1  
100
```

```
100
```

### **Sample Output 1:**

```
TRUE
```

### **Sample Input 2:**

```
3  
100  
1  
100
```

### **Sample Output 2:**

```
FALSE
```

### **Main:**

```
import java.util.*;  
public class Main {  
    public static void main (String[] args)  
    {  
        Scanner sc = new Scanner(System.in);  
        int n = sc.nextInt();  
        int[] arr = new int[n];  
        for(int i=0;i<n;i++){  
            arr[i] = sc.nextInt();  
        }  
        System.out.println(UserMainCode.checkScores(arr, n));  
        sc.close();  
    }  
}
```

### **UserMainCode:**

```
public class UserMainCode {  
    public static boolean checkScores(int arr[], int n){  
        boolean b = false;  
        for(int i=0;i<n-1;i++){  
            if(arr[i] == 100){  
                if(arr[i+1] == 100){  
                    b = true;  
                    break;  
                }  
            }  
        }  
        return b;  
    }  
}
```

## 84. ArrayFront

Write a program to read a integer array and return true if one of the first 4 elements in the array is 9 else return false.

Note: The array length may be less than 4.

Include a class UserMainCode with a static method **scanArray** which accepts the integer array. The return type is true / false.

Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.

### Input and Output Format:

Input consists of an integer n which is the number of elements followed by n integer values.

Output consists of TRUE / FALSE.

Refer sample output for formatting specifications.

### Sample Input 1:

```
6  
1  
2  
3  
4  
5  
6
```

### Sample Output 1:

FALSE

### Sample Input 2:

```
3  
1  
2  
9
```

### Sample Output 2:

TRUE

### Main:

```
import java.util.*;  
  
public class Main {  
  
    public static void main(String []args) {  
  
        Scanner sc=new Scanner(System.in);  
  
        int s=sc.nextInt();  
  
        int []a=new int[s];  
  
        for(int i=0;i<s;i++)  
  
        {  
  
            a[i]=sc.nextInt();  
  
        }  
    }  
}
```

```

    if(UserMainCode.scanArray(a)==true)
        System.out.println("TRUE");
    else
        System.out.println("FALSE");
    sc.close();
}
}

```

### UserMainCode:

```

public class UserMainCode {
    public static boolean scanArray(int[] a)
    {
        int u=0,l=0;
        boolean b=false;
        if(a.length>=4)
            l=4;
        else
            l=a.length;
        for(int i=0;i<l;i++)
            if(a[i]==9)
                u=10;
            if(u==10)
                b=true;
        return b;
    }
}

```

## 85. Word Count

Given a string array (s) and non negative integer (n) and return the number of elements in the array which have same number of characters as the givent int N.

Include a class UserMainCode with a static method **countWord** which accepts the string array and integer. The return type is the string formed based on rules.

Create a Class Main which would be used to accept the string and integer and call the static method present in UserMainCode.

### Input and Output Format:

Input consists of a an integer indicating the number of elements in the string array followed the elements and ended by the non-negative integer (N).

Output consists of a integer .

Refer sample output for formatting specifications.

### Sample Input 1:

```

4
a
bb
b
ccc

```

1  
**Sample Output 1:**

2

**Sample Input 2:**

5

dog

cat

monkey

bear

fox

3

**Sample Output 2:**

3

**Main:**

```
import java.util.*;
public class Main {
    public static void main(String []args){
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        String[] str=new String[n];
        for(int i=0;i<n;i++)
        {
            str[i]=sc.next();
        }
        int c=sc.nextInt();
        System.out.println(UserMainCode.countWord(n,str,c));
        sc.close();
    }
}
```

**UserMainCode:**

```
public class UserMainCode {
    public static int countWord(int n,String str[],int c)
    {
        int count=0;
        for(int i=0;i<str.length;i++)
        {
            if(str[i].length()==c)
            {
                count++;
            }
        }
        return count;
    }
}
```

## 86. Find Distance

Write a Program that accepts four int inputs( $x_1, y_1, x_2, y_2$ ) as the coordinates of two points.  
Calculate the distance between the two points using the below formula.

Formula : square root of( $(x_1-x_2)^2 + (y_1-y_2)^2$ )

Then, Round the result to return an int

Include a class UserMainCode with a static method **findDistance** which accepts four integers. The return type is integer representing the formula.

Create a Class Main which would be used to accept the input integers and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of four integers.

Output consists of a single integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

```
3  
4  
5  
2
```

### **Sample Output 1:**

```
3
```

### **Sample Input 2:**

```
3  
1  
5  
2
```

### **Sample Output 2:**

```
2
```

### **Main:**

```
import java.util.*;  
public class Main {  
    public static void main (String[] args)  
    {  
        Scanner s=new Scanner(System.in);  
        int a=s.nextInt();  
        int b=s.nextInt();  
        int c=s.nextInt();  
        int d=s.nextInt();  
        System.out.println(UserMainCode.findDistance(a,b,c,d));  
        s.close();  
    }  
}
```

### **UserMainCode:**

```
public class UserMainCode {  
    public static int findDistance(int a,int b,int c,int d) {  
        long q=(int)Math.round(Math.sqrt(((a-c)*(a-c))+((b-d)*(b-d))));  
        return (int) q;  
    }  
}
```

## **87. Word Count - II**

Write a program to read a string and count the number of words present in it.

Include a class UserMainCode with a static method **countWord** which accepts the string.

The return type is the integer giving out the count of words.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of a string.

Output consists of integer.

Refer sample output for formatting specifications.

**Sample Input 1:**

Today is Sunday

**Sample Output 1:**

3

**Main:**

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.nextLine();
        UserMainCode.countWord(s1);
        s.close();
    }
}
```

**UserMainCode:**

```
import java.util.StringTokenizer;
public class UserMainCode {
    public static void countWord(String s1){
        StringTokenizer st=new StringTokenizer(s1, " ");
        int n=st.countTokens();
        System.out.println(n);
    }
}
```

## 88. Sum of Max & Min

Write a Program that accepts three integers, and returns the sum of maximum and minimum numbers.

Include a class UserMainCode with a static method getSumMaxMin which accepts three integers. The return type is integer representing the formula.

Create a Class Main which would be used to accept the input integers and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of three integers.

Output consists of a single integer.

Refer sample output for formatting specifications.

**Sample Input 1:**

12

17

19

**Sample Output 1:**

31

```

Main:
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        int a=s.nextInt();
        int b=s.nextInt();
        int c=s.nextInt();
        System.out.println(UserMainCode.getSumMaxMin(a,b,c));
        s.close();
    }
}

```

```

UserMainCode:
public class UserMainCode {
    public static int getSumMaxMin(int a,int b,int c)
    {
        int d=0;
        if(a<b&&b<c)
        {
            d=a+c;
        }
        else if(a<b&&b>c)
        {
            d=b+c;
        }
        else if(a>b&&b<c)
        {
            d=a+b;
        }
        return d;
    }
}

```

## 89. Decimal to Binary Conversion

Write a Program that accepts a decimal number n, and converts the number to binary. Include a class UserMainCode with a static method **convertDecimalToBinary** which accepts an integer. The return type is long representing the binary number. Create a Class Main which would be used to accept the input integer and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of single integer.

Output consists of a single long.

Refer sample output for formatting specifications.

### **Sample Input 1:**

5

### **Sample Output 1:**

101

### **MAIN**

```

import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        System.out.println(UserMainCode.convertDecimalToBinary(n));
    }
}

```

```
s.close();  
}  
}
```

### UserMainCode

```
public class UserMainCode {  
    public static long convertDecimalToBinary(int n){  
        String s1=Integer.toBinaryString(n);  
        long y=Long.parseLong(s1);  
    return y;  
}  
}
```

### 90.String Processing - V

Write a program to read a string and also a number N. Form a new string made up of n repetitions of the last n characters of the String. You may assume that n is between 1 and the length of the string.

Include a class UserMainCode with a static method **returnLastRepeatedCharacters** which

accepts the string and the number n. The return type is the string as per the problem statement.

Create a Class Main which would be used to accept the string and integer and call the static method present in UserMainCode.

#### Input and Output Format:

Input consists of a string and integer.

Output consists of a string.

Refer sample output for formatting specifications.

#### Sample Input 1:

Hello

2

#### Sample Output 1:

lolo

#### Sample Input 2:

Hello

3

#### Sample Output 2:

lloolloo

### MAIN

```
import java.util.Scanner;  
public class Main  
{  
  
    public static void main(String[] args)  
    {  
        Scanner s=new Scanner(System.in);  
        String s1=s.nextLine();  
        int n1=s.nextInt();  
        System.out.println(UserMainCode.returnLastRepeatedCharacters(s1,n1));  
        s.close();  
    }  
}
```

```
}
```

## USERMAINCODE

```
public class UserMainCode{
    public static String returnLastRepeatedCharacters(String s1, int n1)
{
StringBuffer sb = new StringBuffer();
for(int i = 0 ; i < n1 ; i++)
sb.append(s1.substring(s1.length()-n1, s1.length()));
return sb.toString();
}
```

## 91.Regular Expression - III

Given a string (s) apply the following rules.

1. String should not begin with a number.

If the condition is satisfied then print TRUE else print FALSE.

Include a class UserMainCode with a static method **validateString** which accepts the string.

The return type is the boolean formed based on rules.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### Input and Output Format:

Input consists of a string.

Output consists of TRUE or FALSE .

Refer sample output for formatting specifications.

### Sample Input 1:

ab2

### Sample Output 1:

TRUE

### Sample Input 2:

72CAB

### Sample Output 2:

FALSE

## MAIN

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args)

    {

        Scanner s=new Scanner(System.in);

        String s1=s.nextLine();
```

```

        if(UserMainCode.validateString(s1)==true)
System.out.println("TRUE");

        else System.out.println("FALSE");

        s.close();

    }

}

```

### **USERMAINCODE**

```

public class UserMainCode {
    public static boolean validateString(String s)
    {
        boolean b=false;
        if(s.charAt(0)=='0'||s.charAt(0)=='1'||s.charAt(0)=='2'||s.charAt(0)=='3'||s.charAt(0)=='4'||s.charAt(0)=='5'||s.charAt(0)=='6'||s.charAt(0)=='7'||s.charAt(0)=='8'||s.charAt(0)=='9'){
            b=false;
        }
        else
            b=true;
        return b;
    }
}

```

## **92.String Processing - TrimCat**

Write a program to read a string and return a new string which is made of every alternate characters starting with the first character. For example NewYork will generate Nwok, and Samurai will generate Smri.

Include a class UserMainCode with a static method getAlternateChars which accepts the string. The return type is the modified string.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

#### **Input and Output Format:**

Input consists of a string.

Output consists of string.

Refer sample output for formatting specifications.

#### **Sample Input 1:**

Hello

#### **Sample Output 1:**

Hlo

## **MAIN**

```

import java.util.Scanner;
public class Main
{
    public static void main(String[] args)

```

```

{
    Scanner s=new Scanner(System.in);
String s1=s.nextLine();
System.out.println(UserMainCode.getAlternateChars(s1));
s.close();
}
}

```

#### **USERMAINCODE**

```

public class UserMainCode{
public static String getAlternateChars(String s)
{
//String s1=s.replaceAll(" ", "");
StringBuffer sbf = new StringBuffer();
for(int i = 0; i < s.length() ; i=i+2)
{
sbf.append(s.charAt(i));
}
String str = sbf.toString();
return str;
}
}

```

### **93. String Processing - Username**

Write a program to read a valid email id and extract the username.

Note - user name is the string appearing before @ symbol.

Include a class UserMainCode with a static method fetchUserName which accepts the string. The return type is the modified string.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

#### **Input and Output Format:**

Input consists of a string.

Output consists of string.

Refer sample output for formatting specifications.

#### **Sample Input 1:**

[admin@xyz.com](mailto:admin@xyz.com)

#### **Sample Output 1:**

admin

### **MAIN CLASS**

```

import java.util.Scanner;

public class Main {
public static void main(String[] args) {
Scanner s=new Scanner(System.in);
String s1=s.nextLine();
System.out.println(UserMainCode.fetchUserName(s1));
s.close();
}
}

```

#### **USERMAINCODE**

```

import java.util.StringTokenizer;
public class UserMainCode {
    public static String fetchUserName(String s1) {
        StringTokenizer st=new StringTokenizer(s1,"@");
        String s2=st.nextToken();
        return(s2);
    }
}

```

## 94. String Processing - VII

Write a program to read a two strings and one int value(N). check if Nth character of first String from start and Nth character of second String from end are same or not. If both are same return true else return false.

Check need not be Case sensitive

Include a class UserMainCode with a static method **isEqual** which accepts the two strings and a integer n. The return type is the TRUE / FALSE.

Create a Class Main which would be used to read the strings and integer and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of two strings and an integer.

Output consists of TRUE / FALSE .

Refer sample output for formatting specifications.

### **Sample Input 1:**

AAAA

abab

2

### **Sample Output 1:**

TRUE

### **Sample Input 2:**

MNOP

QRST

3

### **Sample Output 2:**

FALSE

MAIN

```

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner s=new Scanner(System.in);

        String s1=s.nextLine();

        String s2=s.nextLine();

        int n=s.nextInt();
    }
}

```

```
boolean output=UserMainCode.isEqual(s1,s2,n);

System.out.println(output);

s.close();

}

}

USERMAINCODE

public class UserMainCode {

public static boolean isEqual(String s1,String s2,int n){

boolean a=false;

if(n<s1.length()&&n<s2.length())

{

char c=s1.charAt(n);

char d=s2.charAt(s2.length()-n);

String s3=Character.toString(c);

//System.out.println(s3);

String s4=Character.toString(d);

//System.out.println(s4);

if(s3.equalsIgnoreCase(s4))

{

a=true;

}
}
```

```
}

else

{

a=false;

}

}

return a;

}

}
```

## 95. Largest Difference

Write a program to read a integer array, find the largest difference between adjacent elements and display the index of largest difference.

### EXAMPLE:

input1: {2,4,5,1,9,3,8}

output1: 4 (here largest difference 9-1=8 then return index of 9 ie,4)

Include a class UserMainCode with a static method **checkDifference** which accepts the integer array. The return type is integer.

Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.

### Input and Output Format:

Input consists of an integer n which is the number of elements followed by n integer values.

Output consists of integer.

Refer sample output for formatting specifications.

### Sample Input 1:

```
7
2
4
5
1
9
3
8
```

### Sample Output 1:

```
4
```

### MAIN CLASS

```

import java.util.Scanner;

public class Main{
public static void main(String[] args) {
Scanner s=new Scanner(System.in);
int m=s.nextInt();
int[] n1=new int[m];
for(int i=0;i<m;i++){
    n1[i]=s.nextInt();
}
System.out.println(UserMainCode.checkDifference(n1));
s.close();
}
}

```

### USERMAIN CODE

```

public class UserMainCode {
public static int checkDifference(int[] n1){
int n2,n3=0,n4=0,i;
for(i=0;i<n1.length-1;i++){
n2=Math.abs(n1[i]-n1[i+1]);
if(n2>n3){
n3=n2;
n4=i+1; }}
return n4;
}
}

```

### 1.Start Case

Write a program to read a sentence in string variable and convert the first letter of each word to capital case. Print the final string.

Note: - Only the first letter in each word should be in capital case in final string.

Include a class **UserMainCode** with a static method **printCapitalized** which accepts a string.

The return type (String) should return the capitalized string.

Create a Class Main which would be used to accept a string and call the static method present in UserMainCode.

#### **Input and Output Format:**

Input consists of a strings.

Output consists of a String (capitalized string).

Refer sample output for formatting specifications.

#### **Sample Input:**

Now is the time to act!

#### **Sample Output:**

Now Is The Time To Act!

### MAIN CLASS

```

import java.util.Scanner;

```

```

public class Main {
    public static void main(String[] args){
        Scanner s=new Scanner(System.in);
        String s1=s.nextLine();
        System.out.println(UserMainCode.printCapitalized(s1));
        s.close();
    }
}

```

## USERMAIN CODE

```

import java.util.StringTokenizer;

public class UserMainCode{
    public static String printCapitalized(String s1){
        StringBuffer sb=new StringBuffer();
        StringTokenizer t=new StringTokenizer(s1, " ");
        while(t.hasMoreTokens()){
            String s2=t.nextToken();
            String s3=s2.substring(0,1);
            String s4=s2.substring(1, s2.length());
            sb.append(s3.toUpperCase()).append(s4).append(" ");
        }
        return sb.toString();
    }
}

```

## 2. Maximum Difference

Write a program to read an integer array and find the index of larger number of the two adjacent numbers with largest difference. Print the index.

Include a class **UserMainCode** with a static method **findMaxDistance** which accepts an integer array and the number of elements in the array. The return type (Integer) should return index.

Create a Class Main which would be used to accept an integer array and call the static method present in UserMainCode.

### Input and Output Format:

Input consists of  $n+1$  integers, where  $n$  corresponds the size of the array followed by  $n$  integers. Output consists of an Integer (index).

Refer sample output for formatting specifications.

### Sample Input :

```

6
4
8
6
1
9
4

```

### Sample Output :

```

4

```

[In the sequence 4 8 6 1 9 4 the maximum distance is 8 (between 1 and 9). The function should return the index of the greatest of two. In this case it is 9 (which is at index 4). output = 4.]

### Main:

```

import java.util.Scanner;
public class Main {
    public static void main(String[] args) {Scanner s=new Scanner(System.in);
    int n=s.nextInt();
    int a[]={};
    for(int i=0;i<n;i++)
    {
    a[i]=s.nextInt();
    }
    int max=UserMainCode.findMaxDistance(a);
    System.out.println(max);
    s.close();
    }
}

UserMainCode:
public class UserMainCode {
    static int findMaxDistance(int[] a)
    {

        int max=0,index=0;
        for(int i=0;i<19;i++)
        {
        int d=Math.abs(a[i]-a[i+1]);
        if(d>max)
        {
        max=d;
        if(a[i]>a[i+1])
        {
        index=i;
        }
        else
        {
        index=i+1;
        }
        }
        }
    }
    return index;
}
}

```

### 3. Palindrome - In Range

Write a program to input two integers, which corresponds to the lower limit and upper limit respectively, and find the sum of all palindrome numbers present in the range including the two numbers. Print the sum.

Include a class **UserMainCode** with a static method **addPalindromes** which accepts two integers. The return type (Integer) should return the sum if the palindromes are present, else return 0.

Create a Class Main which would be used to accept two integer and call the static method present in UserMainCode.

Note1 : A palindrome number is a number which remains same after reversing its digits.

Note2 : A single digit number is not considered as palindrome.

**Input and Output Format:**

Input consists of 2 integers, which corresponds to the lower limit and upper limit respectively.

Output consists of an Integer (sum of palindromes).

Refer sample output for formatting specifications.

### Sample Input :

130

150

### Sample Output :

272

**(131+141 = 272)**

### Main:

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        int n1=s.nextInt();
        int n2=s.nextInt();
        System.out.println(UserMainCode.addPalindromes(n1,n2));
        s.close();
    }
}
```

### UserMainCode:

```
public class UserMainCode {
    public static int addPalindromes(int n1,int n2){
        int sum=0;
        for(int i=n1;i<=n2;i++){
            int r=0,n3=i;
            while(n3!=0){
                r=(r*10)+(n3%10);
                n3=n3/10;
            }
            if(r==i)
                sum=sum+i;
        }
        return sum;
    }
}
```

## 4. PAN Card

Write a program to read a string and validate PAN no. against following rules:

1. There must be eight characters.
2. First three letters must be alphabets followed by four digit number and ends with alphabet
3. All alphabets should be in capital case.

Print "Valid" if the PAN no. is valid, else print "Invalid".

Include a class **UserMainCode** with a static method **validatePAN** which accepts a string.

The

return type (Integer) should return 1 if the string is a valid PAN no. else return 2.

Create a Class Main which would be used to accept a string and call the static method

present in UserMainCode.

### **Input and Output Format:**

Input consists of a string, which corresponds to the PAN number.

Output consists of a string - "Valid" or "Invalid"

Refer sample output for formatting specifications.

### **Sample Input 1:**

ALD3245E

### **Sample Output 1:**

Valid

### **Sample Input 2:**

OLE124F

### **Sample Output 2:**

Invalid

### **Main:**

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.nextLine();
        UserMainCode.validatePAN(s1);
        s.close();
    }
}
```

### **UserMainCode:**

```
public class UserMainCode {public static void validatePAN(String s1) {
    if(s1.matches("[A-Z]{3}[0-9]{4}[A-Z]{1}"))
    {
        System.out.println("Valid");
    }
    else
        System.out.println("Invalid");
}}
```

## **5. Fibonacci Sum**

Write a program to read an integer n, generate fibonacci series and calculate the sum of first n numbers in the series. Print the sum.

Include a class **UserMainCode** with a static method **getSumOfNfibos** which accepts an integer n. The return type (**Integer**) should return the sum of n fibonacci numbers.

Create a Class Main which would be used to accept an integer and call the static method present in UserMainCode.

**Note:** First two numbers in a Fibonacci series are 0, 1 and all other subsequent numbers are sum of its previous two numbers. Example - 0, 1, 1, 2, 3, 5...

### **Input and Output Format:**

Input consists of an integer, which corresponds to n.

Output consists of an Integer (sum of fibonacci numbers).

Refer sample output for formatting specifications.

**Sample Input :**

5

**Sample Output :**

7

[0 + 1 + 1 + 2 + 3 = 7]

**Main:**

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        System.out.println(UserMainCode.getSumOfNfibos(n));
        s.close();
    }
}
```

**UserMainCode:**

```
public class UserMainCode {
    public static int getSumOfNfibos(int n){
        int a=-1,b=1,c=0,d=0;
    for(int i=0;i<n;i++)
    {
        c=a+b;
        d=d+c;
        a=b;
        b=c;
    }
    return d;
}
}
```

## 6. Test Vowels

Write a program to read a string and check if given string contains exactly five vowels in any order. Print “Yes” if the condition satisfies, else print “No”.

Assume there is no repetition of any vowel in the given string and all characters are lowercase.

Include a class **UserMainCode** with a static method **testVowels** which accepts a string.

The

return type (Integer) should return 1 if all vowels are present, else return 2.

Create a Class Main which would be used to accept a string and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of a string.

Output consists of a string (“Yes” or “No”).

Refer sample output for formatting specifications.

**Sample Input 1:**

acbisouzze

**Sample Output 1:**

Yes

**Sample Input 2:**

cbisouzze

**Sample Output 2:**

No

**Main:**

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.nextLine();
        int b=UserMainCode.testVowels(s1);
        if(b==1)
            System.out.println("Yes");
        else
            System.out.println("No");
        s.close();
    }
}
```

**UserMainCode:**

```
public class UserMainCode {

    public static int testVowels(String s1) {
        int b;
        int n1=0,n2=0,n3=0,n4=0,n5=0;
        String s2=s1.toLowerCase();
        for(int i=0;i<s2.length();i++){
            char c=s2.charAt(i);
            if(c=='a')
                n1++;
            if(c=='e')
                n2++;
            if(c=='i')
                n3++;
            if(c=='o')
                n4++;
            if(c=='u')
                n5++;
            if(n1==1&&n2==1&&n3==1&&n4==1&&n5==1)
                b=1;
            else b=2;
            return b;
        }

    }
}
```

**7.Dash Check**

Write a program to read two strings and check whether or not they have dashes in the same places. Print “Yes” if the condition satisfies, else print “No”. Include a class **UserMainCode** with a static method **compareDashes** which accepts two strings. The return type (Integer) should return 1 if all dashes are placed correctly, else return 2.

Create a Class Main which would be used to accept two strings and call the static method present in UserMainCode.

**Note:** The strings must have exactly the same number of dashes in exactly the same positions. The strings might be of different length.

**Input and Output Format:**

Input consists of two strings.

Output consists of a string ("Yes" or "No").

Refer sample output for formatting specifications.

**Sample Input 1:**

hi—there-you.

12--(134)-7539

**Sample Output 1:**

Yes

**Sample Input 2:**

-15-389

-xyw-zzy

**Sample Output 2:**

No

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.nextLine();
        String s2=s.nextLine();
        int p=UserMainCode.compareDashes(s1,s2);
        if(p==1)
            System.out.println("Yes");
        else
            System.out.println("No");
        s.close();
    }
}

import java.util.ArrayList;

public class UserMainCode {
    public static int compareDashes(String s1, String s2) {
        ArrayList<Integer>l1=new ArrayList<Integer>();
        for(int i=0;i<s1.length();i++)
        {
            if(s1.charAt(i)=='-')
            {
                l1.add(i);
            }
        }
        ArrayList<Integer>l2=new ArrayList<Integer>();
        for(int i=0;i<s2.length();i++)
        {
            if(s2.charAt(i)=='-')
            {
                l2.add(i);
            }
        }
        //System.out.println(l1);
        //System.out.println(l2);
        if(l1.equals(l2))
```

```

{
    return 1;
}
else
    return 2;
}
}

```

## 8.Reverse Split

Write a program to read a string and a character, and reverse the string and convert it in a format such that each character is separated by the given character. Print the final string. Include a class **UserMainCode** with a static method **reshape** which accepts a string and a character. The return type (String) should return the final string.

Create a Class Main which would be used to accept a string and a character, and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string and a character.

Output consists of a string (the final string).

Refer sample output for formatting specifications.

### **Sample Input:**

Rabbit

### **Sample Output:**

t-i-b-b-a-R

```

import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.nextLine();
        String s2=s.next();
        System.out.println(UserMainCode.reshape(s1,s2));
        s.close();
    }
}

public class UserMainCode {
    public static String reShape(String s,String s1){
        StringBuffer sb=new StringBuffer(s);
        StringBuffer sb2=new StringBuffer();
        String s2=sb.reverse().toString();
        for(int i=0;i<s2.length();i++)
        {
            sb2.append(s2.charAt(i));
            sb2.append(s1);
        }
        sb2.deleteCharAt(sb2.length()-1);
        //System.out.println(sb2.toString());
        return sb2.toString();
    }
}

```

## 9.Remove 10's

Write a program to read an integer array and remove all 10s from the array, shift the other elements towards left and fill the trailing empty positions by 0 so that the modified array is of the same length of the given array.

Include a class **UserMainCode** with a static method **removeTens** which accepts the number

of elements and an integer array. The return type (Integer array) should return the final array.

Create a Class Main which would be used to read the number of elements and the input array, and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of  $n+1$  integers, where  $n$  corresponds to size of the array followed by  $n$  elements of the array.

Output consists of an integer array (the final array).

Refer sample output for formatting specifications.

**Sample Input :**

```
5
1
10
20
10
2
```

**Sample Output :**

```
1
20
```

```
import java.io.*;
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int i;
        int n = sc.nextInt();
        int a[] = new int[n];
        for (i = 0; i < n; i++) {
            a[i] = sc.nextInt();
        }
        UserMainCode.removeTens(a);
        sc.close();
    }
}
```

```

}

import java.util.*;
public class UserMainCode {
public static void removeTens(int a[]){
    Scanner sc = new Scanner(System.in);

    int i,k = 0;
    int b[] = new int[a.length];
    ArrayList<Integer> al = new ArrayList<Integer>();
    for (i = 0; i <a.length; i++) {
        if (a[i] != 10) {
            al.add(a[i]);
        }
    }
    if (al.size() < a.length) {
        k = a.length- al.size();
        for (i = 0; i < k; i++) {
            al.add(0);
        }
    }
    int b1[] = new int[a.length];
    for (i = 0; i < a.length; i++) {
        b1[i] = al.get(i);
        System.out.println(b1[i]);
    }
}
}

```

## 10.Last Letters

Write a program to read a sentence as a string and store only the last letter of each word of

the sentence in capital letters separated by \$. Print the final string.

Include a class **UserMainCode** with a static method **getLastLetter** which accepts a string.

The return type (string) should return the final string.

Create a Class Main which would be used to read a string, and call the static method present

in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of a string (the final string).

Refer sample output for formatting specifications.

### **Smaple Input :**

This is a cat

### **Sample Output :**

\$\$\$\$A\$T

**Main:**

```
public class Main {
```

```

public static void main(String[] args) {
    Scanner s=new Scanner(System.in);
    String input=s.nextLine();
    System.out.println(UserMainCode.getLastLetter(input));

}

UserMainCode:

import java.util.*;
public class UserMainCode {
    public static String getLastLetter(String input){
        String str1=null;
        StringTokenizer st=new StringTokenizer(input, " ");
        StringBuffer sb=new StringBuffer();
        while(st.hasMoreTokens()){
            str1=st.nextToken();
            // String str2=Character.toString(str1.charAt(str1.length()-1));
            String str2=str1.substring(str1.length()-1);
            String str3= str2.toUpperCase();
            sb.append(str3).append("$");
        }sb.deleteCharAt(sb.length()-1);
        return sb.toString();
    }
}

```

## 11.Largest Key in HashMap

Write a program that constructs a hashmap and returns the value corresponding to the largest key.

Include a class UserMainCode with a static method **getMaxKeyValue** which accepts a string.

The return type (String) should be the value corresponding to the largest key.

Create a Class Main which would be used to accept Input string and call the static method

present in UserMainCode.

### **Input and Output Format:**

Input consists of  $2n+1$  values. The first value corresponds to size of the hashmap. The next  $n$

pair of numbers equals the integer key and value as string.

Output consists of a string which is the value of largest key.

Refer sample output for formatting specifications.

### **Sample Input 1:**

```

3
12
amron
9
Exide
7

```

SF

### Sample Output 1:

Amron

#### Main:

```
import java.util.*;
public class Main {
    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        HashMap<Integer, String> hm=new HashMap<Integer, String>();
        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        for(int i=0;i<n;i++)
        {
            int a=s.nextInt();
            String s1=s.next();
            hm.put(a,s1);

        }
        System.out.println(UserMainnode.getMaxKeyValue(hm));
    }
}
```

#### Hashmap:

```
import java.util.*;
import java.util.HashMap;
import java.util.Iterator;
public class UserMainnode {
    public static String getMaxKeyValue(HashMap<Integer, String> hm) {
        int max=0;
        String s3=null;
        Iterator<Integer> itr=hm.keySet().iterator();
        while(itr.hasNext())
        {
            int b=itr.next();
            if(b>max)
            {
                max=b;
                s3=hm.get(b);
            }
        }
        return (s3);
    }
}
```

#### 12.All Numbers

Write a program to read a string array and return 1 if all the elements of the array are

numbers, else return -1.

Include a class UserMainCode with a static method **validateNumber** which accepts a string

array. The return type (integer) should be -1 or 1 based on the above rules.

Create a Class Main which would be used to accept Input string array and call the static method present in UserMainCode.

The string array is said to be valid if all the elements in the array are numbers. Else it is invalid.

### **Input and Output Format:**

Input consists of an integer specifying the size of string array followed by n strings.

Refer sample output for formatting specifications.

### **Sample Input 1:**

```
4  
123  
24.5  
23  
one
```

### **Sample Output 1:**

invalid

### **Sample Input 2:**

```
2  
123  
24.5
```

### **Sample Output 2:**

valid

### **Main:**

```
import java.util.*;  
public class Main {  
    public static void main(String[] args) {  
        Scanner s = new Scanner(System.in);  
        int n = s.nextInt();  
        String[] s1 = new String[n];  
        for(int i=0;i<n;i++){  
            s1[i] = s.next();  
        }  
  
        int out=(userMainCode.validateNumber(s1));  
        System.out.println(out);  
    }  
}
```

### **UserMainCode:**

```
class userMainCode{  
    public static int validateNumber(String[] s1){  
        int b =0 ,count,out=0;  
        for(int i=0;i<s1.length;i++){  
            String s2 = s1[i];  
            if(s2.matches("[0-9]{1,}"))
```

```

    {
        count =0;
        for(int j=0;j<s2.length();j++)

        {
            char c = s2.charAt(j);
            if(c=='.')
                count++;
        }

        if(count>1)
            b=1;
    }
    else
        b=1;
}
if(b==0){
    out=1;
}
else out=-1;
return out;
}

}

```

### 13.Day of the Week

Write a program to read a date as string (MM-dd-yyyy) and return the day of week on that date.

Include a class UserMainCode with a static method **getDay** which accepts the string. The

return type (string) should be the day of the week.

Create a Class Main which would be used to accept Input string and call the static method

present in UserMainCode.

#### **Input and Output Format:**

Input consists of a string.

Output consists of a string.

Refer sample output for formatting specifications.

#### **Sample Input 1:**

07-13-2012

#### **Main:**

```

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;
public class Main {

    public static void main(String[] args) throws ParseException {
        Scanner sc=new Scanner(System.in);
        String s1=sc.nextLine();
        System.out.println(UserMainCode.getDay(s1));
    }
}

```

```

    // TODO Auto-generated method stub

}

UserMainCode:

import java.text.SimpleDateFormat;
import java.text.ParseException;
import java.util.Date;
public class UserMainCode {

public static String getDay(String s1) throws ParseException
{
SimpleDateFormat sdf=new SimpleDateFormat("MM-dd-yyyy");
SimpleDateFormat sdf1=new SimpleDateFormat("EEEE");
Date d=sdf.parse(s1);
String s=sdf1.format(d);
return s;
}
}

```

## 14.Max Substring

Write a program to accept two string inputs. The first being a source string and second one

a delimiter. The source string contains the delimiter at various locations. Your job is to return the substring with maximum number of characters. If two or more substrings have

maximum number of characters return the substring which appears first. The size of the delimiter is 1.

Include a class UserMainCode with a static method **extractMax** which accepts the string.

The return type (string) should be the max substring.

Create a Class Main which would be used to accept Input string and call the static method

present in UserMainCode.

### **Input and Output Format:**

Input consists of a source string and delimiter.

Output consists of a string.

Refer sample output for formatting specifications.

### **Sample Input 1:**

delhi-pune-patna

### **Sample Output 1:**

Delhi\

### **Main:**

```
import java.util.*;
```

```

public class Main {
    public static void main(String[] args){

        Scanner sc=new Scanner(System.in);
        String input1=sc.next();
        String input2=sc.next();
        System.out.println(UserMainCode.extractMax(input1,input2));

    }
}

```

**Usermaincode:**

```

import java.util.StringTokenizer;
import java.util.*;

public class UserMainCode {
    public static String extractMax(String input1,String input2){
        int max=0;
        String s3=null;
        StringTokenizer st=new StringTokenizer(input1,"-");
        while( st.hasMoreTokens())
        {
            String s2=st.nextToken();
            int n=s2.length();
            if(n>max)
            {
                max=n;
                s3=s2;
            }
        }
        return(s3);
    }
}

```

## 15. States and Capitals

Write a program that constructs a hashmap with “state” as key and “capital” as its value.

If

the next input is a state, then it should return capital\$state in lowercase.

Include a class UserMainCode with a static method **getCapital** which accepts a hashmap.

The return type is the string as given in the above statement

Create a Class Main which would be used to accept Input string and call the static method

present in UserMainCode.

### Input and Output Format:

Input consists of  $2n+2$  values. The first value corresponds to size of the hashmap. The next  $n$

pair of numbers contains the state and capital. The last value consists of the “state” input.

Output consists of a string as mentioned in the problem statement.

Refer sample output for formatting specifications.

**Sample Input 1:**

```
3
Karnataka
Bangaluru
Punjab
Chandigarh
Gujarat
Gandhinagar
Punjab
```

**Sample Output 1:**

```
chandigarh$punjab
```

**Main:**

```
import java.util.HashMap;
import java.util.Scanner;
public class Main {

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        HashMap<String, String> hm=new
        HashMap<String, String>();
        for(int i=0;i<n;i++)
        {
            String s1=sc.next();
            String s2=sc.next();
            hm.put(s1,s2);
        }
        String sa=sc.next();
        System.out.print(UserMainCode.getCapital(hm,sa));
    }
}
```

**UserMainCode:**

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
public class UserMainCode {
    public static String getCapital(HashMap<String, String>
hm, String sa)
    {
        String chan=null;
        Iterator<String>it=hm.keySet().iterator();
        StringBuffer sb=new StringBuffer();
        while(it.hasNext()){
            String a=it.next();
            if(a.equals(sa))
            {
                chan=hm.get(a);
            }
        }
        return chan;
    }
}
```

```

        sb.append(chan).append("$").append(sa);
    }
}
return sb.toString();
}
}

```

## 16.Simple String Manipulation - II

Write a program to read a string and return an integer based on the following rules.

If the first word and the last word in the String match, then return the number of characters

in the word else return sum of the characters in both words. Assume the Strings to be case - sensitive.

Include a class UserMainCode with a static method **calculateWordSum** which accepts a

string. The return type (integer) should be based on the above rules.

Create a Class Main which would be used to accept Input string and call the static method

present in UserMainCode.

### **Input and Output Format:**

Input consists of a string with maximum size of 100 characters.

Output consists of a string.

Refer sample output for formatting specifications.

### **Sample Input 1:**

COGNIZANT TECHNOLOGY SOLUTIONS COGNIZANT

### **Sample Output 1:**

9

### **Sample Input 2:**

HOW ARE YOU

### **Sample Output 2:**

6

### **Main:**

```

import java.util.*;
public class Main {

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        String inpList=sc.nextLine();
        System.out.println(UserMainCode.calculateWordSum(inpList));
    }
}

```

### **UserMainCode:**

```
import java.util.*;
```

```

public class UserMainCode {
    public static int calculateWordSum(String inp) {

        int count=0;
        String st[]=inp.split(" ");
        String s1=st[0];
        String slst=st[st.length-1];
        if(s1.equals(slst))
        {
            count=s1.length();
        }
        else
        {
            count=s1.length()+slst.length();
        }
        return count;
    }
}

```

## 17.Vowels, Arrays & ArrayLists

Write a program to read an array of strings and return an arraylist which consists of words

whose both first and last characters are vowels. Assume all inputs are in lowercase.

Include a class UserMainCode with a static method **matchCharacter** which accepts a string

array. The return type shoud be an arraylist which should contain elements as mentioned

above.

Create a Class Main which would be used to accept Input array and call the static method

present in UserMainCode.

### **Input and Output Format:**

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in

the array. The next 'n' string correspond to the elements in the array.

Output consists of strings which are elements of arraylist

Refer sample output for formatting specifications.

### **Sample Input 1:**

```

4
abcde
pqrs
abci
orto

```

### **Sample Output 1:**

```

abcde
abci
orto

```

**Main:**

```
package vowels;
import java.util.*;

public class Main {

    public static void main(String[] args) {

        int n;
        Scanner sc=new Scanner(System.in);
        n=Integer.parseInt(sc.nextLine());
        String[] str=new String[n];
        for(int i=0;i<n;i++)
        {
            str[i]=sc.nextLine();
        }
        ArrayList<String> arr=new ArrayList<String>();
        arr=UserMainCode.matchCharacter(str);
        Iterator<String> it=arr.iterator();
        while(it.hasNext())
        {
            System.out.println(it.next());
        }

    }
}
```

**Usermaincode:**

```
package vowels;
import java.util.*;
public class UserMainCode {

    public static ArrayList<String> matchCharacter (String[] ss)
    {
        ArrayList<String> as=new ArrayList<String>();
        for(int i=0;i<ss.length;i++)
        {
            String sp=ss[i];
            char[] mp=sp.toLowerCase().toCharArray();
            if((mp[0]=='a'||mp[0]=='e'||mp[0]=='i'||mp[0]=='o'||mp[0]=='u')&
&(mp[sp.length()-1]=='a'||mp[sp.length()-1]=='e'||mp[sp.length()-
1]=='i'||mp[sp.length()-
1]=='o'||mp[sp.length()-1]=='u'))
            {
                as.add(sp);
            }
        }
        return as;
    }
}
```

## 18.Transfer from Hashmap to ArrayList

Write a program that constructs a hashmap with “employee id” as key and “name” as its value. Based on the rules below, on being satisfied, the name must be added to the arraylist.

i)First character should be small and the last character should be Capital.

ii)In name at least one digit should be there.

Include a class UserMainCode with a static method **getName** which accepts a hashmap.

The

return type is an arraylist as expected in the above statement

Create a Class Main which would be used to accept Input string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of  $2n+1$  values. The first value corresponds to size of the hashmap. The next  $n$  pair of numbers contains the employee id and name.

Output consists of arraylist of strings as mentioned in the problem statement.

Refer sample output for formatting specifications.

### **Sample Input 1:**

4

1

ravi5raJ

2

sita8gitA

3

ram8sitA

4

rahul

### **Sample Output 1:**

ravi5raJ

sita8gitA

ram8sitA

### **main:**

```
import java.util.*;
import java.text.*;
public class Main {
public static void main(String[] args) {
    HashMap<Integer, String> hm1=new HashMap<Integer, String>();
    int n;
    Scanner sc=new Scanner(System.in);
    n=Integer.parseInt(sc.nextLine());
    for(int i=0;i<n;i++)
    {
        hm1.put(Integer.parseInt(sc.nextLine()),sc.nextLine());
    }
    ArrayList<String> al1=new ArrayList<String>();
    al1=UserMainCode.getName(hm1);
    Iterator<String> it=al1.iterator();
    while(it.hasNext())
    {
```

```

        System.out.println(it.next());
    }

}
}

```

**Usermaincode:**

```

import java.util.*;
import java.text.*;
public class UserMainCode {
    public static ArrayList<String> getName(HashMap<Integer, String> hm1)
    {
        ArrayList<String> al2=new ArrayList<String>();
        Iterator<Integer> it =hm1.keySet().iterator();
        while(it.hasNext())
        {
            int id=it.next();
            String name=hm1.get(id);
            if(name.matches("[a-z]{1,}.*[0-9]{1,}.*[A-Z]{1}"))
                al2.add(name);
        }
        return al2;
    }
}

```

## 19.Max Admissions

Write a program that reads details about number of admissions per year of a particular college, return the year which had maximum admissions. The details are stored in an arraylist with the first index being year and next being admissions count.

Include a class UserMainCode with a static method **getYear** which accepts a arraylist. The return type is an integer indicating the year of max admissions.

Create a Class Main which would be used to accept Input string and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of  $2n+1$  values. The first value corresponds to size of the data (year & admissions). The next  $n$  pair of numbers contains the year and admissions count.

Output consists of an integer as mentioned in the problem statement.

Refer sample output for formatting specifications.

**Sample Input 1:**

```

4
2010
200000
2011
300000
2012
45000
2013
25000

```

**Sample Output 1:**

2011

USERMAINCODE:

```
import java.util.ArrayList;

public class UserMainCode

{

public static int year (ArrayList<Integer> a1)

{

int max=0,pos=0;

for(int i=1;i<a1.size();i+=2)

{

if(a1.get(i)>max)

{

max=a1.get(i);

pos=i;

}

}

return a1.get(pos-1);

}

}
```

MAIN:

```
import java.util.*;

class Main

{

public static void main(String [] args)

{

Scanner s=new Scanner(System.in);

ArrayList<Integer> a1=new ArrayList<Integer>();
```

```

int n=s.nextInt();

n=n*2;

for(int i=0;i<n;i++)

{

a1.add(s.nextInt());

}

System.out.println(UserMainCode.year(a1));

s.close();

}

}

```

## 20.Sum Non Prime Numbers

Write a program to calculate the sum of all the non prime positive numbers less than or equal to the given number.

Note: prime is a natural number greater than 1 that has no positive divisors other than 1 and itself

Example:

input = 9

Prime numbers = 2,3,5 and 7

output =  $1+4+6+8+9=28$

Include a class **UserMainCode** with a static method “**addNumbers**” that accepts an integer

argument and returns an integer.

Create a class **Main** which would get an integer as input and call the static method **validateNumber** present in the UserMainCode.

**Input and Output Format:**

Input consists of an integer.

Output consists of an integer.

**Sample Input:**

9

**Sample Output:**

28

**Main:**

```

import java.util.*;

public class Main {
    public static void main(String[] args) {
    {
        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        System.out.println(UserMainCode.addNumbers(n));
    }
}

```

```

        }
    }

Usermaincode:

public class UserMainCode {
    public static int addNumbers(int n) {
        int sum=0;int k=0;int sum1=0;
        for(int i=1; i<=n; i++)
        { k=0;
        for(int j=1; j<=i; j++)
        {
        if(i%j==0)
        k++;
        }
        if(k!=2)
        {
        sum=sum+i;
        }
        }
        return sum;
    }
}

```

## 21.Date Format Conversion

Given a date string in the format dd/mm/yyyy, write a program to convert the given date to the format dd-mm-yy.

Include a class **UserMainCode** with a static method “**convertDateFormat**” that accepts a

String and returns a String.

Create a class **Main** which would get a String as input and call the static method **convertDateFormat** present in the UserMainCode.

### **Input and Output Format:**

Input consists of a String.

Output consists of a String.

### **Sample Input:**

12/11/1998

### **Sample Output:**

12-11-98

### **Main:**

```

import java.util.*;
import java.text.*;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.next();
        UserMainCode.convertDateFormat(s1);
    }
}

```

### Usermaincode:

```

import java.util.*;
import java.text.*;
public class UserMainCode {
    public static void convertDateFormate(String s1) {
        SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
        sdf.setLenient(false);
        try {
            Date d1=sdf.parse(s1);
            SimpleDateFormat sdf1=new SimpleDateFormat("dd-MM-yy");
            String s2=sdf1.format(d1);
            System.out.println(s2);
        } catch (ParseException e) {
            e.printStackTrace();
        }
    }
}

```

## 22.Valid Date

Given a date string as input, write a program to validate if the given date is in any of the following formats:

dd.mm.yyyy  
dd/mm/yy  
dd-mm-yyyy

Include a class **UserMainCode** with a static method “**validateDate**” that accepts a String and

returns an integer. This method returns 1 if the date is valid, else return -1.

Create a class **Main** which would get a String as input and call the static method **validateDate** present in the UserMainCode.

### **Input and Output Format:**

Input consists of a String.

Output consists of a String that is either 'Valid' or 'Invalid'.

### **Sample Input 1:**

12.03.2012

### **Sample Output 1:**

Valid

### **Sample Input 2:**

27#01#1977

### **Sample Output 2:**

Invalid

### **UserMainCode:**

```

public class UserMainCode

{
    public static int dateformat(String s1) throws ParseException
    {
        String s2="";

```

```
int n=-1;

if(s1.matches("[0-9]{2}[.]{1}[0-9]{2}[.]{1}[0-9]{4}"))

{

SimpleDateFormat sdf=new SimpleDateFormat("dd.MM.yyyy");

Date d=sdf.parse(s1);

s2=sdf.format(d);

n=1;

}

else if(s1.matches("[0-9]{2}[/]{1}[0-9]{2}[/]{1}[0-9]{2}"))

{

SimpleDateFormat sdf1=new SimpleDateFormat("dd/MM/yy");

Date d1=sdf1.parse(s1);

s2=sdf1.format(d1);

n=1;

}

else if(s1.matches("[0-9]{2}[-]{1}[0-9]{2}[-]{1}[0-9]{4}"))

{

SimpleDateFormat sdf2=new SimpleDateFormat("dd-MM-yyyy");

Date d2=sdf2.parse(s1);

s2=sdf2.format(d2);

n=1;

}

else

{

n=-1;

}

return n;
```

```

}

}

MAIN:

import java.text.ParseException;

import java.util.*;

class Main

{

public static void main(String [] args) throws ParseException

{

Scanner s=new Scanner(System.in);

String s1=s.next();

int b=UserMainCode.dateFormat(s1);

if(b==1)

{



System.out.println("Valid");

}

Else

{

System.out.println("Invalid");

}

s.close();

}

}

```

## 23.Convert Format

Given a 10 digit positive number in the format XXX-XXX-XXXX as a string input, write a program to convert this number to the format XX-XX-XXX-XXX.

Include a class **UserMainCode** with a static method “**convertFormat**” that accepts a String

argument and returns a String.

Create a class **Main** which would get a String as input and call the static method **convertFormat** present in the UserMainCode.

**Input and Output Format:**

Input consists of a String.

Output consists of a String.

**Sample Input:**

555-666-1234

**Sample Output:**

55-56-661-234

**Main:**

```
import java.util.*;
import java.text.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        String s=sc.next();
        System.out.println(UserMainCode.convertFormate(s));
    }
}
```

**Usermaincode:**

```
import java.util.*;
import java.text.*;
public class UserMainCode {
    public static String convertFormate(String s) {
        StringTokenizer t=new StringTokenizer(s,"-");
        String s1=t.nextToken();
        String s2=t.nextToken();
        String s3=t.nextToken();
        StringBuffer sb=new StringBuffer();
        sb.append(s1.substring(0, s1.length()-1)).append('-');
        sb.append(s1.charAt(s1.length()-1)).append(s2.charAt(0)).append('-');
        sb.append(s2.substring(1, s2.length()))).append(s3.charAt(0)).append('-');
        sb.append(s3.substring(1, s3.length())));
        return sb.toString();
    }
}
```

## 24.Add and Reverse

Given an int array and a number as input, write a program to add all the elements in the array greater than the given number. Finally reverse the digits of the obtained sum and print

it.

Include a class **UserMainCode** with a static method “**addAndReverse**” that accepts 2 arguments and returns an integer. The first argument corresponds to the integer array and the second argument corresponds to the number.

Create a class **Main** which would get the required input and call the static method **addAndReverse** present in the UserMainCode.

**Example:**

Input Array = {10,15,20,25,30,100}

Number = 15

sum = 20 + 25 + 30 + 100 = 175

output = 571

**Input and Output Format:**

The first line of the input consists of an integer that corresponds to the number of elements in the array.

The next n lines of the input consists of integers that correspond to the elements in the array.

The last line of the input consists of an integer that corresponds to the number.

Output consists of a single integer.

**Sample Input**

```
6
10
15
20
25
30
100
15
```

**Sample Output**

```
571
```

**Main:**

```
import java.util.*;

public class Main {

    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        int a[]=new int[n];
        for(int i=0;i<n;i++)
        {
            a[i]=sc.nextInt();
        }
        int b=sc.nextInt();
        System.out.println(UserMainCode.addAndReverse(n,b,a)) ;
        sc.close();
    }
}

Usermaincode:
import java.util.*;
public class UserMainCode {
```

```

public static int addAndReverse(int n,int b,int a[])
{
    int i=0,sum=0,r=0;
    for(i=0;i<a.length;i++)
    {
        if(a[i]>b)
        {
            sum=sum+a[i];
        }
    }
    System.out.println(sum);
    while(sum!=0)
    {
        r=((r*10)+(sum%10));
        sum=sum/10;
    }
    return r;
}
}

```

## 25.Next Year day

Given a date string in dd/mm/yyyy format, write a program to calculate the day which falls on the same date next year. Print the output in small case.

The days are sunday, monday, tuesday, wednesday, thursday, friday and saturday.

Include a class **UserMainCode** with a static method “**nextYearDay**” that accepts a String and

returns a String.

Create a class **Main** which would get a String as input and call the static method **nextYearDay** present in the UserMainCode.

### **Input and Output Format:**

Input consists of a String.

Output consists of a String.

### **Sample Input:**

13/07/2012

### **Sample Output:**

Saturday

#### **Main:**

```

import java.util.*;
import java.text.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        String s1=sc.next();
        UserMainCode u=new UserMainCode();
        {
            System.out.println(u.nextYearDay(s1));
        }
    }
}

```

#### **Usercodemain:**

```

import java.util.*;
import java.text.*;

```

```

public class UserMainCode
{
    public String nextYearDay(String s1)
    {
        String s=null;
        SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
        sdf.setLenient(false);
        try {
            Date d1=sdf.parse(s1);
            Calendar cal=Calendar.getInstance();
            cal.setTime(d1);
            cal.add(Calendar.YEAR, 1);
            Date d2=cal.getTime();
            SimpleDateFormat sdf1=new SimpleDateFormat("EEEE");
            s=sdf1.format(d2);
        }
        catch (ParseException e)
        {
            e.printStackTrace();
        }
        return s;
    }
}

```

## 26.Sum Squares of Digits

Write a program that accepts a positive number as input and calculates the sum of squares of individual digits of the given number.

Include a class **UserMainCode** with a static method “**getSumOfSquaresOfDigits**” that accepts an integer argument and returns an integer.

Create a class **Main** which would get an integer as input and call the static method **getSumOfSquaresOfDigits** present in the UserMainCode.

### **Input and Output Format:**

Input consists of an integer.

Output consists of an integer.

### **Sample Input:**

321

### **Sample Output:**

14

### **Main:-**

```

import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        UserMainCode.getSumOfSquaresOfDigits(n);
        s.close();
    }
}

```

### **UserMainCode:-**

```

import java.util.*;

```

```

public class UserMainCode {
    public static void getSumOfSquaresOfDigits(int n) {
        int a=n;
        int rem=0;
        int sum=0;
        while(a!=0)
        {
            rem=a%10;
            sum=sum+(rem*rem);
            a=a/10;
        }
        System.out.println(sum);
    }
}

```

## 27.Even and Odd Index Sum

Write a program that accepts a positive number as input and calculates the sum of digits at even indexes (say evenSum) and sum of digits at odd indexes (say oddSum) in the given number. If both the sums are equal , print 'yes', else print no.

Example:

input = 23050

evenSum =  $2 + 0 + 0 = 2$

oddSum =  $3 + 5 = 8$

output = no

Include a class **UserMainCode** with a static method “**sumOfOddEvenPositioned**” that accepts an integer and returns an integer. The method returns 1 if the 2 sums are equal. Else the method returns -1.

Create a class **Main** which would get an integer as input and call the static method **sumOfOddEvenPositioned** present in the UserMainCode.

### **Input and Output Format:**

Input consists of an integer.

Output consists of a string that is either “yes” or “no”.

### **Sample Input 1:**

23050

### **Sample Output 1:**

no

### **Sample Input 2:**

231

### **Sample Output 2:**

Yes

**Main:-**

```

import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        UserMainCode.sumOfOddEvenPositioned(n);
        sc.close();
    }
}

```

```
}
```

#### UserMainCode:-

```
import java.util.*;
public class UserMainCode {
public static void sumOfOddEvenPositioned(int n) {
int rem = 0, i = 0;
int a[] = new int[10];
while (n > 0) {
rem = n % 10;
a[i] = rem;
n = n / 10;
i++;
}
int sume = 0, sumo = 0;
for (int j = i - 1; j >= 0; j--) {
if(j%2!=0)
{
sumo = sumo + a[j];
}
else
{
sume = sume + a[j];
}
}
if (sume == sumo) {
System.out.println("Yes");
} else
System.out.println("No");
}
}
```

## 28.Remove 3 Multiples

Write a program that accepts an ArrayList of integers as input and removes every 3rd element and prints the final ArrayList.

Suppose the given arrayList contains 10 elements remove the 3rd, 6th and 9th elements.  
Include a class **UserMainCode** with a static method “**removeMultiplesOfThree**” that accepts

an ArrayList<Integer> as argument and returns an ArrayList<Integer>.

Create a class **Main** which would get the required input and call the static method **removeMultiplesOfThree** present in the UserMainCode.

#### Input and Output Format:

The first line of the input consists of an integer n, that corresponds to the number of elements to be added in the ArrayList.

The next n lines consist of integers that correspond to the elements in the ArrayList.

Output consists of an ArrayList of integers.

#### Sample Input:

```
6
3
1
11
```

19

17

19

### Sample Output

3

1

19

17

### Main:-

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Scanner;
public class Main{
public static void main(String []args){
Scanner sc=new Scanner(System.in);
ArrayList<Integer> al=new ArrayList<Integer>();
ArrayList<Integer> al1=new ArrayList<Integer>();
int n=Integer.parseInt(sc.nextLine());
for(int i=0;i<n;i++)
{
al.add(sc.nextInt());
}
al1=UserMainCode.removeMultiplesOfThree(al);
Iterator it=al1.iterator();
while(it.hasNext())
{
System.out.println(it.next());
}
}
}
```

### UserMainCode:-

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.StringTokenizer;
public class UserMainCode
{
public static ArrayList<Integer> removeMultiplesOfThree(ArrayList<Integer> al)
{
ArrayList<Integer> al2=new ArrayList<Integer>();
for(int i=0;i<al.size();i++)
{
if((i+1)%3!=0)
al2.add(al.get(i));
}
return al2;
}
}
```

## 29.String Occurrences - II

Obtain two strings S1,S2 from user as input. Your program should count the number of times S2 appears in S1.

Return the count as output. Note - Consider case.

Include a class UserMainCode with a static method **getSubstring** which accepts two string variables. The return type is the count.

Create a Class Main which would be used to accept two Input strings and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of two strings with maximum size of 100 characters.

Output consists of an integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

catcowcat

cat

### **Sample Output 1:**

2

### **Sample Input 2:**

catcowcat

CAT

### **Sample Output 2:**

0

### **Main:-**

```
MAIN  
import java.util.Scanner;
```

```
public class Main {  
  
    public static void main(String[] args){  
  
        Scanner sc=new Scanner(System.in);  
  
        String s=sc.nextLine();  
  
        String s1=sc.nextLine();  
  
        System.out.println(UserMainCode.getSubstring(s, s1));  
  
        sc.close();  
  
    }  
  
}
```

#### USERMAINCODE

```
public class UserMainCode{  
  
    public static int getSubstring(String s, String s1){  
  
        int t=s1.length();  
  
  
        int count=0;  
  
        for(int i=0;i<s.length()-t+1;i++)  
  
        {  
  
            String s3=s.substring(i,t+i);  
  
            if(s3.equals(s1))  
  
            {  
  
                count++;  
  
            }  
  
        }  
  
        return count;  
  
    }  
  
}
```

## **30. Programming Logic**

Write a Program that accepts three integer values (a,b,c) and returns their sum. However, if one of the values is 13 then it does not count towards the sum and the next number also does not count. So for example, if b is 13, then both b and c do not count.

Include a class UserMainCode with a static method **getLuckySum** which accepts three integers. The return type is integer representing the sum.

Create a Class Main which would be used to accept the input integers and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of three integers.

Output consists of a single integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

1  
2  
3

### **Sample Output 1:**

6

### **Sample Input 2:**

1  
2  
13

### **Sample Output 2:**

3

### **Sample Input 3:**

13

3

8

### Sample Output 3:

8

Main:-

```
import java.util.Scanner;
public class Main{
public static void main(String[] args){
Scanner s=new Scanner(System.in);
int a=s.nextInt();

int b=s.nextInt();
int c=s.nextInt();
System.out.println(UserMainCode.LuckySum(a,b,c));
}
}
```

UserMainCode:-

```
public class UserMainCode{
public static int luckySum(int a, int b, int c)
{
if(a == 13)
return 0;
if(b == 13)
return a;
if(c == 13)
return (a + b);
return (a + b + c);
}
}
```

## 31.Triplets

Given an integer array, Write a program to find if the array has any triplets. A triplet is a value if it appears 3 consecutive times in the array.

Include a class UserMainCode with a static method **checkTriplets** which accepts an integer

array. The return type is boolean stating whether its a triplet or not.

Create a Class Main which would be used to accept the input array and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of  $n+1$  integers. The first integer would represent the size of array and the next  $n$  integers would have the values.

Output consists of a string stating TRUE or FALSE.

Refer sample output for formatting specifications.

**Sample Input 1:**

7

3

3

5

5

5

2

3

**Sample Output 1:**

TRUE

**Sample Input 2:**

7

5

3

5

1

5

2

3

### Sample Output 2:

FALSE

#### Main:-

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        int n;
        Scanner sc=new Scanner(System.in);
        n=sc.nextInt();
        int[] a=new int[n];
        for(int i=0;i<n;i++)
        {
            a[i]=sc.nextInt();
        }
        boolean s=UserMainCode.checkTriples(a);
        if(s==true)
            System.out.println("TRUE");
        else
            System.out.println("FALSE");
    }
}
```

#### UserMainCode:-

```
import java.util.*;

public class UserMainCode {
    public static boolean checkTriples(int[] a)
    {
```

```
boolean b=false;

for(int i=0;i<a.length-2;i++)
{
    if((a[i]==a[i+1])&&(a[i+1]==a[i+2]))
    {
        b=true;
    }
}

return b;
}
```

## 32.Repeat Front

Given a string (s) and non negative integer (n) apply the following rules.

1. Display the first three characters as front.
2. If the length of the string is less than 3, then consider the entire string as front and repeat it n times.

Include a class UserMainCode with a static method **repeatFirstThreeCharacters** which accepts the string and integer. The return type is the string formed based on rules.

Create a Class Main which would be used to accept the string and integer and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string and integer.

Output consists of a string .

Refer sample output for formatting specifications.

### **Sample Input 1:**

Coward

2

### **Sample Output 1:**

CowCow

### **Sample Input 2:**

So

3

### **Sample Output 2:**

SoSoSo

**Main:-**

```
import java.util.*;

public class Main {

    public static void main(String []args){

        Scanner sc=new Scanner(System.in);

        String s=sc.nextLine();

        int n=Integer.parseInt(sc.nextLine());

        System.out.println(UserMainCode.repeatFirstThreeCharacters(s,n));

        sc.close();

    }

}
```

**UserMaincode:-**

```
import java.util.*;

public class UserMainCode

{

    public static String repeatFirstThreeCharacters(String s,int n)

{
```

```
StringBuffer sb=new StringBuffer();
StringBuffer sb1=new StringBuffer();
if(s.length()>3)
{ sb.append(s.substring(0,3));
s=sb.toString();
}
for(int i=0;i<n;i++)
sb1.append(s);
return sb1.toString();
}
```

### 33.Sorted Array

Write a program to read a string array, remove duplicate elements and sort the array.

Note:

1. The check for duplicate elements must be case-sensitive. (AA and aa are NOT duplicates)

2. While sorting, words starting with upper case letters takes precedence.

Include a class UserMainCode with a static method **orderElements** which accepts the string

array. The return type is the sorted array.

Create a Class Main which would be used to accept the string array and integer and call the static method present in UserMainCode.

#### **Input and Output Format:**

Input consists of an integer n which is the number of elements followed by n string values.

Output consists of the elements of string array.

Refer sample output for formatting specifications.

**Sample Input 1:**

6

AAA

BBB

AAA

AAA

CCC

CCC

**Sample Output 1:**

AAA

BBB

CCC

**Sample Input 2:**

7

AAA

BBB

aaa

AAA

Abc

A

b

**Sample Output 2:**

A

AAA

Abc

BBB

aaa

b

**Main:-**

```
import java.util.*;

public class Main

{
    public static void main(String[] args)
    {
        int n;
        Scanner sin = new Scanner(System.in);
        n = sin.nextInt();
        String[] a1 = new String[n];
        for(int i=0;i<n;i++)
        {
            a1[i] = sin.next();
        }
        a1 = UserMainCode.orderElements(a1);
        for(int i=0;i<a1.length;i++)
        System.out.println(""+a1[i]);
    }
}
```

**UserMainCode:-**

```
import java.util.*;
```

```

public class UserMainCode
{
    public static String[] orderElements(String[] arr)
    {
        HashSet<String> al=new HashSet<String>();
        for(int i=0;i<arr.length;i++)
        {
            al.add(arr[i]);
        }

        Iterator<String> itr=al.iterator();
        String ar[] = new String[al.size()];
        int i =0 ;
        while(itr.hasNext()){
            ar[i] = itr.next();
            i++;
        }

        Arrays.sort(ar);
        return ar;
    }
}

```

### 34.Pattern Matcher

Write a program to read a string and check if it complies to the pattern 'CPT-XXXXXX' where XXXXXX is a 6 digit number. If the pattern is followed, then print TRUE else print FALSE.

Include a class UserMainCode with a static method **CheckID** which accepts the string. The return type is a boolean value.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

## **Input and Output Format:**

Input consists of a string.

Output should print TRUE or FALSE .

Refer sample output for formatting specifications.

### **Sample Input 1:**

CPT-302020

### **Sample Output 1:**

TRUE

### **Sample Input 2:**

CPT123412

### **Sample Output 2:**

FALSE

Main:

```
import java.util.*;

public class Main

{

    public static void main(String[] args)

    {

        Scanner sc = new Scanner(System.in);

        String s = sc.next();

        System.out.println(UserMainCode.CheckID(s));

        sc.close();

    }

}

UserMainCode:

public class UserMainCode

{

    public static boolean CheckID(String s)
```

```

{
boolean b=false;

if(s.matches("(CPT)[-]{1}[0-9]{6}"))
{
b=true;
}

else
{
b=false;
}

return b;
}
}

```

### 35.Playing with String - I

Given a string array and non negative integer (n) apply the following rules.

1. Pick nth character from each String element in the String array and form a new String.
2. If nth character not available in a particular String in the array consider \$ as the character.
3. Return the newly formed string.

Include a class UserMainCode with a static method **formString** which accepts the string and

integer. The return type is the string formed based on rules.

Create a Class Main which would be used to accept the string and integer and call the static method present in UserMainCode.

#### **Input and Output Format:**

Input consists of a an integer which denotes the size of the array followed by the array of strings and an integer (n).

Output consists of a string .

Refer sample output for formatting specifications.

**Sample Input 1:**

4

ABC

XYZ

EFG

MN

3

**Sample Output 1:**

CZG\$

Main:

```
import java.util.Scanner;

public class Main

{
    public static void main(String[] args)

    {
        Scanner s=new Scanner(System.in);

        int n=Integer.parseInt(s.nextLine());

        String[] sc=new String[n];

        for(int i=0;i<n;i++)

        {
            sc[i]=s.nextLine();
        }

        int a=Integer.parseInt(s.nextLine());

        System.out.println(UserMainCode.formString(n,sc,a));

        s.close();
    }
}
```

```

UserMainCode:

public class UserMainCode {

    public static String formString(int n, String[] input, int a)
    {
        StringBuffer sb=new StringBuffer();
        for(int i=0;i<n;i++)
        {
            if(input[i].length()>=a)
            {
                String a1=input[i];
                sb.append(a1.charAt(a-1));
            }
            else
            {
                sb.append('$');
            }
        }
        return sb.toString();
    }
}

```

## 36. Regular Expression - 1

Given a string (s) apply the following rules.

1. String should be only four characters long.
2. First character can be an alphabet or digit.
3. Second character must be uppercase 'R'.
4. Third character must be a number between 0-9.

If all the conditions are satisfied then print TRUE else print FALSE.

Include a class UserMainCode with a static method **validate** which accepts the string. The return type is the boolean formed based on rules.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of TRUE or FALSE .

Refer sample output for formatting specifications.

### **Sample Input 1:**

vR4u

### **Sample Output 1:**

TRUE

### **Sample Input 2:**

vRau

### **Sample Output 2:**

FALSE

### **Sample Input 3:**

vrau

### **Sample Output 3:**

FALSE

```
S.36) import java.util.Scanner;  
  
public class Main {  
  
    public static void main(String []args){  
  
        Scanner sc=new Scanner(System.in);  
  
        String n=sc.nextLine();  
  
        System.out.println(UserMainCode.validate(n));  
  
        sc.close();  
    }  
}
```

```

public class UserMainCode
{
    public static String validate(String s)
    {
        String w="FALSE";
        if(s.length()==4 &&
            (Character.isDigit(s.charAt(0))||Character.isAlphabetic(s.charAt(0)))&&s.charAt(1)
            =='R')
        {
            if(Character.isDigit(s.charAt(2)))
            w="TRUE";
        }
        return w;
    }
}

```

### **37.Regular Expression – 2 (Age Validator)**

Given the age of a person as string, validate the age based on the following rules.

1. Value should contain only numbers.
2. Value should be non-negative.
3. Value should be in the range of 21 to 45'.

If all the conditions are satisfied then print TRUE else print FALSE.

Include a class UserMainCode with a static method **ValidateAge** which accepts the string.

The return type is the boolean formed based on rules.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

#### **Input and Output Format:**

Input consists of a string.

Output consists of TRUE or FALSE .

Refer sample output for formatting specifications.

**Sample Input 1:**

23

**Sample Output 1:**

TRUE

**Sample Input 2:**

-34

**Sample Output 2:**

FALSE

**Sample Input 3:**

3a

**Sample Output 3:**

FALSE

AcB/TRUE

Main:

```
import java.util.*;
public class Main {
    public static void main(String[] args){
        Scanner s=new Scanner(System.in);
        //Regular Expression - 2
        (Age Validator) pg.No:150
        String n=s.nextLine();
        boolean b=UserMainCode.ValidateAge(n);
        if(b==true)
        {
            System.out.println("TRUE");
        }
        else
            System.out.println("FALSE");
```

```

        s.close();
    }
}

UserMainCode:

public class UserMainCode {
    public static boolean ValidateAge(String n)
    {
        boolean b = false;
        if(n.matches("[0-9]{2}"))
        {
            //Regular Expression - 2
            (Age Validator) pg.No:150
            int a=Integer.parseInt(n);
            if(a>0&&a>=21&&a<=45)
            {
                b=true;
            }
            else
                b=false;
        }
        return b;
    }
}

```

### 38. Regular Expression – 3 (Phone Validator)

Given a phone number as string, validate the same based on the following rules.

1. Value should contain only numbers.
2. Value should contain 10 digits.
3. Value should not start with 00.

If all the conditions are satisfied then print TRUE else print FALSE.

Include a class UserMainCode with a static method **validatePhone** which accepts the string.

The return type is the boolean formed based on rules.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of TRUE or FALSE .

Refer sample output for formatting specifications.

### **Sample Input 1:**

9987684321

### **Sample Output 1:**

TRUE

### **Sample Input 2:**

0014623452

### **Sample Output 2:**

FALSE

Main:

```
import java.util.*;
public class Main {
public static void main(String[]args){
Scanner s=new Scanner(System.in);
String s1=s.nextLine();
boolean b1=UserMainCode.validatePhone(s1);
if(b1==true)
{
System.out.println("TRUE"); //phone validation pq.no:151
}
else
{
System.out.println("FALSE");
}
s.close();
}
}
```

**UserMainCode:**

```
public class UserMainCode {  
    public static boolean validatePhone(String s1)  
    {  
        boolean b=false;  
        if(s1.matches("[0]{1}[0]{1}[0-9]{8}"))  
        {  
            b=false;  
        }  
        //phone validation pg.no:151  
    }  
    else  
    {  
        b=true;  
    }  
    return b;  
}
```

**39.String Splitter**

Write a program which would accept a string and a character as a delimiter. Apply the below rules

1. Using the delimiter, split the string and store these elements in array.
2. Reverse each element of the string and convert it into lowercase.

Include a class UserMainCode with a static method **manipulateLiteral** which accepts the string and character. The return type is the string array formed.

Create a Class Main which would be used to accept the string and character and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of a string and character.

Output consists of a string array.

Refer sample output for formatting specifications.

**Sample Input 1:**

AAA/bba/ccc/DDD

/

**Sample Output 1:**

```
aaa
abb
ccc
ddd
import java.util.*;
public class Main
{
    public static void main(String[] args)
    {
        Scanner s = new Scanner(System.in);
        String ip1=s.next();
        char ip2='/';
        String op[]=UserMainCode.manipulateLiteral(ip1,ip2);
        for(int i=0;i<op.length;i++)
            System.out.println(op[i]);
        s.close();
    }
    import java.util.ArrayList;
    import java.util.StringTokenizer;
    public class UserMainCode
    {
        public static String[] manipulateLiteral(String ip1, char ip2)
        {
            StringTokenizer t1 = new StringTokenizer(ip1,"/");
            ArrayList<String> lst = new ArrayList<String>();
            while(t1.hasMoreTokens())
            {
                StringBuffer sb = new StringBuffer();
                sb.append(t1.nextToken().toLowerCase());
                lst.add(sb.toString());
            }
        }
    }
}
```

```
lst.add(sb.reverse().toString());  
}  
  
String[] op = new String[lst.size()];  
  
for(int i = 0;i<lst.size();i++)  
{  
  
op[i] = lst.get(i);  
  
}  
  
return op;  
}  
}
```

## 40.Vowel Count

Write a program to read a string and count the number of vowels present in it.

Include a class UserMainCode with a static method **tellVowelCount** which accepts the string. The return type is the integer giving out the count of vowels.

Note: The check is case-insensitive.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

NewYork

### **Sample Output 1:**

2

### **Sample Input 2:**

Elephant

## Sample Output 2:

```
3

import java.util.*;

public class Main

{

    public static void main(String[] args) // Second set: 40.Vowel
Count//



    {

Scanner sc=new Scanner(System.in);

String s=sc.nextLine();

int max=UserMainCode.tellVowelCount(s);

System.out.println(max);

sc.close();

}

}

public class UserMainCode {

    public static int tellVowelCount(String s)

    {

        int max=0;

        int count=0;

        for(int i=0;i<s.length();i++)

        {

            char c=s.charAt(i);

            if(c=='a' || c=='e' || c=='i' || c=='o' || c=='u' || c=='A' || c=='E' || c=='I' ||
c=='O' || c=='U')

            {

                count++;

            }

        }

        if(count>max)
```

```
{  
    max=count;  
}  
  
return max;  
}  
}
```

## 41.Playing with String - II

Write a program to accept a string array as input, convert all the elements into lowercase and sort the string array. Display the sorted array.

Include a class UserMainCode with a static method **sortArray** which accepts the string array.

The return type is the string array formed based on requirement.

Create a Class Main which would be used to accept the string array and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a an integer which denotes the size of the array followed by the array of strings,

Output consists of a string array.

Refer sample output for formatting specifications.

### **Sample Input 1:**

5

AAA

BB

CCCC

A

ABCDE

## Sample Output 1:

```
a  
aaa  
abcde  
bb  
cccc  
import java.util.*;  
public class Main {  
    public static void main(String[] args) {  
        Scanner s=new Scanner(System.in);  
        int n=s.nextInt();  
        String s1[]=new String[n];  
        String s2[]=new String[n];  
        for(int i=0;i<n;i++)  
            //S.41.Playing with  
String - II//  
        {  
            s1[i]=s.next();  
        }  
        s2=UserMainCode.sortArray(s1,n);  
        for (int i = 0; i < n; i++) {  
            System.out.println(s2[i]);  
        }  
        s.close();  
    }  
import java.util.Arrays;  
public class UserMainCode {  
    public static String[] sortArray(String s1[],int n){
```

```

String s2[] = new String[n];

for (int i = 0; i < n; i++)

{

    s2[i] = s1[i].toLowerCase();

}

Arrays.sort(s2);

return s2;

}
}

```

## 42. Median Calculation

Write a program to accept an int array as input, and calculate the median of the same.

Median Calculation Procedure:

1. Sort the sequence of numbers.
2. The total number count is odd, Median will be the middle number.

The total number count is even, Median will be the average of two middle numbers, After calculating the average, round the number to nearest integer.

Include a class UserMainCode with a static method **calculateMedian** which accepts the int array. The return type is the integer which would be the median.

Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a an integer which denotes the size of the array followed by the array of integers.

Output consists of a integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

7

1

2

1

4

7

1

2

**Sample Output 1:**

2

**Sample Input 2:**

6

52

51

81

84

60

88

**Sample Output 2:**

71

Main

```
import java.util.*;  
  
public class Main  
{  
    public static void main(String[] args)  
    {
```

```
int n;  
Scanner sin = new Scanner(System.in);  
n = sin.nextInt();  
int[] a1 = new int[n];  
for(int i=0;i<n;i++)  
{  
    a1[i] = sin.nextInt();  
}  
System.out.println(""+UserMainCode.calculateMedian(a1));  
sin.close();  
}  
}
```

### UserMainCode

```
import java.util.Arrays;  
  
public class UserMainCode  
{  
    public static int calculateMedian(int[] a)  
    {  
        Arrays.sort(a);  
        int length = a.length;  
        int result=0,mid=0,midNext=0;  
        if((length%2) != 0)  
        {  
            mid = (length/2)+1;  
            result = a[mid];  
        }  
        else  
        {  
            mid = length/2;  
        }  
    }  
}
```

```

        midNext = mid+1;

        float add = a[mid-1]+a[midNext-1];

        float div = add/2;

        result = Math.round(div);

    }

    return result;
}

}

```

### **43.Sequence in Array**

Write a program to accept an int array as input, and check if [1,2,3] appears somewhere in the same sequence.

Include a class UserMainCode with a static method **searchSequence** which accepts the int

array. The return type is a boolean which returns true or false.

Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.

#### **Input and Output Format:**

Input consists of a an integer which denotes the size of the array followed by the array of integers.

Output should print true or false.

Refer sample output for formatting specifications.

#### **Sample Input 1:**

9

11

-2

5

1  
2  
3  
4  
5  
6

**Sample Output 1:**

TRUE

**Sample Input 2:**

6  
-2  
5  
1  
3  
2  
6

**Sample Output 2:**

FALSE

Main

```
import java.util.*;  
  
public class Main  
{  
    public static void main(String[] args)  
    {  
        Scanner s=new Scanner(System.in);  
        int n=s.nextInt();  
        int a[]={};
```

```

for(int i=0;i<n;i++){
    a[i]=s.nextInt();
}
System.out.println(UserMainCode.searchsequence(a));
s.close();
}
}

UserMainCode

public class UserMainCode {
    public static boolean searchsequence(int[] a)
    {
        boolean b = false;
        for(int i = 0 ; i< a.length-3; i++)
        {
            if(a[i]==1 && a[i+1]==2 && a[i+2]==3)
                b = true;
        }
        return b;
    }
}

```

#### 44.Asterisk & Characters

Write a program to read a string and return true or false based on the below rule:

1. Return true if for every '\*' in the string, there are same characters both side immediately before and after the star, else return false.

Include a class UserMainCode with a static method **scanStarNeighbors** which accepts the

string. The return type is the boolean TRUE or FALSE based on the rule.

Note: The check is case-insensitive.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of TRUE or FALSE.

Refer sample output for formatting specifications.

### **Sample Input 1:**

Hello\*World

### **Sample Output 1:**

FALSE

### **Sample Input 2:**

Welcome\*elizabeth

### **Sample Output 2:**

TRUE

Main

```
import java.util.*;

public class Main {

    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);

        String input=s.next();

        System.out.println( UserMainCode. scanStarNeighbors (input));
        s.close();
    }

}

UserMainCode

import java.util.StringTokenizer;
```

```

public class UserMainCode {

    public static boolean scanStarNeighbors (String input) {

        boolean b=false;

        StringTokenizer t=new StringTokenizer(input,"*");

        String s1=t.nextToken();

        String s2=t.nextToken();

        String s3=s1.substring(s1.length()-1);

        String s4=s2.substring(0,1);

        if(s3.equalsIgnoreCase(s4))

            b=true;

        return b;

    }

}

```

## 45.Occurance Count

Write a program to read a string that contains a sentence and read a word. Check the number of occurrences of that word in the sentence.

Include a class UserMainCode with a static method **countWords** which accepts the two strings. The return type is the integer giving the count.

Note: The check is case-sensitive.

Create a Class Main which would be used to accept the two strings and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of two strings.

Output consists of count indicating the number of occurrences.

Refer sample output for formatting specifications.

**Sample Input 1:**

Hello world Java is best programming language in the world  
world

**Sample Output 1:**

2

**Sample Input 2:**

hello world

World

**Sample Output 2:**

0

Main

```
import java.util.*;  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        Scanner s=new Scanner(System.in);  
  
        String s1=s.nextLine();  
        String s2=s.nextLine();  
  
        int v=UserMainCode.countWords(s1,s2);  
  
        System.out.println(v);  
  
        s.close();  
    }  
}
```

UserMainCode

```
import java.util.StringTokenizer;  
  
public class UserMainCode {  
  
    public static int countWords(String s1,String s2){  
  
        StringTokenizer t=new StringTokenizer(s1," ");  
  
        int c=0;
```

```

while(t.hasMoreTokens())
{
    String s3=t.nextToken();
    if(s3.equals(s2))
        c++;
}
return c;
}
}

```

## 46.Regular Expressions - III

Write a program to read two strings S1 & S2, compute the number of times that S2 appears in S1.

Include a class UserMainCode with a static method **searchString** which accepts the two strings. The return type is the integer giving the count.

Note: The check is case-insensitive.

Create a Class Main which would be used to accept the two strings and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of two strings.

Output consists of count indicating the number of occurrences.

Refer sample output for formatting specifications.

### **Sample Input 1:**

Catcowcat

cat

### **Sample Output 1:**

**Sample Input 2:**

Catcowcat

catp

**Sample Output 2:**

0

Main

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.next();
        String s2=s.next();
        int v=UserMainCode.searchString(s1,s2);
        System.out.println(v);
        s.close();
    }
}
```

UserMainCode

```
public class UserMainCode {
    public static int searchString(String s1,String s2){
        int c=0;
        int t=s2.length();
        for(int i=0;i<s1.length()-t+1;i++){
            if(s2.equals(s1.substring(i,t+i))){
                c++;
            }
        }
    }
}
```

```
    return c;  
}  
  
}
```

## 47.Strings Processing

Write a program to read a string that contains comma separated fruit names and also a number N. Pick the nth fruit and return it. If the total number of elements are less than the number specified in N, then return the last element.

Include a class UserMainCode with a static method **findFruitName** which accepts the string and the number n. The return type is the string which has the fruit name.

Create a Class Main which would be used to accept the string and integer and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string and integer.

Output consists of a string.

Refer sample output for formatting specifications.

### **Sample Input 1:**

Apple,Banana,Orange

2

### **Sample Output 1:**

Banana

### **Sample Input 2:**

Apple,Banana,Orange

4

### **Sample Output 2:**

Orange

```
Main
import java.util.Scanner;

public class Main

{
    public static void main(String args[])
    {
        String str=new String();
        Scanner sc=new Scanner(System.in);
        str=sc.nextLine();
        int n=sc.nextInt();
        String k=UserMainCode.findFruitName(str, n);
        System.out.println(k);
        sc.close();
    }
}

UserMainCode
import java.util.StringTokenizer;

public class UserMainCode

{
    public static String findFruitName(String m,int n)
    {
        int i=0;
        String h=null;
        StringTokenizer st=new StringTokenizer(m,",");
        int max=st.countTokens();
        String[] ss=new String[max];
        while(st.hasMoreElements())
        {
            ss[i++]=st.nextToken();
        }
    }
}
```

```

    }

    if(n>max)

        h=ss[i-1];

    else

        h=ss[n-1];

    return h;

}
}

```

## 48.Poor Case

Write a program to read a string and convert the initial letter of each word to uppercase.

Include a class UserMainCode with a static method **changeCase** which accepts the string.

The return type is the modified string.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of a string.

Refer sample output for formatting specifications.

### **Sample Input 1:**

This is cognizant academy

### **Sample Output 1:**

This Is Cognizant Academy

Main

```

import java.util.*;

public class Main {

    public static void main(String[] args){

        Scanner s=new Scanner(System.in);

        String s1=s.nextLine();
    }
}

```

```
        System.out.println(UserMainCode.changeCase(s1));  
        s.close();  
    }  
}
```

### UserMainCode

```
import java.util.StringTokenizer;
```

```
public class UserMainCode {  
  
    public static String changeCase(String s1){  
  
        StringBuffer s5=new StringBuffer();  
        StringTokenizer t=new StringTokenizer(s1, " ");  
  
        while(t.hasMoreTokens()) {  
  
            String s2=t.nextToken();  
  
            String s3=s2.substring(0,1);  
  
            String s4=s2.substring(1, s2.length());  
  
            s5.append(s3.toUpperCase()).append(s4).append(" ");  
        }  
  
        return s5.toString();  
    }  
}
```

## 49.Length of same word

Write a program to read a string containing multiple words find the first and last words, if they are same, return the length and if not return the sum of length of the two words.

Include a class UserMainCode with a static method **compareLastWords** which accepts the

string. The return type is the length as per problem.

Create a Class Main which would be used to accept the string and call the static method

present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of a integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

This is Cognizant Academy

### **Sample Output 1:**

11

### **Sample Input 2:**

Hello World Hello

### **Sample Output 2:**

5

Main

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        // TODO Auto-generated method stub
        String s1=sc.nextLine();
        System.out.println(UserMainCode.compareLastWords(s1));
        sc.close();
    }
}
UserMainCode
import java.util.ArrayList;
```

```

import java.util.List;
import java.util.StringTokenizer;
public class UserMainCode {
    public static int compareLastWords(String s1){
        List<String> l=new ArrayList<String>();
        StringTokenizer t=new StringTokenizer(s1, " ");
        while(t.hasMoreTokens())
        {
            String s2=t.nextToken();
            l.add(s2);
        }
        String s3=l.get(0);
        String s4=l.get(l.size()-1);
        if(s3.equals(s4))
        {
            int n=s3.length();
            System.out.println(n);
        }
        else
        {
            int n1=s3.length();
            int n2=s4.length();
            int n=n1+n2;
        }
        Return n;
    }
}

```

## 50.Perfect Number

Write a program to that takes a positive integer and returns true if the number is perfect

number.

A positive integer is called a perfect number if the sum of all its factors (excluding the number itself, i.e., proper divisor) is equal to its value.

For example, the number 6 is perfect because its proper divisors are 1, 2, and 3, and  $6=1+2+3$ ; but the number 10 is not perfect because its proper divisors are 1, 2, and 5, and  $1+2+5 \neq 10$

Include a class UserMainCode with a static method **getPerfection** which accepts the number. The return type is boolean (true / false).

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a integer.

Output consists of TRUE / FALSE.

Refer sample output for formatting specifications.

### **Sample Input 1:**

28

### **Sample Output 1:**

TRUE

```
import java.util.*;  
  
public class Main {  
  
    public static void main(String[] args){  
  
        Scanner s=new Scanner(System.in);  
  
        int n=s.nextInt();  
  
        boolean j=(UserMainCode.getPerfection(n));  
  
        if(j==true)
```

```

System.out.println("TRUE");
else
System.out.println("FALSE");
}

}

public class UserMainCode {
public static boolean getPerfection(int n){
boolean b=false;
int sum=0;
for(int i=1;i<n;i++){
int r=n%i;
if(r==0)
sum=sum+i;
}
b=(sum==n);
return b;
}
}

```

## 51.Find Digits

For a given double number with atleast one decimal value, Write a program to compute the number of digits before and after the decimal point in the following format –  
noOfDigitsBeforeDecimal:noOfDigitsAfterDecimal.

Note: Ignore zeroes at the end of the decimal (Except if zero is the only digit after decimal).

Refer Example 2 and 3)

Include a class UserMainCode with a static method **findNoDigits** which accepts the decimal

value. The return type is string.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of a double.

Output consists of string.

Refer sample output for formatting specifications.

**Sample Input 1:**

843.21

**Sample Output 1:**

3:2

**Sample Input 2:**

20.130

**Sample Output 2:**

2:2

**Sample Input 3:**

20.130

```
import java.util.*;  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        Scanner s=new Scanner(System.in);  
  
        double d=s.nextDouble();  
  
        System.out.println(UserMainCode.findNoDigits(d));  
  
    }  
}
```

```
import java.util.StringTokenizer;

public class UserMainCode {
    public static String findNoDigits(double d) {
        int n1=0,n2=0;
        String s=String.valueOf(d);
        StringTokenizer t=new StringTokenizer(s,".");
        String s1=t.nextToken();
        String s2=t.nextToken();
        n1=s1.length();
        n2=s2.length();
        if(s1.charAt(0)=='0')
            n1=s1.length()-1;
        if(n2!=1)
            if(s2.charAt(s2.length()-1)=='0')
                n2=s2.length()-1;
        String s3=String.valueOf(n1)+":"+String.valueOf(n2);
        return s3;
    }
}
```

## 52.Employees & Designations

A Company wants to obtain employees of a particular designation. You have been assigned

as the programmer to build this package. You would like to showcase your skills by creating a quick prototype. The prototype consists of the following steps:

Read Employee details from the User. The details would include name and designation in the given order. The datatype for name and designation is string.

Build a hashmap which contains the name as key and designation as value.

You decide to write a function **obtainDesignation** which takes the hashmap and designation as input and returns a string List of employee names who belong to that designation as output. Include this function in class UserMainCode. Display employee name's in ascending order.

Create a Class Main which would be used to read employee details in step 1 and build the hashmap. Call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of employee details. The first number indicates the size of the employees. The next two values indicate the employee name employee designation. The last string would be the designation to be searched.

Output consists of a array values containing employee names.

Refer sample output for formatting specifications.

### **Sample Input 1:**

4

Manish

MGR

Babu

CLK

Rohit

MGR

Viru

PGR

MGR

### **Sample Output 1:**

Manish

Rohit

```
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.Scanner;
public class Main {
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        int k1=Integer.parseInt(sc.nextLine());
        LinkedHashMap<String, String> hm=new LinkedHashMap<String, String>();
        for(int i=0;i<k1;i++)
        {
            String k=sc.nextLine();
            String s=sc.nextLine();
            hm.put(k,s);
        }
        String n=sc.nextLine();
        LinkedHashMap<String, String> hm1=new LinkedHashMap<String, String>();
        hm1=UserMainCode.obtainDesignation(hm,n);
        Iterator<String> it=hm1.keySet().iterator();
        while(it.hasNext())
        {
```

```
String s2=it.next();
System.out.println(s2);
}
}
}
```

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.Map;
import java.util.Scanner;
public class UserMainCode
{
    public static LinkedHashMap<String,String>
    obtainDesignation(LinkedHashMap<String,String> h1,String n)
    {
        int k=0;
        LinkedHashMap<String,String> hm1=new LinkedHashMap<String,String>();
        Iterator<String>it=h1.keySet().iterator();
        while(it.hasNext())
        {
            String s2=it.next();
            String s3=h1.get(s2);
            if(s3.equals(n))
                hm1.put(s2,s3);
        }
        return hm1;
    }
}
```

}

### 53. Grade Calculator

A School wants to give assign grades to its students based on their marks. You have been assigned as the programmer to automate this process. You would like to showcase your skills by creating a quick prototype. The prototype consists of the following steps:

Read student details from the User. The details would include name, mark in the given order. The datatype for name is string, mark is float.

You decide to build a hashmap. The hashmap contains name as key and mark as value.

**BUSINESS RULE:**

1. If Mark is less than 60, then grade is FAIL.

2. If Mark is greater than or equal to 60, then grade is PASS.

Note: FAIL/PASS should be in uppercase.

Store the result in a new Hashmap with name as Key and grade as value.

4. You decide to write a function **calculateGrade** which takes the above hashmap as input and returns the hashmap as output. Include this function in class UserMainCode.

Create a Class Main which would be used to read student details in step 1 and build the hashmap. Call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of student details. The first number indicates the size of the students. The next two values indicate the name, mark.

Output consists of a name and corresponding grade for each student.

Refer sample output for formatting specifications.

**Sample Input 1:**

3

Avi

76.36

Sunil

68.42

Raja

36.25

### **Sample Output 1:**

Avi

PASS

Sunil

PASS

Raja

FAIL

```
import java.util.Iterator;  
  
import java.util.LinkedHashMap;  
  
import java.util.Scanner;  
  
public class Main {  
  
    public static void main(String[] args)  
    {  
  
        Scanner sc=new Scanner(System.in);  
  
        int k1=Integer.parseInt(sc.nextLine());  
  
        LinkedHashMap<String, String> hm=new LinkedHashMap<String, String>();  
  
        for(int i=0;i<k1;i++)  
        {
```

```
String k=sc.nextLine();

String s=sc.nextLine();

hm.put(k,s);

}

String n=sc.nextLine();

LinkedHashMap<String,String> hm1=new LinkedHashMap<String,String>();

hm1=UserMainCode.obtainDesignation(hm,n);

Iterator<String> it=hm1.keySet().iterator();

while(it.hasNext())

{

String s2=it.next();

System.out.println(s2);

}

}

}

import java.util.HashMap;

import java.util.Iterator;

import java.util.LinkedHashMap;

import java.util.Map;

import java.util.Scanner;

public class UserMainCode

{

public static LinkedHashMap<String,String>

obtainDesignation(LinkedHashMap<String,String> h1,String n)

{

int k=0;

LinkedHashMap<String,String> hm1=new LinkedHashMap<String,String>();

Iterator<String>it=h1.keySet().iterator();
```

```
while(it.hasNext())
{
    String s2=it.next();
    String s3=h1.get(s2);
    if(s3.equals(n))
        hm1.put(s2,s3);
}
return hm1;
}}
```

## 54. DOB - Validation

Write a program to validate the Date of Birth given as input in String format (MM/dd/yyyy) as per the validation rules given below. Return true for valid dates else return false.

1. Value should not be null
2. month should be between 1-12, date should be between 1-31 and year should be a four digit number.

Include a class UserMainCode with a static method **ValidateDOB** which accepts the string. The return type is TRUE / FALSE.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of TRUE / FALSE.

Refer sample output for formatting specifications.

### **Sample Input 1:**

12/23/1985

**Sample Output 1:**

TRUE

**Sample Input 2:**

31/12/1985

**Sample Output 2:**

FALSE

```
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;
public class Main {
    public static void main(String[] args)
    {
        String str=new String();
        Scanner sc=new Scanner(System.in);
        str=sc.nextLine();
        Boolean b=UserMainCode.ValidateDOB(str);
        if(b=="true")
            System.out.println("TRUE");
        if(b=="false")
            System.out.println("FALSE");
    }
}
```

```
import java.text.SimpleDateFormat;
import java.util.Date;
public class UserMainCode {
    public static Boolean ValidateDOB(String str){
```

```

Boolean b="false";

SimpleDateFormat sdf=new SimpleDateFormat("MM/dd/yyyy");

sdf.setLenient(false);

try

{

Date d1=sdf.parse(str);

return b="true";

}

catch(Exception e)

{

return b="false";

}

}

}

```

## **55.Experience Validator**

Write a program to validate the experience of an employee.

An employee who has recently joined the organization provides his year of passing and total number of years of experience in String format. Write code to validate his experience against the current date.

- 1) Input consists of two String first represent the year of passed out and the second string represent the year of experience.
- 2) create a function with name **validateExp** which accepts two string as input and boolean as output.
- 3) The difference between current year and year of pass should be more than or equal to Experience

Return true if all condition are true.

Note: Consider 2015 as the current year.

Include a class UserMainCode with the static function validateExp

Create a Class Main which would be used to accept the boolean and call the static method present in UserMainCode.

### **Input and Output Formate:**

Input consists of two Strings.

output will display true if the given data are correct.

### **Sample Input:**

2001

5

### **Sample Output:**

TRUE

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Scanner;
public class Main {
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        String s=sc.nextLine();
        String s1=sc.nextLine();
        System.out.println(UserMainCode.validateExp(s,s1));
    }
}
```

```
import java.util.Calendar;
import java.util.Date;
public class UserMainCode {
    public static boolean validateExp(String s,String s1)
```

```
{  
    int y1=Integer.parseInt(s);  
    Date d=new Date();  
    Calendar c=Calendar.getInstance();  
    int y2=c.get(Calendar.YEAR);  
    int y=Math.abs(y1-y2);  
    int e=Integer.parseInt(s1);  
    if(y>=e)  
        return true;  
    else  
        return false;  
}
```

## 56. ArrayList to String Array

Write a program that performs the following actions:

Read n strings as input.

Create an arraylist to store the above n strings in this arraylist.

Write a function `convertToStringArray` which accepts the arraylist as input.

The function should sort the elements (strings) present in the arraylist and convert them into a string array.

Return the array.

Include a class `UserMainCode` with the static method **convertToStringArray** which accepts

an arraylist and returns an array.

Create a Class `Main` which would be used to read n strings and call the static method present in `UserMainCode`.

### Input and Output Format:

Input consists of n+1 integers. The first integer denotes the size of the arraylist, the next n

strings are values to the arraylist.

Output consists of an arrayas per step 4.

Refer sample output for formatting specifications.

### **Sample Input 1:**

4

a

d

c

b

### **Sample Output 1:**

a

b

c

d

```
import java.util.*;

public class Main

{
    public static void main(String[] args)

    {
        Scanner s=new Scanner(System.in);

        ArrayList<String> l=new ArrayList<String>();

        int n=s.nextInt();

        for(int i=0;i<n;i++)

        {
            l.add(s.next());
        }
    }
}
```

```

        }

        String a[]={};String[n];
        a=UserMainCode.convertToStringArray(l);
        for(int j=0;j<n;j++)
        {
            System.out.println(a[j]);
        }
    }

import java.util.ArrayList;
import java.util.Collections;

class UserMainCode
{
    public static String[] convertToStringArray(ArrayList<String> l)
    {
        Collections.sort(l);
        String [] a = l.toArray(new String[l.size()]);
        return a;
    }
}

```

## 57.State ID generator

Write a program to generate the state ID.

- 1)Read n Strings as input(as State Name).
- 2)Create a String Array to Store the above Input.
- 3)Write a function **getStatetId** which accepts String Array as input.

4)Create a `HashMap<String, String>` which stores state name as key and state Id as Value.

5)The function `getStatId` returns the `HashMap` to the Main Class.

Include UserMainCode Class With static method `getStatId` which accepts String array and return a hashmap.

Create a Class Main which would be used to read n strings and call the static method present in UserMainCode.

### **Input and Output Format:**

Input Consists of an integer n denotes the size of the string array.

Output consists of an `HashMap` displayed in the string array order.

### **Sample Input 1:**

3

Kerala

Gujarat

Goa

### **Sample Output 1:**

KER:Kerala

GUJ:Gujarat

GOA:Goa

#### **Main Class**

```
import java.util.*;  
  
public class Main  
{  
    public static void main(String[] args)  
    {  
        Scanner s=new Scanner(System.in);  
        int n=s.nextInt();
```

```
String[] s1=new String[n];

for(int i=0;i<n;i++)
{
    s1[i]=s.next();
}

HashMap<String, String> hm = new HashMap<String, String>();

hm = UserMainCode.putvalues(s1);

for(Map.Entry<String, String> ans: hm.entrySet())
{
    System.out.println(ans.getKey()+ ":" +ans.getValue());
}
}
```

User main code

```
import java.util.ArrayList;

import java.util.HashMap;

import java.util.Map;

public class UserMainCode{

    public static HashMap<String, String> putvalues(String[] s1)
    {
        HashMap<String, String> hm = new HashMap<String, String>();

        ArrayList<String> lst1 = new ArrayList<String>();

        ArrayList<String> lst2 = new ArrayList<String>();
```

```

for(String s : s1)

    lst1.add(s.toUpperCase().substring(0,3));

for(String s : s1)

    lst2.add(s);

for(int i=0;i<s1.length;i++)

{

    hm.put(lst1.get(i),lst2.get(i));

}

return hm;

}

```

## 58. ArrayList to String Array

Write a program that performs the following actions:

- 1.Read m strings as input (fruit names).
- 2.Create an arraylist to store the above m strings in this arraylist.
- 3.Read n strings as input (fruit names).
- 4.Create an arraylist to store the above n strings in this arraylist.
- 5.Write a function fruitSelector which accepts the arraylists as input.
- 6.Remove all fruits whose name ends with 'a' or 'e' from first arrayList and remove all fruits whose name begins with 'm' or 'a' from second arrayList then combine the two lists and return the final output as a String array.
- 7.If the array is empty the program will print as "No fruit found"

Include a class UserMainCode with the static method **fruitSelector** which accepts the two arraylists and returns an array.

Create a Class Main which would be used to read n strings and call the static method

present in UserMainCode.

**Input and Output Format:**

Input consists of an integer (m) denoting the size of first arraylist. The next m elements would be the values of the first arraylist. The next input would be n denoting the size of the second arraylist. The next n elements would be the values of the second arraylist.

Output consists of an array as per step 6. Refer sample output for formatting specifications.

**Sample Input 1:**

3

Apple

Cherry

Grapes

4

Orange

Mango

Melon

Apple

**Sample Output 1:**

Cherry

Grapes

Orange

**USERMAINCODE:**

```
import java.util.ArrayList;  
import java.util.*;  
  
public class UserMainCode {
```

```
public static String[] fruitSelector(ArrayList<String> a1,ArrayList<String> a2)

{
    ArrayList<String> a3=new ArrayList<String>();

    for(int i=0;i<a1.size();i++)
    {
        String s1=a1.get(i);

        if(s1.charAt(s1.length()-1)!='a'&&s1.charAt(s1.length()-1)!='e'&&s1.charAt(s1.length()-1)!='A'&&s1.charAt(s1.length()-1)!='E')

        {
            a3.add(s1);
        }
    }

    ArrayList<String> a4=new ArrayList<String>();

    for(int j=0;j<a2.size();j++)
    {
        String s2=a2.get(j);

        if(s2.charAt(0)!='m'&&s2.charAt(0)!='a'&&s2.charAt(0)!='M'&&s2.charAt(0)!='A')

        {
            a4.add(s2);
        }
    }

    a3.addAll(a4);

    Collections.sort(a3);

    String st[]=new String[a3.size()];

    for(int k=0;k<a3.size();k++)
    {
        st[k]=a3.get(k);
    }

    return st;
}
```

```
}
```

```
}
```

MAIN:

```
import java.util.*;  
  
import java.util.ArrayList;  
  
public class Main {  
  
    public static void main(String [] args)  
    {  
  
        Scanner s=new Scanner(System.in);  
  
        int m=s.nextInt();  
  
        ArrayList<String> aa1=new ArrayList<String>();  
  
        for(int i=0;i<m;i++)  
        {  
  
            aa1.add(s.next());  
  
        }  
  
        int n=s.nextInt();  
  
        ArrayList<String> aa2=new ArrayList<String>();  
  
        for(int j=0;j<n;j++)  
        {  
  
            aa2.add(s.next());  
  
        }  
  
        int k;  
  
        String st[]=UserMainCode.fruitSelector(aa1,aa2);  
  
        for( k=0;k<st.length;k++)  
        {  
  
            System.out.println(st[k]);  
  
        }  
  
        if(st.length==0)
```

```
System.out.println("No Fruit Found");

s.close();

}

}
```

## 59)Elements in ArrayList

Use Collection Methods.

Write a program that takes two ArrayLists as input and finds out all elements present either in A or B, but not in both.

Include a class UserMainCode with the static method arrayListSubtractor which accepts the two arraylists and returns an array.

Create a Class Main which would be used to read the inputs and call the static method present in UserMainCode.

### Input and Output Format:

Input consists of an integer (m) denoting the size of first arraylist. The next m elements would be the values of the first arraylist. The next input would be n denoting the size of the second arraylist. The next n elements would be the values of the second arraylist.

Output consists of an array. The elements in the output array need to be printed in sorted order.

Refer sample output for formatting specifications.

### Sample Input 1:

```
4
1
8
3
5
2
3
5
```

### Sample Output 1:

```
1
8
```

### Sample Input 2:

```
4
9
1
3
5
4
1
3
5
6
```

### Sample Output 2:

```
6
```

9

## MAIN:

```
import java.util.*;
public class Main
{
public static void main(String[] args)
{
int n,m;
Scanner sin = new Scanner(System.in);
n = sin.nextInt();
ArrayList<Integer> a1 = new ArrayList<Integer>(n);
for(int i=0;i<n;i++)
{
int k = sin.nextInt();
a1.add(k);
}
m = sin.nextInt();
ArrayList<Integer> a2 = new ArrayList<Integer>(m);
for(int i=0;i<m;i++)
{
int k = sin.nextInt();
a2.add(k);
}
int[] result = UserMainCode.arrayListSubtractor(a1,a2);
Arrays.sort(result);
for(int i=0;i<result.length;i++)
System.out.println(result[i]);
}
}
```

## USERMAINCODE:

```
import java.util.ArrayList;
public class UserMainCode
{
public static int[] arrayListSubtractor(ArrayList<Integer>
arrlist1,ArrayList<Integer>
arrlist2)
{
int count=0,key;
int max = arrlist1.size();
if(arrlist1.size() < arrlist2.size())
max = arrlist2.size();
ArrayList<Integer> temp = new ArrayList<Integer>(max);
for(int i=0;i<arrlist1.size();i++)
{
key = (int)arrlist1.get(i);
if(arrlist2.indexOf(key) == -1)
{
++count;
temp.add(key);
}
}
for(int i=0;i<arrlist2.size();i++)
{
key = (int)arrlist2.get(i);
if(arrlist1.indexOf(key) == -1)
```

```

{
if(!temp.contains(key))
{
++count;
temp.add(key);
}
}
}

int[] result = new int[count];
for(int i=0;i<count;i++)
result[i] = (int)temp.get(i);
return result;
}
}

```

## 60.Price Calculator - II

Write a small price calculator application with the below mentioned flow:

1. Read a value n indicating the total count of devices. This would be followed by the name and price of the device. The datatype for name would be String and price would be float.
2. Build a hashmap containing the peripheral devices with name as key and price as value.
3. Read a value m indicating the number of devices for which the price has to be calculated. This would be followed by device names.
4. For each devices mentioned in the array calcuate the total price.
5. You decide to write a function costEstimator which takes the above hashmap and array as input and returns the total price (float) as output with two decimal points. Include this function in class UserMainCode.

Create a Class Main which would be used to read details in step 1 and build the hashmap. Call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of device details. The first number indicates the size of the devices. The next two values indicate the name,price.

This would be followed by m indicating the size of the device array. The next m values would be the device names.

Output consists of the total price in float.

Refer sample output for formatting specifications.

### **Sample Input 1:**

```

3
Monitor
1200.36
Mouse
100.42
Speakers
500.25
2
Speakers
Mouse

```

### **Sample Output 1:**

600.67

## MAIN:

```
import java.util.HashMap;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner S=new Scanner(System.in);
        int n=S.nextInt();

        HashMap<String, Float> m1=new HashMap<String, Float>();
        for(int i=0;i<n;i++)
        {
            String name=S.next();
            float price=S.nextFloat();
            m1.put(name,price);
        }
        int m=S.nextInt();
        String s[]=new String[m];
        for(int j=0;j<m;j++)
        {
            s[j]=S.next();
        }
        System.out.println(UserMainCode.getTheTotalCostOfPeripherals
(m1,s));
    }
}
```

## USERMAINCODE:

```
import java.util.HashMap;
import java.util.Iterator;

public class UserMainCode {
    public static float getTheTotalCostOfPeripherals(HashMap<String,Float> m1,
String[] s) {
        float f=(float) 0;
        Iterator<String> i=m1.keySet().iterator();
        while(i.hasNext()){
            String s1=i.next();
            float f1=m1.get(s1);
            for(int j=0;j<s.length;j++)
            if(s[j].equals(s1))
                f+=f1;
        }
        return f;
    }
}
```

## 61.String Processing - ZigZag

Write a program to read a string containing date in DD-MM-YYYY format. find the number of days in the given month.

Note - In leap year February has got 29 days.

Include a class UserMainCode with a static method **getLastDayOfMonth** which accepts the

string. The return type is the integer having number of days.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

12-06-2012

### **Sample Output 1:**

30

### **Sample Input 2:**

10-02-2012

### **Sample Output 2:**

29

## **MAIN:**

```

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.*;
public class Main {
    public static void main(String[] args) throws IOException,
ParseException {
        Scanner S=new Scanner(System.in);
        String s1=S.next();
        UserMainCode.getLastDayOfMonth(s1);
    }
}

```

## **USERMAINCODE:**

```

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

public class UserMainCode {
    public static void getLastDayOfMonth(String s1) throws ParseException{
        SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");
        Calendar cal=Calendar.getInstance();
        Date d1=sdf.parse(s1);
        cal.setTime(d1);
        int n=cal.getActualMaximum(Calendar.DAY_OF_MONTH);
        System.out.println(n);
    }
}

```

## **62.Leap Year**

Write a program to read a string containing date in DD/MM/YYYY format and check if its a leap year. If so, return true else return false.

Include a class UserMainCode with a static method **isLeapYear** which accepts the string. The

return type is the boolean indicating TRUE / FALSE.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of TRUE / FALSE.

Refer sample output for formatting specifications.

### **Sample Input 1:**

23/02/2012

### **Sample Output 1:**

TRUE

### **Sample Input 2:**

12/12/2011

### **Sample Output 2:**

FALSE

## **MAIN:**

```
import java.io.IOException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.*;
public class Main {
    public static void main(String[] args) throws IOException, ParseException {
        Scanner S=new Scanner(System.in);
        String s1=S.next();
        UserMainCode.isLeapyear(s1);
    }
}
```

## **USERMAINCODE:**

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.GregorianCalendar;
import java.util.StringTokenizer;

public class UserMainCode {
    public static void isLeapyear(String s1) throws ParseException{
        SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
        GregorianCalendar g=new GregorianCalendar();
        StringTokenizer t=new StringTokenizer(s1,"/");
        String s2=t.nextToken();
        String s3=t.nextToken();
        String s4=t.nextToken();
        int n1=Integer.parseInt(s4);
        Date d1=sdf.parse(s1);
        boolean b=g.isLeapYear(n1);
        System.out.println(b);
    }
}
```

## **63) Largest Chunk**

Write a program to read a string and return the length of the largest "chunk" in the string.

A chunk is a repetition of same character 2 or more number of times. If the given string

does not contain any repeated chunk of characters return -1.

Include a class UserMainCode with a static method **getLargestSpan** which accepts the string.

The return type is the integer.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

This place is soooo good

### **Sample Output 1:**

4

### **MAIN:**

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner S=new Scanner(System.in);
        String s1=S.nextLine();
        System.out.println(UserMainCode.getLargestSpan(s1));
    }
}
```

### **USERMAINCODE :**

```
import java.util.StringTokenizer;

public class UserMainCode {
    public static int getLargestSpan(String s1) {
        int max=0;
        StringTokenizer t=new StringTokenizer(s1, " ");
        while(t.hasMoreTokens()){
            String s2=t.nextToken();
            int n=0;
            for(int i=0;i<s2.length()-1;i++)
                if(s2.charAt(i)==s2.charAt(i+1))
                    n++;
            if(n>max)
                max=n;
        }
        return (max+1);
    }
}
```

## **64) Largest Span**

Write a program to read a integer array, find the largest span in the array.

Span is the count of all the elements between two repeating elements including the repeated elements.

Include a class UserMainCode with a static method **getLargestSpan** which accepts the integer array. The return type is integer.

Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of an integer n which is the number of elements followed by n integer values.

Output consists of integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

```
6  
4  
2  
1  
4  
5  
7
```

### **Sample Output 1:**

```
4
```

### **MAIN:**

```
import java.util.Scanner;  
public class Main {  
    public static void main(String[] args)  
    {  
        Scanner sc=new Scanner(System.in);  
        int n=sc.nextInt();  
        int []a=new int[n];  
        for(int i=0;i<n;i++)  
        {  
            a[i]=sc.nextInt();  
        }  
        System.out.print(UserMainCode.getLargestSpan(a,n));  
    }}
```

### **USERMAINCODE:**

```
public class UserMainCode {  
    public static int getLargestSpan(int[] x,int n)  
    {  
        int gap=0,max=0;  
        for(int i=0;i<n;i++)  
        {  
            for(int j=i+1;j<n;j++)  
            {  
                if(x[i]==x[j])  
                {  
                    gap=j;  
                }  
            }  
            if(gap-i>max)  
                max=gap-i;  
        }  
        return max+1;  
    }  
}
```

### **65 )Even Sum & Duplicate Elements**

Write a program to read a integer array, Remove the duplicate elements and display sum of even numbers in the output. If input array contain only odd number then return -1.

Include a class UserMainCode with a static method **sumElements** which accepts the integer

array. The return type is integer.

Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of an integer n which is the number of elements followed by n integer values.

Output consists of integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

```
7  
2  
3  
54  
1  
6  
7  
7
```

### **Sample Output 1:**

```
62
```

### **Sample Input 2:**

```
6  
3  
7  
9  
13  
17  
21
```

### **Sample Output 2:**

```
-1
```

### **MAIN:**

```
import java.util.HashMap;  
import java.util.LinkedHashMap;  
import java.util.LinkedHashSet;  
import java.util.Scanner;  
public class Main  
{  
    public static void main(String args[])  
    {  
        Scanner sc=new Scanner(System.in);  
        int n=sc.nextInt();  
        int a[]={};  
        for(int i=0;i<n;i++)  
        {  
            a[i]=sc.nextInt();  
        }  
        System.out.println(UserMainCode.sumElements(a));  
    }  
}
```

**USERMAINCODE:**

```
import java.util.Iterator;
import java.util.LinkedHashSet;
public class UserMainCode {
public static int sumElements(int a[])
{
LinkedHashSet<Integer> h1=new LinkedHashSet<Integer>();
int s=0;
for(int i=0;i<a.length;i++)
{
h1.add(a[i]);
}
Iterator<Integer> it=h1.iterator();
while(it.hasNext())
{
int k=it.next();
if(k%2==0)
{
s=s+k;
}
}
if(s>0)
return s;
else
return -1;
}
}
```

## 66. Regular Expression - III

Given a string (s) apply the following rules.

- I)At least 8 characters must be present
- II)At least one capital letter must be present
- III)At least one small letter must be present
- Iv)At least one special symbol must be present
- V)At least one numeric value must be present

If the condition is satisfied then print valid else print invalid.

Include a class UserMainCode with a static method passwordValidation which accepts the string. The return type is the string.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

Input and Output Format:

Input consists of a string.

Output consists of string (valid / invalid) .

Refer sample output for formatting specifications.

Sample Input 1:

Technology\$1213

Sample Output 1:

valid

**Main:**

```
import java.util.Scanner;
```

```

public class Main {
    public static void main(String[] args)
    {
        Scanner s=new Scanner(System.in);
        String a=s.next();
        System.out.println(UserMainCode.passwordValidation(a));
        s.close();
    }
}

UserMainCode:
public class UserMainCode
{
    public static String passwordValidation(String a)
    {
        String k;
        if(a.matches(".*[0-
9]{1,}.*")&&a.matches(".*[@#$]{1,}.*")&&a.length()>=8&&a.matches(".*[A-
Z]{1,}.*")&&a.matches(".*[a-z].*"))
        {
            k="validinput";
        }
        else
        {
            k="Invalidinput";
        }
        return k;
    }
}

```

## 67. Integer Factorial

Give an array of integer as input, store the numbers and their factorials in an hashmap and print the same.

Include a class UserMainCode with a static method getFactorial which accepts the integer array. The return type is the hashmap which is printed key:value.

Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.

Input and Output Format:

Input consists of a number denoting the size of the array and followed by the elements.

Output consists of a hashmap printed in the output format .

Refer sample output for formatting specifications.

Sample Input1:

```

4
2
3
5
4

```

Sample Output1:

2:2

3:6

5:120

4:24

**Main:**

```
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.Scanner;
```

```
public class Main
{
    public static void main(String[] args)
    {
        Scanner s=new Scanner(System.in);
        int a=Integer.parseInt(s.nextLine());
        int[] k=new int[a];
        for(int i=0;i<a;i++)
        {
            k[i]=s.nextInt();
        }
        LinkedHashMap<Integer,Integer> hm=new LinkedHashMap<Integer,Integer>();
        hm=UserMainCode.getFactorial(k);
        Iterator<Integer> it=hm.keySet().iterator();
        for(int i=0;i<a;i++)
        {
            int n=it.next();
            int fac=hm.get(n);
            System.out.println(n+":"+fac);
            s.close();
        }
    }
}
```

**UserMainCode;**

```
import java.util.LinkedHashMap;
```

```
public class UserMainCode
{
    public static LinkedHashMap<Integer,Integer> getFactorial(int[] k)
    {
        LinkedHashMap<Integer,Integer> hm1=new LinkedHashMap<Integer,Integer>();
        for(int i=0;i<k.length;i++)
        {
            int u=1;
            for(int j=1;j<=k[i];j++)
            {
                u=u*j;
            }
        }
    }
}
```

```

    }
    hm1.put(k[i],u);
}
return hm1;
}

}

```

### **68. String processing – Long + Short + Long**

Obtain two strings S1,S2 from user as input. Your program should form a string of “long+short+long”, with the shorter string inside of the longer String.

Include a class UserMainCode with a static method getCombo which accepts two string variables. The return type is the string.

Create a Class Main which would be used to accept two Input strings and call the static method present in UserMainCode.

Input and Output Format:

Input consists of two strings with maximum size of 100 characters.

Output consists of an string.

Refer sample output for formatting specifications.

Sample Input 1:

Hello

Hi

Sample Output 1:

HelloHiHello

**Main;**

```
import java.util.Scanner;
```

```

public class Main
{
public static void main(String[] args)
{
Scanner s=new Scanner(System.in);
String s1=s.next();
String s2=s.next();
System.out.println(UserMainCode.getCombo(s1,s2));
s.close();
}
}

```

**UserMainCode;**

```

public class UserMainCode
{
public static String getCombo(String s1,String s2)
{
StringBuffer sb=new StringBuffer();
int p=s1.length();
int q=s2.length();
if(p>q)

```

```

{
sb.append(s1).append(s2).append(s1);
}
else
{
sb.append(s2).append(s1).append(s2);
}
return sb.toString();
}

}

```

### **69. Age for Voting**

Given a date of birth (dd/MM/yyyy) of a person in string, compute his age as of 01/01/2015.  
If his age is greater than 18, then println eligible else println not-eligible.

Include a class UserMainCode with a static method getAge which accepts the string value.  
The return type is the string.

Create a Class Main which would be used to accept the two string values and call the static  
method present in UserMainCode.

Input and Output Format:

Input consists of two string.

Output consists of a string.

Refer sample output for formatting specifications.

Sample Input 1:

16/11/1991

Sample Output 1:

eligible

**Main:**

```
import java.util.Scanner;
```

```

public class Main {
public static void main(String[] args)
{
Scanner s=new Scanner(System.in);
String a=s.nextLine();
System.out.println(UserMainCode.getAge(a));
s.close();
}
}

```

**UserMainCode:**

```
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;
```

```

public class UserMainCode
{
public static String getAge(String n)
{

```

```

Scanner s=new Scanner(System.in);
int year=0;
String s1=s.next();
SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
try
{
Date d=sdf.parse(n);
Date d1=sdf.parse(s1);
int y=d.getYear();
int y1=d1.getYear();
int m=d.getMonth();
int m1=d1.getMonth();
int day=d.getDate();
int day1=d1.getDate();
year=y1-y;
if(m>m1)
year--;
else if(m==m1)
{if(day<day1)
year--;
}
}
catch(Exception e)
{
e.printStackTrace();
}
if(year>18)
return "eligible";
else
return "not-eligible";
}
}

```

## 1. Unique Even Sum

Write a program to read an array, eliminate duplicate elements and calculate the sum of even numbers (values) present in the array.

Include a class UserMainCode with a static method addUniqueEven which accepts a single integer array. The return type

(integer) should be the sum of the even numbers. In case there is no even number it should return -1.

Create a Class Main which would be used to accept Input array and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of  $n+1$  integers. The first integer corresponds to  $n$ , the number of elements in the array. The next ' $n$ ' integers correspond to the elements in the array.

In case there is no even integer in the input array, print no even numbers as output. Else print the sum.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20.

Sample Input 1:

```
4  
2  
5  
1  
4
```

Sample Output 1:

```
6
```

Sample Input 2:

```
3  
1  
1  
1
```

Sample Output 2:

no even numbers

**Main:**

```
import java.util.Scanner;
```

```
public class Main  
{  
    public static void main(String[] args)  
    {  
        Scanner s=new Scanner(System.in);  
        int n=s.nextInt();  
        int[] a=new int [n];  
        for(int i=0;i<n;i++)  
        {  
            a[i]=s.nextInt();  
        }  
        System.out.println(UserMainCode.addUniqueEven(a));  
        s.close();  
    }  
}
```

```
import java.util.Iterator;  
import java.util.LinkedHashSet;
```

```
public class UserMainCode  
{  
    public static int addUniqueEven(int[] a)
```

```

{
int sum=0;
LinkedHashSet<Integer> hm=new LinkedHashSet<Integer>();
for(int i=0;i<a.length;i++)
{
hm.add(a[i]);
}
Iterator<Integer> im=hm.iterator();
while(im.hasNext())
{
int b=im.next();
if(b%2==0)
sum=sum+b;
}
if(sum>0)
{
return sum;
}
else
    return -1;
}
}

```

## 2. Palindrome & Vowels

Write a program to check if a given string is palindrome and contains at least two different vowels.

Include a class UserMainCode with a static method checkPalindrome which accepts a string.  
The return type (integer)

should be 1 if the above condition is satisfied, otherwise return -1.

Create a Class Main which would be used to accept Input string and call the static method present in UserMainCode.

Note – Case Insensitive while considering vowel, i.e a & A are same vowel, But Case sensitive while considering

palindrome i.e abc CbA are not palindromes.

Input and Output Format:

Input consists of a string with maximum size of 100 characters.

Output consists of a single Integer.

Refer sample output for formatting specifications.

Sample Input 1:

abceecba

Sample Output 1:

valid

Sample Input 2:

abcd

Sample Output 2:

invalid

**Main;**

import java.util.Scanner;

```

public class Main
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        String s=sc.nextLine();
        System.out.println(UserMainCode.checkPalindrome(s));
        sc.close();
    }
}

UserMainCode;
import java.util.Iterator;
import java.util.LinkedHashSet;

public class UserMainCode
{
    public static int checkPalindrome(String s)
    {
        StringBuffer sb=new StringBuffer(s);
        int k=0;
        LinkedHashSet<Character>l1=new LinkedHashSet<Character>();
        String s2=sb.reverse().toString();
        if(s2.equals(s))
        {
            String s3=s2.toLowerCase();
            for(int i=0;i<s3.length();i++)
            {
                l1.add(s3.charAt(i));
            }
            Iterator<Character> it=l1.iterator();
            while(it.hasNext())
            {
                char a=it.next();
                if(a=='a'||a=='e'||a=='i'||a=='o'||a=='u')
                k++;
            }
        }
        if(k>=2)
        return 1;
        else
        return -1;
    }
}

```

### 3. Strings – Unique & Existing Characters

Obtain two strings from user as input. Your program should modify the first string such that all the characters are replaced

by plus sign (+) except the characters which are present in the second string.

That is, if one or more characters of first string appear in second string, they will not be replaced by +.

Return the modified string as output. Note - ignore case.

Include a class UserMainCode with a static method replacePlus which accepts two string variables. The return type is the modified string.

Create a Class Main which would be used to accept two Input strings and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of two strings with maximum size of 100 characters.

Output consists of a single string.

Refer sample output for formatting specifications.

**Sample Input 1:**

abcxyz

axdef

**Sample Output 1:**

a++ x++

**Sample Input 2:**

ABCDEF

feCBAd

**Sample Output 2:**

ABCDEF

**Main;**

import java.util.Scanner;

```
public class Main
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        String s=sc.next();
        String s1=sc.next();
        System.out.println(UserMainCode.replacePlus(s,s1));
        sc.close();
    }
}
```

**UserMainCode;**

```
public class UserMainCode
{
    public static String replacePlus(String s,String s1)
    {
        String s2=s.toLowerCase();
        String s3=s1.toLowerCase();
```

```

StringBuffer sb=new StringBuffer();
for(int i=0;i<s.length();i++)
{
char c=s2.charAt(i);
if(s3.indexOf(c)==-1)
sb.append("+");
else
sb.append(s.charAt(i));
} return sb.toString();
}
}
}
}

```

#### **4. Longest Word**

Write a Program which finds the longest word from a sentence. Your program should read a sentence as input from user and  
return the longest word. In case there are two words of maximum length return the word which comes first in the sentence.

Include a class UserMainCode with a static method getLargestWord which accepts a string  
The return type is the longest  
word of type string.

Create a Class Main which would be used to accept two Input strings and call the static  
method present in UserMainCode.

**Input and Output Format:**

Input consists of a string with maximum size of 100 characters.

Output consists of a single string.

Refer sample output for formatting specifications.

**Sample Input 1:**

Welcome to the world of Programming

**Sample Output 1:**

Programming

**Sample Input 2:**

ABC DEF

**Sample Output 2:**

ABC

**Main;**

import java.util.Scanner;

```

public class Main
{
public static void main(String[] args)
{
Scanner s=new Scanner(System.in);
String s1=s.nextLine();
System.out.println(UserMainCode.getLargestWord(s1));
s.close();
}

```

```

}
UserMainCode;

import java.util.StringTokenizer;

public class UserMainCode
{
    public static String getLargestWord(String s1)
    {
        int max=0;
        String s2=new String();
        StringTokenizer t=new StringTokenizer(s1," ");
        {
            while(t.hasMoreTokens()){
                String s3=t.nextToken();
                int n=s3.length();
                if(n>max){
                    max=n;
                    s2=s3;
                }
            }
        }
    }
}

```

## 5. String Occurrences

Obtain two strings from user as input. Your program should count the number of occurrences of second word of second sentence in the first sentence.

Return the count as output. Note - Consider case.

Include a class UserMainCode with a static method **countNumberOfWords** which accepts two string variables. The return type is the modified string.

Create a Class Main which would be used to accept two Input strings and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of two strings with maximum size of 100 characters.

Output consists of a single string.

Refer sample output for formatting specifications.

### **Sample Input 1:**

abc bcd abc bcd abc abc

av abc

**Sample Output 1:**

4

**Sample Input 2:**

ABC xyz AAA

w abc

**Sample Output 2:**

0

**UserMainCode**

```
import java.util.StringTokenizer;

public class UserMainCode {

    public static void countNoOfWords(String s1, String s2) {
        int count=0;

        StringTokenizer st=new StringTokenizer(s2, " ");
        String s3=st.nextToken();
        String s4=st.nextToken();

        //System.out.println(s4);

        StringTokenizer st1=new StringTokenizer(s1, " ");
        while(st1.hasMoreTokens())
        {
            String s5=st1.nextToken();
            if(s4.equals(s5))
            {
```

```
        count++;
    }
}

System.out.println(count);
}
}
```

## Main

```
import java.util.*;

public class Main

{
    /**
     * @param args
     */
    public static void main(String[] args)
    {
        Scanner s=new Scanner(System.in);

        String s1=s.nextLine();

        String s2=s.nextLine();

        UserMainCode.countNoOfWords(s1,s2);

        s.close();
    }
}
```

## 6. ArrayList Manipulation

Write a program that performs the following actions:

1. Read  $2n$  integers as input.
2. Create two arraylists to store  $n$  elements in each arraylist.
3. Write a function **generateOddEvenList** which accepts these two arraylist as input.
4. The function fetch the odd index elements from first array list and even index elements from second array list and add them to a new array list according to their index.
5. Return the arraylist.

Include a class UserMainCode with the static method **generateOddEvenList** which accepts two arraylist and returns an arraylist.

Create a Class Main which would be used to read  $2n$  integers and call the static method present in UserMainCode.

Note:

- The index of first element is 0.
- Consider 0 as an even number.
- Maintain order in the output array list

#### **Input and Output Format:**

Input consists of  $2n+1$  integers. The first integer denotes the size of the arraylist, the next  $n$  integers are values to the first arraylist, and the last  $n$  integers are values to the second arraylist.

Output consists of a modified arraylist as per step 4.

Refer sample output for formatting specifications.

#### **Sample Input 1:**

5

12

13

14

15

16

2

3

4

5

6

**Sample Output 1:**

2

13

4

15

6

**UserMainCode**

```
import java.util.ArrayList;
import java.util.Iterator;
public class UserMainCode
{
    public static ArrayList<Integer> generateOddEvenList
    (ArrayList<Integer>a1,ArrayList<Integer>a2)
    {
        ArrayList<Integer>a3=new ArrayList<Integer>();
        for(int i=0;i<a1.size();i++)
        {
            if(i%2==0)
                a3.add(a2.get(i));
            else
                a3.add(a1.get(i));
        }
        return a3;
    }
}
```

```
}
```

## Main

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Scanner;

public class Main {

    public static void main(String []args){

        Scanner sc=new Scanner(System.in);

        int s=Integer.parseInt(sc.nextLine());

        ArrayList<Integer>al1=new ArrayList<Integer>();
        ArrayList<Integer>al2=new ArrayList<Integer>();

        for(int i=0;i<s;i++)

            al1.add(sc.nextInt());

        for(int i=0;i<s;i++)

            al2.add(sc.nextInt());

        ArrayList<Integer>al3=new ArrayList<Integer>();

        al3=UserMainCode.generateOddEvenList(al1,al2);

        Iterator<Integer> it=al3.iterator();

        while(it.hasNext())

        {

            int n=it.next();

            System.out.println(n);

            sc.close();

        }

    }

}
```

}

## 7. Duplicate Characters

Write a Program which removes duplicate characters from the string. Your program should read a sentence (string) as input from user and return a string removing duplicate characters. Retain the first occurrence of the duplicate character. Assume the characters are case – sensitive.

Include a class UserMainCode with a static method **removeDuplicates** which accepts a string. The return type is the modified sentence of type string.

Create a Class Main which would be used to accept the input string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string with maximum size of 100 characters.

Output consists of a single string.

Refer sample output for formatting specifications.

### **Sample Input 1:**

hi this is sample test

### **Sample Output 1:**

hi tsample

### **Sample Input 2:**

ABC DEF

### **Sample Output 2:**

ABC DEF

### **UserMainCode**

```
import java.util.HashSet;  
import java.util.Iterator;  
import java.util.LinkedHashSet;
```

```
import java.util.StringTokenizer;

public class UserMainCode

{
    public static void removeDuplicates(String s1) {
        char a[]={s1.toCharArray()};

        StringBuffer sb=new StringBuffer();

        LinkedHashSet<Character>hs=new LinkedHashSet<Character>();

        for(int i=0;i<a.length;i++)
        {
            hs.add(a[i]);
        }

        Iterator<Character>itr=hs.iterator();

        while(itr.hasNext())
        {
            char o=itr.next();

            if(o!=' ')
            {
                sb.append(o);
            }
        }

        System.out.println(sb);
    }
}
```

## Main

```
import java.util.*;
```

```

public class Main {

public static void main(String[] args)

{

    Scanner s=new Scanner(System.in);

    String s1=s.nextLine();

    UserMainCode.removeDuplicates(s1);

    s.close();

}

}

```

## **8. Mastering Hashmap**

You have recently learnt about hashmaps and in order to master it, you try and use it in all of your programs.

Your trainer / teacher has given you the following exercise:

1. Read  $2n$  numbers as input where the first number represents a key and second one as value. Both the numbers are of type integers.
2. Write a function **getAverageOfOdd** to find out average of all values whose keys are represented by odd numbers.

Assume the average is an int and never a decimal number. Return the average as output. Include this function in class UserMainCode.

Create a Class Main which would be used to read  $2n$  numbers and build the hashmap. Call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a  $2n+1$  integers. The first integer specifies the value of  $n$  (essentially the hashmap size). The next pair of  $n$  numbers denote the key and value.

Output consists of an integer representing the average.

Refer sample output for formatting specifications.

### **Sample Input 1:**

4

2

34

1

4

5

12

4

22

**Sample Output 1:**

8

**UserMainCode**

```
import java.util.HashMap;
import java.util.Scanner;
import java.util.HashMap;
import java.util.Iterator;
public class UserMainCode {
    public static int getAverageOfOdd(HashMap<Integer, Integer> h1)
    {
        int av=0, c=0, s=0;
        Iterator<Integer> it=h1.keySet().iterator();
        while(it.hasNext())
        {
            int a=it.next();
            if(a%2!=0)
            {
                int b=h1.get(a);
```

```

s=s+b;

c++;
}

}

av=s/c;

return av;

}}



Main

import java.util.*;

public class Main

{

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

HashMap<Integer,Integer> h1=new HashMap<Integer,Integer>();

for(int i=0;i<n;i++)

{

h1.put(sc.nextInt(),sc.nextInt());

}

System.out.println(UserMainCode.getAverageOfOdd(h1));

sc.close();

}

}

```

## 9. Managers & Hashmaps

A Company wants to automate its payroll process. You have been assigned as the programmer to build this package. You would like to showcase your skills by creating a quick prototype. The prototype consists of the following steps:

1. Read Employee details from the User. The details would include id, designation and salary in the given order. The datatype for id is integer, designation is string and salary is integer.
2. You decide to build two hashmaps. The first hashmap contains employee id as key and designation as value, and the second hashmap contains same employee ids as key and salary as value.
3. The company decides to hike the salary of managers by 5000. You decide to write a function **increaseSalaries** which takes the above hashmaps as input and returns a hashmap with only managers id and their increased salary as output. Include this function in class UserMainCode.

Create a Class Main which would be used to read employee details in step 1 and build the two hashmaps.

Call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of employee details. The first number indicates the size of the employees. The next three values indicate the employee id, employee designation and employee salary.

Output consists of a single string. Refer sample output for formatting specifications.

**SampleInput1:**

2

2

programmer

3000

8

manager

50000

**SampleOutput1:**

8

55000

**UserMainCode**

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.HashMap;
public class UserMainCode
{public static HashMap<Integer, Integer>
display(HashMap<Integer, String>hm,HashMap<Integer, Integer>hm1)
{
HashMap<Integer, Integer>hm3=new HashMap<Integer, Integer>();
Iterator<Integer> it=hm.keySet().iterator();
while(it.hasNext())
{
int id=it.next();
String name=hm.get(id);
if(name.equals("manager"))
{int salary=hm1.get(id)+5000;
hm3.put(id,salary);
}
}
return hm3;
}
}
```

```
Main
import java.util.HashMap;
import java.util.Iterator;
import java.util.Scanner;
public class Main {
public static void main(String []args){
```

```

Scanner sc=new Scanner(System.in);

int s=Integer.parseInt(sc.nextLine());

HashMap<Integer,String>hm=new HashMap<Integer,String>();

HashMap<Integer,Integer>hm1=new HashMap<Integer,Integer>();

for(int i=0;i<s;i++)

{

int id=Integer.parseInt(sc.nextLine());

hm.put(id, sc.nextLine());

hm1.put(id,Integer.parseInt(sc.nextLine()));

}

HashMap<Integer,Integer>hm2=new HashMap<Integer,Integer>();

hm2=UserMainCode.display(hm,hm1);

Iterator<Integer> it=hm2.keySet().iterator();

while(it.hasNext())

{

int n=it.next();

int fac=hm2.get(n);

System.out.println(n);

System.out.println(fac);

}

}

}

```

#### **10. Check first and last word**

Write a program to check if the first word and the last word in the input string match.

Include a class **UserMainCode** with a static method “**check**” that accepts a string argument and returns an int. If the first word and the last word in the string match, the method returns the number of characters in the word. Else the method returns the sum of the number of characters in the first word and last word.

Create a class **Main** which would get the input as a String and call the static method **check** present in the UserMainCode.

**Input and Output Format:**

Input consists of a string.

Output is an integer.

**Sample Input 1:**

how are you you are how

**Sample Output 1:**

3

**Sample Input 2:**

how is your child

**Sample Output 2:**

8

**UserMainCode**

```
import java.util.StringTokenizer;

public class UserMainCode

{
    public static int check(String s)

    {
        int count=0;

        String fin="";
        StringTokenizer st=new StringTokenizer(s);

        String ini=st.nextToken();

        while(st.hasMoreTokens())
        {
            fin=st.nextToken();
        }
    }
}
```

```

if(ini.equals(fin))

count=ini.length();

else

count=ini.length()+fin.length();

return count;

}

}

```

## Main

```

import java.util.HashMap;

import java.util.Iterator;

import java.util.Scanner;

public class Main {

public static void main(String []args){

Scanner sc=new Scanner(System.in);

String age=sc.nextLine();

System.out.println(UserMainCode.check(age));

sc.close();

}}

```

## 11. Concatenate Characters

Given an array of Strings, write a program to take the last character of each string and make a new String by concatenating it.

Include a class **UserMainCode** with a static method “**concatCharacter**” that accepts a String array as input and returns the new String.

Create a class **Main** which would get the String array as input and call the static method **concatCharacter** present in the

UserMainCode.

**Input and Output Format:**

The first line of the input consists of an integer n that corresponds to the number of strings in the input string array.

The next n lines of the input consist of the strings in the input string array.

Output consists of a string.

**Sample Input:**

3

ab

a

abcd

**Sample Output:**

Bad

UserMainCode

```
public class UserMainCode
{
    public static String concatCharacter(String[] a)
    {
        StringBuffer sb=new StringBuffer();
        for(int i=0;i<a.length;i++)
            sb.append(a[i].charAt(a[i].length()-1));
        return sb.toString();
    }
}
```

Main

```

import java.util.*;

public class Main

{
    public static void main(String []args){

Scanner sc=new Scanner(System.in);

int s=Integer.parseInt(sc.nextLine());

String []a=new String[s];

for(int i=0;i<s;i++)

{

a[i]=sc.nextLine();

}

System.out.println(UserMainCode.concatCharacter(a));

sc.close();

}
}

```

## 12.Anagram

Write a program to check whether the two given strings are anagrams.

Note: Rearranging the letters of a word or phrase to produce a new word or phrase, using all the original letters exactly once is called Anagram."

Include a class **UserMainCode** with a static method "**getAnagram**" that accepts 2 strings as arguments and returns an int.

The method returns 1 if the 2 strings are anagrams. Else it returns -1.

Create a class **Main** which would get 2 Strings as input and call the static method **getAnagram** present in the UserMainCode.

### **Input and Output Format:**

Input consists of 2 strings. Assume that all characters in the string are lower case letters.

Output consists of a string that is either “Anagrams” or “Not Anagrams”.

**Sample Input 1:**

eleven plus two

twelve plus one

**Sample Output 1:**

Anagrams

**Sample Input 2:**

orchestra

carthorse

**Sample Output 2:**

Anagrams

**Sample Input 3:**

cognizant

technologies

**Sample Output 3:**

Not Anagrams

**UserMainCode**

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Scanner;

public class UserMainCode

{
    public static void getAnagram(String s1, String s2)
    {
        List<Character> l1=new ArrayList<Character>();
        List<Character> l2=new ArrayList<Character>();
        int n1=s1.length();
        int n2=s2.length();
        if(n1!=n2)
            System.out.println("Not Anagrams");
        else
        {
```

```
List<Character> l2=new ArrayList<Character>();

String s3=s1.replace(" ","");
String s4=s2.replace(" ","");
String s5=s3.toUpperCase();
String s6=s4.toUpperCase();

for (int i = 0; i < s5.length(); i++)
{
    l1.add(s5.charAt(i));
}

for (int i = 0; i < s6.length(); i++)
{
    l2.add(s6.charAt(i));
}

Collections.sort(l1);
Collections.sort(l2);

// System.out.println(l1);
// System.out.println(l2);

if(l1.equals(l2))
    System.out.println("Anagram");
else
    System.out.println("Not Anagram");
}
```

## Main

```
import java.util.ArrayList;
```

```

import java.util.Collections;
import java.util.List;
import java.util.Scanner;

public class Main{
    public static void main(String[] args)
    {
        Scanner sc =new Scanner(System.in);
        String s1=sc.nextLine();
        String s2=sc.nextLine();

        UserMainCode.getAnagram(s1,s2);

    }
}

```

### 13.Calculate Meter Reading

Given 2 strings corresponding to the previous meter reading and the current meter reading, write a program to calculate electricity bill.

The input string is in the format ""AAAAAXXXXX".

AAAAA is the meter code and XXXXX is the meter reading.

FORMULA: (XXXXX-XXXXX)\*4

Hint: if AAAAA of input1 and input2 are equal then separate the XXXXX from string and convert to integer. Assume that AAAAA of the 2 input strings will always be equal.

Include a class **UserMainCode** with a static method "**calculateMeterReading**" that accepts 2 String arguments and returns an integer that corresponds to the electricity bill. The 1<sup>st</sup> argument corresponds to the previous meter reading and the 2<sup>nd</sup> argument corresponds to the current meter reading.

Create a class **Main** which would get 2 Strings as input and call the static method **calculateMeterReading** present in the **UserMainCode**.

#### **Input and Output Format:**

Input consists of 2 strings. The first input corresponds to the previous meter reading and the second input corresponds to the current meter reading.

Output consists of an integer that corresponds to the electricity bill.

#### **Sample Input:**

CSECE12390

CSECE12400

#### **Sample Output:**

40

```

import java.util.Scanner;
public class Main{
    public static void main (String[] args)
    {
        // your code goes here
        Scanner sc = new Scanner(System.in);
    }
}

```

```

String input1=sc.next();
String input2=sc.next();
System.out.println(UserMainCode.calculateMeterReading(input1,input2));
sc.close();
}

public class UserMainCode {
    public static int calculateMeterReading(String input1, String input2)
    {
        int n1=Integer.parseInt(input1.substring(5,input1.length()));
        int n2=Integer.parseInt(input2.substring(5,input2.length()));
        int n=Math.abs((n2-n1)*4);
        return n;
    }
}

```

#### 14.Retirement

Given an input as HashMap which contains key as the ID and dob as value of employees, write a program to find out employees eligible for retirement. A person is eligible for retirement if his age is greater than or equal to 60.

Assume that the current date is 01/01/2014.

Include a class **UserMainCode** with a static method “retirementEmployeeList” that accepts a `HashMap<String, String>` as input and returns a `ArrayList<String>`. In this method, add the Employee IDs of all the retirement eligible persons to list and return the sorted list.

(Assume date is in dd/MM/yyyy format).

Create a class **Main** which would get the `HashMap` as input and call the static method **retirementEmployeeList** present in the `UserMainCode`.

#### Input and Output Format:

The first line of the input consists of an integer `n`, that corresponds to the number of employees.

The next 2 lines of the input consists of strings that correspond to the id and dob of employee 1.

The next 2 lines of the input consists of strings that correspond to the id and dob of employee 2.

and so on...

Output consists of the list of employee ids eligible for retirement in sorted order.

#### Sample Input :

```

4
C1010
02/11/1987
C2020
15/02/1980
C3030
14/12/1952
T4040
20/02/1950

```

#### Sample Output :

[C3030, T4040]

```

import java.text.ParseException;
import java.util.LinkedHashMap;
import java.util.Scanner;
public class Main
{
    public static void main(String args[]) throws ParseException{ Scanner sc=new
    Scanner(System.in); int
    n=Integer.parseInt(sc.nextLine());
    LinkedHashMap<String, String>a1=new LinkedHashMap<String, String>(); for(int
    i=0;i<n;i++) {
    a1.put(sc.nextLine(),sc.nextLine());
    }
    System.out.println(UserMainCode.retirementEmployeeList(a1));
    }
}
import java.text.*;
import java.util.*;
public class UserMainCode {

```

```

public static ArrayList<String>
retirementEmployeeList(LinkedHashMap<String, String>a1) throws ParseException
{
    ArrayList<String>a1=new ArrayList<String>();
    Iterator <String>it=a1.keySet().iterator();
    while(it.hasNext())
    {String s=it.next();
    String s1=a1.get(s);
    SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
    sdf.setLenient(false);
    try{
    Date d=new Date();
    Date d1=new Date();
    String a=s1;
    String b="01/01/2014";
    d=sdf.parse(a);
    d1=sdf.parse(b);
    long t=d.getTime();
    long t1=d1.getTime();
    long t3=t1-t;
    long l1=(24 * 60 * 60 * 1000);
    long l=l1*365;
    long res=t3/l;
    if(res>=60)
    {
    a1.add(s);
    }
    }
    catch (Exception e) {
    e.printStackTrace();
    }
    }
    Collections.sort(a1);
    return a1;
}
}

```

## 15.Kaprekar Number

Write a program to check whether the given input number is a Kaprekar number or not.

**Note :** A positive whole number ‘n’ that has ‘d’ number of digits is squared and split into two pieces, a right-hand piece that has ‘d’ digits and a left-hand piece that has remaining ‘d’ or ‘d-1’ digits. If the sum of the two pieces is equal to the number, then ‘n’ is a Kaprekar number.

If its Kaprekar number assign to output variable 1 else -1.

Example 1:

Input1:9

$9^2 = 81$ , right-hand piece of 81 = 1 and left hand piece of 81 = 8

Sum = 1 + 8 = 9, i.e. equal to the number. Hence, 9 is a Kaprekar number.

Example 2:

Input1:45

Hint:

$45^2 = 2025$ , right-hand piece of 2025 = 25 and left hand piece of 2025 = 20

Sum = 25 + 20 = 45, i.e. equal to the number. Hence, 45 is a Kaprekar number."

Include a class **UserMainCode** with a static method "**getKaprekarNumber**" that accepts an integer argument and returns an integer. The method returns 1 if the input integer is a Kaprekar number. Else the method returns -1.

Create a class **Main** which would get the an Integer as input and call the static method **getKaprekarNumber** present in the **UserMainCode**.

### Input and Output Format:

Input consists of an integer.

Output consists of a single string that is either “Kaprekar Number” or “Not A Kaprekar Number”

**Sample Input 1:**

```

9
Sample Output 1:
Kaprekar Number
Sample Input 2:
45
Sample Output 2:
Kaprekar Number
Sample Input 3:
4
Sample Output 3:
Not A Kaprekar Number

import java.util.*;
public class Main{
public static void main(String[] args)
{
Scanner sc=new Scanner(System.in);
int n=sc.nextInt();
int v=UserMainCode.getKaprekarNumber(n);
if (v==1)
System.out.println("Kaprekar Number");
else
System.out.println("Not a Kaprekar Number");
}}

public class UserMainCode {

    public static int getKaprekarNumber(int a)
    {
        int count=0,j=0;
        int a1=a;
        while(a1!=0)
        {
            count=count+1;
            a1=a1/10;
        }
        int square=a*a;
        String s=Integer.toString(square);
        String s1=s.substring(0,count);
        String s2=s.substring(count);
        int x=Integer.parseInt(s1);
        int y=Integer.parseInt(s2);
        int result =x+y;
        if(result==a){
            j=1;
        }
        else
        {
            j=2;
        }
        return j;
    }
}

```

## 16.Vowels

Given a String input, write a program to find the word which has the the maximum number of vowels. If two or more words have the maximum number of vowels, print the first word.

Include a class **UserMainCode** with a static method “**storeMaxVowelWord**” that accepts a string argument and returns the word containing the maximum number of vowels.

Create a class **Main** which would get the a String as input and call the static method **storeMaxVowelWord** present in the **UserMainCode**.

**Input and Output Format:**

Input consists of a string. The string may contain both lower case and upper case letters.  
Output consists of a string.

**Sample Input :**

What is your name?

**Sample Output :**

your

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        String s1 =sc.nextLine();
        UserMainCode.storeMaxVowelWord(s1);

    }
}

import java.util.StringTokenizer;

public class UserMainCode {
    public static void storeMaxVowelWord(String s1) {
        int i = 0;
        StringTokenizer st = new StringTokenizer(s1, " ");
        int len = 0;
        int count = 0;
        int count2 = 0;
        String s6 = null;
        while (st.hasMoreTokens()) {
            String s5 = st.nextToken();
            len = s5.length();
            count=0;
            for (i = 0; i < len; i++) {
                if (s5.charAt(i) == 'a' || s5.charAt(i) == 'e'|| s5.charAt(i) == 'i'
                || s5.charAt(i) == 'o'|| s5.charAt(i) == 'u'
                ||s5.charAt(i) == 'A' ||s5.charAt(i) == 'E' ||s5.charAt(i) == 'I'
                ||s5.charAt(i) == 'O' ||s5.charAt(i) == 'U')
                    count++;
            }
            if (count > count2)
            {
                count2 = count;
                s6 = s5;
            }
        }
        System.out.println(s6);
    }
}
```

### 17.Unique Characters REPEATED

Given a String as input , write a program to count and print the number of unique characters in it.

Include a class **UserMainCode** with a static method “**checkUnique**” that accepts a String as input and returns the number of unique characters in it. If there are no unique characters in the string, the method returns -1.

Create a class **Main** which would get a String as input and call the static method **checkUnique** present in the UserMainCode.

**Input and Output Format:**

Input consists of a string.

Output consists of an integer.

**Sample Input 1:**

HOWAREYOU

**Sample Output 1:**

(Hint :Unique characters are : H,W,A,R,E,Y,U and other characters are repeating)

**Sample Input 2:**

MAMA

**Sample Output2:**

-1

```

import java.util.*;
public class Main
{
public static void main(String[] args)
{
    Scanner sc=new Scanner(System.in);
String s1=sc.next();
UserMainCode.checkUnique(s1);
}
}

import java.util.*;
public class UserMainCode {

    public static void checkUnique(String s1)
    {
        String s2=s1.toLowerCase();
        StringBuffer sb=new StringBuffer(s2);
        int l=sb.length();
        int count=0;
        for(int i=0;i<l;i++)
        { count=0;
        for(int j=i+1;j<l;j++)
        {
            if(sb.charAt(i)==sb.charAt(j))
            {
                sb.deleteCharAt(j);
                count++;
                j--;
                l--;
            }
        }
        if(count>0)
        {
            sb.deleteCharAt(i);
            i--;
            l--;
        }
    }
    if(sb.length()==0)
    {
        System.out.println(-1);
    }
    else
        System.out.println(sb.length());
    }
}

```

## 18.average of primes

Write a program to read an array and find average of all elements located at index i, where i is a prime number. Type cast the average to an int and return as output. The index starts from 0.

Include a class UserMainCode with a static method **addPrimeIndex** which accepts a single integer array. The return type (integer) should be the average of all elements located at index i where i is a prime number.

Create a Class Main which would be used to accept Input array and call the static method present in UserMainCode.

### Input and Output Format:

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

Output consists of a single Integer.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20 and minimum number of elements is 3.

### Sample Input 1:

```
4  
2  
5  
2  
4
```

### Sample Output 1:

```
3
```

```
import java.util.Scanner;  
public class Main{  
    public static void main (String[] args)  
{  
    // your code goes here  
    Scanner sc = new Scanner(System.in);  
    int n = sc.nextInt();  
    System.out.println(UserMainCode.addPrimeIndex(n));  
}  
import java.util.*;  
public class UserMainCode {  
    public static int addPrimeIndex(int n) {  
        Scanner sc=new Scanner(System.in);  
        int[] a = new int[n];  
        for(int i=0;i<n;i++){  
            a[i] = sc.nextInt();  
        }  
        int sum=0;  
        int count=0;  
        int sum_count=0;  
        for(int i=0;i<a.length;i++)  
        {  
            count=0;  
            for(int j=1;j<=i;j++)  
            {  
                if(i%j==0)  
                {  
                    count++;  
                }  
            }  
            if(count==2)  
            {  
                sum=sum+a[i];  
                sum_count++;  
            }  
        }  
        int avg=sum/sum_count;  
        return avg;  
}}
```

## 19. ArrayList and Set Operations

Write a program that performs the following actions:

1. Read  $2n$  integers as input & a set operator (of type char).
2. Create two arraylists to store  $n$  elements in each arraylist.
3. Write a function **performSetOperations** which accepts these two arraylist and the set operator as input.
4. The function would perform the following set operations:

'+' for SET-UNION

'\*' for SET-INTERSECTION

'-' for SET-DIFFERENCE

Refer to sample inputs for more details.

5. Return the resultant arraylist.

Include a class UserMainCode with the static method **performSetOperations** which accepts two arraylist and returns an arraylist.

Create a Class Main which would be used to read  $2n+1$  integers and call the static method present in UserMainCode.

Note:

- The index of first element is 0.

### **Input and Output Format:**

Input consists of  $2n+2$  integers. The first integer denotes the size of the arraylist, the next  $n$  integers are values to the first arraylist, and the next  $n$  integers are values to the second arraylist and the last input corresponds to that set operation type.

Output consists of a modified arraylist as per step 4.

Refer sample output for formatting specifications.

### **Sample Input 1:**

```
3  
1  
2  
3  
3  
5  
7  
+
```

### **Sample Output 1:**

```
1  
2  
3  
5  
7
```

### **Sample Input 2:**

```
4  
10  
9  
8  
7  
2  
4  
6  
8  
*
```

### **Sample Output 2:**

```
8
```

### **Sample Input 3:**

```
4  
5  
10  
15  
20  
0  
10  
12  
20  
-
```

### **Sample Output 3:**

```
5  
15
```

### **Main:**

```

import java.util.ArrayList;
import java.util.Scanner;
public class Main {
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        int n=Integer.parseInt(sc.nextLine());
        ArrayList<Integer>a1=new ArrayList<Integer>();
        ArrayList<Integer>a2=new ArrayList<Integer>();
        for(int i=0;i<n;i++)
            a1.add(Integer.parseInt(sc.nextLine()));
        for(int i=0;i<n;i++)
            a2.add(Integer.parseInt(sc.nextLine()));
        char c=sc.nextLine().charAt(0);
        System.out.println(UserMainCode.performSetOperations(a1,a2,c));
    }
}

```

### **UserMainCode:**

```

import java.util.ArrayList;
import java.util.ArrayList;
public class UserMainCode {
    public static ArrayList<Integer>
    performSetOperations(ArrayList<Integer>a1,ArrayList<Integer>a2,char c)
    {
        ArrayList<Integer>op1=new ArrayList<Integer>();
        int k=0;
        switch(c)
        {
            case '+':
                a1.removeAll(a2);

```

```
a1.addAll(a2);
op1=a1;
break;
case '*':
a1.retainAll(a2);
op1=a1;
break;
case '-':
for(int i=0;i<a1.size();i++)
{
k=0;
for(int j=0;j<a2.size();j++)
{
if(a1.get(i)==a2.get(j))
k=1;
}
if(k==0)
op1.add(a1.get(i));
}
break;
}
return op1;
}}
}

return tm;
```

```
}
```

## 20.Largest Span

Write a program to read an array and find the size of largest span in the given array  
""span"" is the number of elements between two repeated numbers including both numbers. An array with single element has a span of 1.

Include a class UserMainCode with a static method **getMaxSpan** which accepts a single integer array. The return type (integer) should be the size of largest span.

Create a Class Main which would be used to accept Input array and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

Output consists of a single Integer.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20.

### **Sample Input 1:**

```
5
1
2
1
1
3
```

### **Sample Output 1:**

```
4
```

### **Sample Input 2:**

```
7
1
4
2
1
4
1
5
```

### **Sample Output 2:**

```
6
```

### **MAIN:**

```
import java.util.Scanner;
public class Main {
public static void main(String[] args)
{
Scanner sc=new Scanner(System.in);
int n=sc.nextInt();
int []a=new int[n];
for(int i=0;i<n;i++)
{
a[i]=sc.nextInt();
}
System.out.print(UserMainCode.getMaxSpan(a,n));
}}
```

### **USERMAINCODE:**

```
class UserMainCode {
public static int getMaxSpan(int[] x,int n)
{
int gap=0,max=0;
for(int i=0;i<n;i++)
{
for(int j=i+1;j<n;j++)
{
if(x[i]==x[j])
```

```

gap=j;
}
if(gap-i>max)
max=gap-i;
}
return max+1;
}
}

```

## 21. Max Scorer

Write a program that performs the following actions:

1. Read n strings as input and stores them as an arraylist. The string consists of student information like name and obtained marks of three subjects. Eg: name-mark1-mark2-mark3 [suresh-70-47-12]. The mark would range between 0 – 100 (inclusive).
2. Write a function **highestScorer** which accepts these arraylist and returns the name of the student who has scored the max marks. Assume the result will have only one student with max mark.

Include a class UserMainCode with the static method **highestScorer** which accepts the arraylist and returns the name

(string) of max scorer.

Create a Class Main which would be used to read n strings into arraylist and call the static method present in UserMainCode.

### Input and Output Format:

Input consists of 1 integer and n strings. The first integer denotes the size of the arraylist, the next n strings are score pattern described above.

Output consists of a string with the name of the top scorer.

Refer sample output for formatting specifications.

### Sample Input 1:

```

3
sunil-56-88-23
bindul-88-70-10
john-70-49-65

```

### Sample Output 1:

John

### USERMAINCODE:

```

import java.util.ArrayList;
import java.util.StringTokenizer;
public class UserMainCode
{
public static String highestScorer(ArrayList<String>s1)
{
int max=0;
String s4=null;
for(int i=0;i<s1.size();i++)
{
    String s2=s1.get(i);
    StringTokenizer t=new StringTokenizer(s2, "-");

```

```

        String s3=t.nextToken();
        int n1=Integer.parseInt(t.nextToken());
        int n2=Integer.parseInt(t.nextToken());
        int n3=Integer.parseInt(t.nextToken());
        int n=n1+n2+n3;
        if(n>max)
        {
            max=n;
            s4=s3;
        }
    }
    return s4;
}
}

MAIN:

import java.util.*;
public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
Scanner s=new Scanner(System.in);
int n=s.nextInt();
ArrayList<String> s1=new ArrayList<String>();
for(int i=0;i<n;i++)
{
    s1.add(s.next());
}
System.out.println(UserMainCode.highestScorer(s1));
s.close();

    }
}

```

## 22. Max Vowels

Write a Program which fetches the word with maximum number of vowels. Your program should read a sentence as input from user and return the word with max number of vowels. In case there are two words of maximum length return the word which comes first in the sentence.

Include a class UserMainCode with a static method **getWordWithMaximumVowels** which accepts a string The return type is the longest word of type string.

Create a Class Main which would be used to accept two Input strings and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string with maximum size of 100 characters.

Output consists of a single string.

Refer sample output for formatting specifications.

### **Sample Input 1:**

Appreciation is the best way to motivate

### **Sample Output 1:**

Appreciation

### USERMAINCODE:

```
import java.util.StringTokenizer;

public class UserMainCode {
public static String getWordWithMaximumVowels(String s1)
{
    int i;
    StringTokenizer t=new StringTokenizer(s1, " ");
    int count=0,max=0;
    String s2=null;
    while(t.hasMoreTokens())
    {
        String s3=t.nextToken();
        count=0;
        for(i=0;i<s3.length();i++)
        {

            if(s3.charAt(i)=='a' || s3.charAt(i)=='e' || s3.charAt(i)=='i' || s3.charAt(i)=='o' || s3.charAt(i)=='u'
               || s3.charAt(i)=='A' || s3.charAt(i)=='E' || s3.charAt(i)=='I' || s3.charAt(i)=='O'
               || s3.charAt(i)=='U')
                count++;

        }
        if(count>max)
        {
            max=count;
            s2=s3;
        }
    }
    return s2;
}
}
```

### MAIN:

```
import java.util.*;
public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
Scanner s=new Scanner(System.in);
String s1=s.nextLine();
System.out.println(UserMainCode.getWordWithMaximumVowels(s1));
s.close();
    }

}
```

## 23. All Vowels

Write a Program to check if given word contains exactly five vowels and the vowels are in alphabetical order. Return 1 if the condition is satisfied else return -1. Assume there is no repetition of any vowel in the given string and all letters are in lower case.

Include a class UserMainCode with a static method **testOrderVowels** which accepts a string. The return type is integer based on the condition stated above.

Create a Class Main which would be used to accept two Input strings and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of a string with maximum size of 100 characters.

Output consists of a single string.

Refer sample output for formatting specifications.

**Sample Input 1:**

acebisouzz

**Sample Output 1:**

valid

**Sample Input 2:**

Alphabet

**Sample Output 2:**

Invalid

**USERMAINCODE:**

```
public class UserMainCode {  
    public static int getOrderVowels(String s1)  
{  
  
        String s2="aeiou";  
        StringBuffer sb=new StringBuffer();  
        for(int i=0;i<s1.length();i++)  
        {  
            for(int j=0;j<s2.length();j++)  
            {  
                if(s1.charAt(i)==s2.charAt(j))  
                {  
                    sb.append(s1.charAt(i));  
                }  
            }  
        }  
        if(sb.toString().equals(s2))  
        {  
            return 1;  
        }  
        return -1;  
    }  
}
```

**MAIN:**

```
import java.util.*;  
public class Main {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        Scanner s=new Scanner(System.in);  
        String s1=s.next();
```

```

int b=UserMainCode.getOrderVowels(s1);
if(b==1)
{
    System.out.println("Valid");
}
else
    System.out.println("Invalid");
s.close();
}

}

```

## 24. Adjacent Swaps

Write a Program that accepts a string as a parameter and returns the string with each pair of adjacent letters reversed. If the string has an odd number of letters, the last letter is unchanged.

Include a class UserMainCode with a static method **swapPairs** which accepts a string. The return type is string which is reversed pair of letters.

Create a Class Main which would be used to accept two Input strings and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of a string with maximum size of 100 characters.

Output consists of a single string.

Refer sample output for formatting specifications.

**Sample Input 1:**

forget

**Sample Output 1:**

ofgrte

**Sample Input 2:**

New York

**Sample Output 2:**

eN woYkr

**USERMAINCODE:**

```

import java.util.*;
public class UserMainCode {
public static String swapPairs(String s1)
{
    int i;
    StringBuffer sb=new StringBuffer();
    for(i=0;i<s1.length()-1;i=i+2)
    {
        if(i%2==0)
        {
            char a=s1.charAt(i);
            char b=s1.charAt(i+1);
            sb.append(b).append(a);
        }
        else
            for(i=0;i<s1.length()-1;i=i+2)
        {
            char a=s1.charAt(i);
            char b=s1.charAt(i+1);

```

```

        sb.append(b).append(a);
        sb.append(s1.charAt(s1.length()-1));
    }

}

return sb.toString();
}
}

MAIN:

import java.util.*;
public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
Scanner s=new Scanner(System.in);
String s1=s.nextLine();
System.out.println(UserMainCode.swapPairs(s1));
s.close();
    }

}

```

## 25. Sum of Digits

Write a Program that accepts a word as a parameter, extracts the digits within the string and returns its sum.

Include a class UserMainCode with a static method **getdigits** which accepts a string. The return type is integer representing the sum.

Create a Class Main which would be used to accept the input string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string with maximum size of 100 characters.

Output consists of a single string.

Refer sample output for formatting specifications.

### **Sample Input 1:**

abc12de4

### **Sample Output 1:**

7

### **USERMAINCODE:**

```

public class UserMainCode {
public static int getDigits(String s1)
{
    int sum=0;
    for(int i=0;i<s1.length();i++)
    {
        char a=s1.charAt(i);
        if(Character.isDigit(a))
        {
            int b=Integer.parseInt(String.valueOf(a));
            sum=sum+b;
        }
    }
}

```

```

        }
        return sum;
    }
}

MAIN:

import java.util.*;
public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
Scanner s=new Scanner(System.in);
String s1=s.next();
System.out.println(UserMainCode.getDigits(s1));
s.close();
    }

}

```

## 26. Password

Given a String , write a program to find whether it is a valid password or not.

Validation Rule:

Atleast 8 characters

Atleast 1 number(1,2,3...)

Atleast 1 special character(@,#,%...)

Atleast 1 alphabet(a,B...)

Include a class **UserMainCode** with a static method “**validatePassword**” that accepts a String argument and returns a boolean value. The method returns true if the password is acceptable. Else the method returns false.

Create a class **Main** which would get a String as input and call the static method **validatePassword** present in the UserMainCode.

### **Input and Output Format:**

Input consists of a String.

Output consists of a String that is either “Valid” or “Invalid”.

#### **Sample Input 1:**

cts@1010

#### **Sample Output 1:**

Valid

#### **Sample Input 2:**

punitha3

**Sample Output 2:**

Invalid

**USERMAINCODE:**

```
public class UserMainCode {  
  
    public static boolean validatePassword(String s1)  
  
    {  
  
        boolean b=false;  
  
        if(s1.length()>=8)  
  
            b=true;  
  
        if(b=true)  
  
        {  
  
            if(s1.matches(".*[0-9]{1,}.*")&&s1.matches(".*[a-zA-Z]{1,}.*")&&s1.matches(".*[@#%]{1,}.*"))  
  
            {  
  
                b=true;  
  
            }  
  
        else  
  
            b=false;  
  
        }  
  
        return b;  
  
    }  
  
}
```

**MAIN:**

```
import java.util.*;  
  
public class Main {  
  
    public static void main(String [] args)  
  
    {  
  
        Scanner s=new Scanner(System.in);
```

```

String s1=s.nextLine();

boolean b=(UserMainCode.validatePassword(s1));

if(b==true)
{
    System.out.println("Valid");

}
else

System.out.println("Invalid");

s.close();
}

}

```

## **27. Employee Bonus**

A Company wants to give away bonus to its employees. You have been assigned as the programmer to automate this process. You would like to showcase your skills by creating a quick prototype. The prototype consists of the following steps:

1. Read Employee details from the User. The details would include id, DOB (date of birth) and salary in the given order. The datatype for id is integer, DOB is string and salary is integer.

2. You decide to build two hashmaps. The first hashmap contains employee id as key and DOB as value, and the second hashmap contains same employee ids as key and salary as value.

3. If the age of the employee in the range of 25 to 30 years (inclusive), the employee should get bonus of 20% of his salary and in the range of 31 to 60 years (inclusive) should get 30% of his salary. store the result in TreeMap in which Employee ID as key and revised salary as value. Assume the age is caculated based on the date 01-09-2014.

(Typecast the bonus to integer).

4. Other Rules:

a. If Salary is less than 5000 store -100.

b. If the age is less than 25 or greater than 60 store -200.

c. a takes more priority than b i.e both if a and b are true then store -100.

5. You decide to write a function **calculateRevisedSalary** which takes the above hashmaps as input and returns the treemap as output. Include this function in class UserMainCode.

Create a Class Main which would be used to read employee details in step 1 and build the two hashmaps. Call the static

method present in UserMainCode.

**Input and Output Format:**

Input consists of employee details. The first number indicates the size of the employees. The next three values indicate the employee id, employee DOB and employee salary. The Employee DOB format is “dd-mm-yyyy”

Output consists of a single string.

Refer sample output for formatting specifications.

**Sample Input 1:**

2

1010

20-12-1987

10000

2020

01-01-1985

14400

**Sample Output 1:**

1010

12000

2020

17280

**USERMAINCODE:**

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.TreeMap;
class UserMainCode {
    public static TreeMap<Integer,Integer>
    calculateRevisedSalary(LinkedHashMap<Integer,String>a1,LinkedHashMap<Integer,Integer>a2) throws
    ParseException
    {TreeMap<Integer,Integer>ans=new TreeMap<Integer,Integer>();
    ArrayList<String>al=new ArrayList<String>();
    Iterator <Integer>it=a1.keySet().iterator();
    while(it.hasNext())
    {int s=it.next();
    String s1=a1.get(s);
    SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");
    sdf.setLenient(false);
    try{
```

```

Date d=new Date();
Date d1=new Date();
String a=s1;
String b="01-09-2014";
d=sdf.parse(a);
d1=sdf.parse(b);
long t=d.getTime();
long t1=d1.getTime();
long t3=t1-t;
long l1=(24 * 60 * 60 * 1000);
long l=l1*365;
long res=t3/l;
//System.out.println("Result="+res);
if(res>=25 && res<=30)
{
float bonus=(float)((0.2*a2.get(s))+a2.get(s));
int r=(int)bonus;
ans.put(s,r );
}
else if(res>30 && res<=60)
{
float bonus=(float)((0.3*a2.get(s))+a2.get(s));
int r=(int)bonus;
ans.put(s,r );
}
else if(a2.get(s)<5000)
{
ans.put(s, -100);
}
else if(res<25 ||res>60)
{
ans.put(s, -200);
}
}
catch (Exception e) {
e.printStackTrace();
}
}
return ans;
}
}

```

## **MAIN:**

```

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.Scanner;
import java.util.TreeMap;
public class Main
{
public static void main(String args[]) throws ParseException{
Scanner sc=new Scanner(System.in);
int n=sc.nextInt();
LinkedHashMap<Integer,String>a1=new LinkedHashMap<Integer,String>();
LinkedHashMap<Integer,Integer>a2=new LinkedHashMap<Integer,Integer>();
TreeMap<Integer,Integer>ans=new TreeMap<Integer, Integer>();

```

```

for(int i=0;i<n;i++)
{
int id=sc.nextInt();
a1.put(id,sc.nextInt());
int salary=sc.nextInt();
a2.put(id,salary);
}
ans=UserMainCode.calculateRevisedSalary(a1,a2);
Iterator <Integer>it=ans.keySet().iterator();
while(it.hasNext())
{
int a=it.next();
int b=ans.get(a);
System.out.println(a);
System.out.println(b);
}
}
}
}

```

## **28. Grade Calculator REFER 53 FROM LEVEL2**

A School wants to assign grades to its students based on their marks. You have been assigned as the programmer to automate this process. You would like to showcase your skills by creating a quick prototype. The prototype consists of the following steps:

1. Read student details from the User. The details would include roll no, mark in the given order. The datatype for id is integer, mark is integer.

2. You decide to build a hashmap. The hashmap contains roll no as key and mark as value.

### **3. BUSINESS RULE:**

1. If Mark is greater than or equal to 80 store medal as ""GOLD"".

2. If Mark is less than 80 and greater than or equal to 60 store medal as ""SILVER"".

3 .If Mark is less than 60 and greater than or equal to 45 store medal as ""BRONZE"" else store ""FAIL"".

Store the result in TreeMap in which Roll No as Key and grade as value.

4. You decide to write a function **calculateGrade** which takes the above hashmap as input and returns the treemap as output. Include this function in class UserMainCode.

Create a Class Main which would be used to read employee details in step 1 and build the two hashmap. Call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of employee details. The first number indicates the size of the students. The next two values indicate the roll id, mark.

Output consists of a single string.

Refer sample output for formatting specifications.

**Sample Input 1:**

2

1010

80

100

40

**Sample Output 1:**

100

FAIL

1010

GOLD

**USERMAINCODE:**

```
import java.util.Iterator;
import java.util.HashMap;
import java.util.TreeMap;
public class UserMainCode
{
    public static TreeMap<Integer,String>calculateGrade(HashMap<Integer,Integer>hm)
    {
        TreeMap<Integer,String>tm=new TreeMap<Integer,String>();
        Iterator<Integer> it=hm.keySet().iterator();
        while(it.hasNext())
        {
            int id=it.next();
            int mark=hm.get(id);
            if(mark>=80)
                tm.put(id,"GOLD");
            else if(mark<80 && mark>=60)
                tm.put(id,"SILVER");
            else if(mark<60 && mark>=45)
                tm.put(id,"BRONZE");
            else
                tm.put(id,"FAIL");
        }
        return tm;
    }
}
```

**MAIN:**

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.HashMap;
import java.util.TreeMap;
import java.util.Scanner;
public class Main {
    public static void main(String []args){
        Scanner sc=new Scanner(System.in);
```

```

int s=sc.nextInt();
HashMap<Integer,Integer>hm=new HashMap<Integer,Integer>();
for(int i=0;i<s;i++)
{
hm.put(sc.nextInt(),sc.nextInt());
}
TreeMap<Integer,String>tm=new TreeMap<Integer,String>();
tm=UserMainCode.calculateGrade(hm);
Iterator<Integer> it=tm.keySet().iterator();
for(int i=0;i<s;i++)
{
int n=it.next();
String fac=tm.get(n);
System.out.println(n);
System.out.println(fac);
}
}
}

```

## 29.Digits - II

Write a program to read a non-negative integer n, compute the sum of its digits. If sum is greater than 9 repeat the process and calculate the sum once again until the final sum comes to single digit.Return the single digit. Include a class UserMainCode with a static method getDigitSum which accepts the integer value. The return type is integer. Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

**Input and Output Format:** Input consists of a integer. Output consists of integer. Refer sample output for formatting specifications.

Sample Input 1: 9999 Sample Output 1: 9

Sample Input 2: 698 Sample Output 2: 5

### **MAIN:**

```

import java.util.Scanner;

public class Main {

    public static void main(String[] args)

    {
        Scanner s=new Scanner(System.in);

```

```
int a=s.nextInt();

int sum=UserMainCode.getDigitSum(a);

System.out.println(sum);

}

}
```

### **USERMAINCODE:**

```
public class UserMainCode

{

public static int getDigitSum(int n)

{

    int sum = 0 ;

    int n1=n;

    while(n>10)

    {

        int a = 0 ; sum = 0;

        while(n!=0)

        {

            a = n%10;

            sum+=a;

            n=n/10;

        }

        n=sum;

    }

}
```

```
    return sum;  
}  
  
}
```

### 30.Anagrams

Write a program to read two strings and checks if one is an anagram of the other. An anagram is a word or a phrase that can be created by rearranging the letters of another given word or phrase. We ignore white spaces and letter case. All letters of 'Desperation' can be rearranged to the phrase 'A Rope Ends It'. Include a class UserMainCode with a static method checkAnagram which accepts the two strings. The return type is boolean which is TRUE / FALSE. Create a Class Main which would be used to accept the two strings and call the static method present in UserMainCode.

**Input and Output Format:** Input consists of two strings. Output consists of TRUE / FALSE. Refer sample output for formatting specifications. Sample Input 1: tea eat Sample Output 1: TRUE

Sample Input 2: Desperation A Rope Ends It Sample Output 2: TRUE

### **MAIN:**

```
import java.util.*;  
  
public class Main {  
  
    public static void main(String[] args){  
  
        Scanner s=new Scanner(System.in);  
  
        String s1=s.nextLine();  
  
        String s2=s.nextLine();  
  
        if(s1.equals(s2))  
            System.out.println("TRUE");  
        else  
            System.out.println("FALSE");  
    }  
}
```

```
        System.out.println(UserMainCode.checkAnagram(s1,s2));  
  
    }  
  
}
```

### **USERMAINCODE:**

```
import java.util.*;  
  
import java.text.*;  
  
public class UserMainCode {  
  
    public static boolean checkAnagram(String s1,String s2)  
  
    {  
  
        boolean b=false;  
  
  
        String aj1 =s1.toLowerCase(); //ANAGRAMS  
        String aj2=s2.toLowerCase();  
  
  
        ArrayList<Character> a1 = new ArrayList<Character>();  
        ArrayList<Character> a2 = new ArrayList<Character>();  
  
        for(int i=0;i<aj1.length();i++)  
  
        {  
  
            char c=aj1.charAt(i);  
  
            if(c!=' ')  
  
            {
```

```

    a1.add(c);

}

}

for(int j=0;j<aj2.length();j++)

{
    char c=aj2.charAt(j);

    if(c!=' ')
    {
        a2.add(c);
    }
}

if(a1.size()==a2.size())

{
    if(a1.containsAll(a2))

    {
        b= true;
    }
}
}

return b;
}
}

```

Write a program to read a integer array of scores, and return a version of the given array where all the 5's have been removed. The remaining elements should shift left towards the start of the array as needed,

and the empty spaces at the end of the array should be filled with 0.

So {1, 5, 5, 2} yields {1, 2, 0, 0}.

Include a class UserMainCode with a static method shiftLeft which accepts the integer array. The return type is modified array.

Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.

Input and Output Format:

Input consists of an integer n which is the number of elements followed by n integer values.

Output consists of modified array.

Refer sample output for formatting specifications.

Sample Input 1: 7 1 5 2 4 5 3 5

Sample Output 1: 1 2 4 3 0 0 0

### **MAIN:**

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args)

    {

        Scanner sc=new Scanner(System.in);

        int size=sc.nextInt();

        int[]m=new int[size];

        int[]n=new int[size];

        for(int i=0;i<size;i++)

        {
```

```

n[i]=sc.nextInt();

}

m=UserMainCode.shiftLeft(n);

for(int i=0;i<size;i++)

{

    System.out.println(m[i]);

}

}

}

```

### **USERMAINCODE:**

```

public class UserMainCode {

    public static int[] shiftLeft(int n[])

    {

        int j=0;

        int[]m=new int[n.length];

        for(int i=0;i<n.length;i++)

        {

            if(n[i]!=5)

            {

                m[j]=n[i];

                j++;

            }

        }

        return m;
    }
}

```

```
 }  
 }
```

## 32.Word Count

Given a string array (s) with each element in the array containing alphabets or digits. Write a program to add all the digits in every string and return the sum as an integer. If two digits appear simultaneously do not consider it as one number. Ex- For 'Hyderabad 21' consider 2 and 1 as two digits instead of 21 as a number.

Include a class UserMainCode with a static method sumOfDigits which accepts the string array. The return type is the integer formed based on rules. Create a Class Main which would be used to accept the string and integer and call the static method present in UserMainCode.

**Input and Output Format:** Input consists of a an integer indicating the number of elements in the string array. Output consists of a integer . Refer sample output for formatting specifications.

Sample Input 1: 5 AAA1

B2B 4CCC A5 ABCDE Sample Output 1: 12

Sample Input 2: 3 12 C23 5CR2 Sample Output 2: 15

### **MAIN:**

```
import java.util.Scanner;  
  
public class Main {  
  
    public static void main(String[] args)  
    {  
  
        Scanner s=new Scanner(System.in);  
  
        int n=s.nextInt();  
  
        String a[]=new String[n];  
  
        for(int i=0;i<n;i++)  
        {
```

```

        a[i]=s.nextInt();

    }

    System.out.println(UserMainCode.sumOfDigits(a));

}

}

```

### **USERMAINCODE:**

```

public class UserMainCode {

    public static int sumOfDigits(String[] s1)

    {

        int sum = 0;

        for(int i=0;i<s1.length;i++)

        {

            String s = s1[i];

            for(int j = 0;j<s.length();j++)

            {

                Character c = s.charAt(j);

                if(Character.isDigit(c))

                {

                    sum+=Integer.parseInt(s.valueOf(c));

                }

            }

            return sum;

        }

    }

```

### 33.Prefix Finder

Given a string array (s) with each element in the array containing 0s and 1s. Write a program to get the number of strings in the array where one String is getting as prefixed in other String in that array . Example 1: Input: {10,101010,10001,1111} Output =2 (Since 10 is a prefix of 101010 and 10001) Example 2: Input: {010,1010,01,0111,10,10} Output =3(01 is a prefix of 010 and 0111. Also, 10 is a prefix of 1010) Note: 10 is NOT a prefix for 10.

Include a class UserMainCode with a static method findPrefix which accepts the string array. The return type is the integer formed based on rules. Create a Class Main which would be used to accept the string and integer and call the static method present in UserMainCode.

**Input and Output Format:** Input consists of a an integer indicating the number of elements in the string array followed by the array. Output consists of a integer . Refer sample output for formatting specifications.

Sample Input 1: 4 0 1 11 110 Sample Output 1: 3

#### **MAIN:**

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args)

    {

        Scanner sc = new Scanner(System.in);

        int n=Integer.parseInt(sc.nextLine());

        String s[]={};

        for(int i=0;i<n;i++)

            s[i]=sc.nextLine();

        System.out.println(UserMainCode.findPrefix(s));

    }

}
```

#### **USERMAINCODE:**

```
import java.util.ArrayList;

import java.util.Iterator;
```

```
import java.util.LinkedHashSet;
public class UserMainCode
{
    public static int findPrefix (String s[]) {
        LinkedHashSet<String>l1=new LinkedHashSet<String>();
        ArrayList<String>a1=new ArrayList<String>();
        int c=0;
        for(int i=0;i<s.length;i++)
            l1.add(s[i]);
        Iterator<String> it=l1.iterator();
        while(it.hasNext())
        {
            a1.add(it.next());
        }
        for(int i=0;i<a1.size();i++)
        {
            String s2=a1.get(i);
            for(int j=0;j<a1.size();j++)
            {
                String s3=a1.get(j);
                if(i!=j&&s3.length()>s2.length())
                {
                    String s4=s3.substring(0,s2.length());
                    if(s2.equals(s4))
                        c++;
                }
            }
        }
    }
}
```

```

        }
    }
}

return c;
}
}

```

### 34.Commons

Given two arrays of strings, return the count of strings which is common in both arrays. Duplicate entries are counted only once. Include a class UserMainCode with a static method countCommonStrings which accepts the string arrays. The return type is the integer formed based on rules. Create a Class Main which would be used to accept the string arrays and integer and call the static method present in UserMainCode.

**Input and Output Format:** Input consists of a an integer indicating the number of elements in the string array followed by the array. Output consists of a integer . Refer sample output for formatting specifications.

Sample Input 1: 3 a c e 3 b d e Sample Output 1: 1

Sample Input 2: 5 ba ba black sheep wool 5 ba ba have any wool Sample Output 2: 2

### **MAIN:**

```

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n1 = sc.nextInt();
        String[] s1 = new String[n1];
        for (int i = 0; i < n1; i++) {
            s1[i] = sc.next();
        }
        int count = 0;
        for (int i = 0; i < n1; i++) {
            for (int j = i + 1; j < n1; j++) {
                if (s1[i].equals(s1[j])) {
                    count++;
                }
            }
        }
        System.out.println(count);
    }
}

```

```

    }

    int n2 = sc.nextInt();

    String[] s2 = new String[n2];

    for (int i = 0; i < n2; i++) {

        s2[i] = sc.next();

    }

    System.out.println(UserMainCode.countCommonStrings(s1,s2));

}

}

```

### **USERMAINCODE:**

```

import java.util.ArrayList;

public class UserMainCode {

    public static int countCommonStrings(String[] s1,String[] s2)

    {

        int count=0;

        ArrayList<String> al = new ArrayList<String>();

        for (int i = 0; i < s1.length; i++) {

            for (int j = 0; j < s2.length; j++) {

                if(s1[i].equals(s2[j])){

                    if(!al.contains(s1[i])){

                        count++;

                    }

                    al.add(s1[i]);

                }

            }

        }

        return count;

    }

}

```

```

    }

    }

    }

    }

    return count;

}

}

```

### 35.Sequence Sum

Write a program to read a non-negative integer n, and find sum of fibonanci series for n number..

Include a class UserMainCode with a static method getFibonacciSum which accepts the integer value. The return type is integer.

The fibonacci seqence is a famous bit of mathematics, and it happens to have a recursive definition.

The first two values in the sequnce are 0 and 1.

Each subsequent value is the sum of the previous two values, so the whole seqence is 0,1,1,2,3,5 and so on.

You will have to find the sum of the numbers of the Fibonaaci series for a given int n.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of a integer.

Output consists of integer.

Refer sample output for formatting specifications.

**Sample Input 1:**

5

**Sample Output 1:**

**MAIN:**

```

import java.util.Scanner;

public class Main {

    public static void main(String[] args)
    {
        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        System.out.println(UserMainCode.getFibonacciSum(n));
    }
}

```

**USERMAINCODE:**

```

import java.util.ArrayList;
import java.util.Scanner;

public class UserMainCode {

    public static int getFibonacciSum(int n){
        int a=0,b=1,c=0,d=1;
        for(int i=3;i<=n;i++){
            c=a+b;
            a=b;
            b=c;
        }
        return c;
    }
}

```

```

d=d+c;

}

return d;

}

}

```

### 36.E-Mail Validation

Write a program to read a string and validate the given email-id as input. Validation Rules: 1. Ensure that there are atleast 5 characters between '@' and '.' 2. There should be only one '.' and one '@' symbol. 3. The '.' should be after the '@' symbol. 4. There must be atleast three characters before '@'. 5. The string after '.' should only be 'com'

Include a class UserMainCode with a static method ValidateEmail which accepts the string. The return type is TRUE / FALSE as per problem. Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

**Input and Output Format:** Input consists of a string. Output consists of TRUE / FALSE. Refer sample output for formatting specifications.

Sample Input 1: test@gmail.com Sample Output 1: TRUE

Sample Input 2: academy@xyz.com Sample Output 2: FALSE

#### **MAIN:**

```

import java.util.*;

public class Main {

public static void main(String[] args) {

Scanner s=new Scanner(System.in);

String ip;

ip=s.nextLine();

boolean b=UserMainCode.ValidateEmail(ip);

if(b==true)

System.out.println("TRUE");

else

```

```
        System.out.println("FALSE");  
    }  
}
```

### USERMAINCODE:

```
import java.util.StringTokenizer;  
  
public class UserMainCode {  
  
    public static boolean ValidateEmail(String ip) {  
  
        int i=0;  
  
        boolean b=false;  
  
        StringTokenizer t=new StringTokenizer(ip,"@");  
  
        String s1=t.nextToken();  
  
        String s2=t.nextToken();  
  
        StringTokenizer t1=new StringTokenizer(s2,".");  
  
        String s3=t1.nextToken();  
  
        String s4=t1.nextToken();  
  
        if(ip.contains("@") && ip.contains("."))  
  
            i++;  
  
        if(i==1)  
  
            if(s3.length()==5)  
  
                if(s1.length()>=3)  
  
                    if(s4.equals("com"))  
  
                        b=true;  
  
        return b;  
    }  
}
```

## 37.Symmetric Difference

Write a program to read two integer array and calculate the symmetric difference of the two arrays. Finally Sort the array.

Symmetric difference is the difference of A Union B and A Intersection B ie. [ (A U B) - (A ^ B)]

Union operation merges the two arrays and makes sure that common elements appear only once. Intersection operation

includes common elements from both the arrays.

Ex - A={12,24,7,36,14} and B={11,26,7,14}.

A U B ={ 7,11,12,14,24,26,36} and

A ^ B = {7,14}

Symmetric difference of A and B after sorting= [A U B] - [ A ^ B] = {11,12,24,26,36}.

Include a class UserMainCode with a static method **getSymmetricDifference** which accepts the integer array. The return

type is an integer array.

Create a Class Main which would be used to accept the two integer arrays and call the static method present in

UserMainCode.

### **Input and Output Format:**

Input consists of an integer n which is the number of elements followed by n integer values. The same sequence is followed

for the next array.

Output consists of sorted symmetric difference array.

Refer sample output for formatting specifications.

### **Sample Input 1:**

11

5

14

26

3

3

5

3

1

### **Sample Output 1:**

1

11

14

26

### **MAIN:**

```
import java.util.*;

public class Main

{
    public static void main(String[] args)
    {
        int n,m;

        Scanner sin = new Scanner(System.in);
    }
}
```

```

n = sin.nextInt();

int[] a1 = new int[n];

for(int i=0;i<n;i++)

{
    a1[i] = sin.nextInt();
}

m = sin.nextInt();

int[] a2 = new int[m];

for(int i=0;i<m;i++)

{
    a2[i] = sin.nextInt();
}

int[] result = UserMainCode.getSymmetricDifference (a1,a2);

for(int i=0;i<result.length;i++)

System.out.println(result[i]);

}
}

```

## USERMAINCODE:

```

import java.util.*;

public class UserMainCode

{

public static int[] getSymmetricDifference (int[] a1,int[] a2)

{

```

```
//int[] a1 = new int[]{11,5,14,26,3};  
  
//int[] a2 = new int[]{5,3,1};  
  
int[] union,inter,result;  
  
int count=0;  
  
int max = a1.length+a2.length;  
  
ArrayList<Integer> temp = new ArrayList<Integer>(max);  
  
/*union*/  
  
for(int i=0;i<a1.length;i++)  
  
{  
  
if(!temp.contains(a1[i]))  
  
{  
  
++count;  
  
temp.add(a1[i]);  
  
}  
  
}  
  
for(int i=0;i<a2.length;i++)  
  
{  
  
if(!temp.contains(a2[i]))  
  
{  
  
++count;  
  
temp.add(a2[i]);  
  
}  
  
}  
  
union = new int[count];  
  
for(int i=0;i<count;i++)
```

```
{  
union[i] = (int)temp.get(i);  
}  
  
Arrays.sort(union);  
  
/*intersection*/  
  
temp = new ArrayList<Integer>(max);  
  
count =0;  
  
Arrays.sort(a2);  
  
for(int i=0;i<a1.length;i++)  
  
{  
  
if(Arrays.binarySearch(a2,a1[i]) >= 0)  
  
{  
  
++count;  
  
temp.add(a1[i]);  
  
}  
  
}  
  
inter = new int[count];  
  
for(int i=0;i<count;i++)  
  
{  
  
inter[i] = (int)temp.get(i);  
  
}  
  
Arrays.sort(inter);  
  
/*difference */  
  
temp = new ArrayList<Integer>(max);  
  
count =0;
```

```

Arrays.sort(inter);

for(int i=0;i<union.length;i++)

{

if(Arrays.binarySearch(inter,union[i]) < 0)

{

++count;

temp.add(union[i]);

}

}

result = new int[count];

for(int i=0;i<count;i++)

{

result[i] = (int)temp.get(i);

}

Arrays.sort(result);

//System.out.println("resultant array : \n "+Arrays.toString(result));

return result;

}
}

```

## 38.Day of Week

Write a program to read a string containing date in DD/MM/YYYY format and prints the day of the week that date falls on.

Return the day in lowercase letter (Ex: monday)

Include a class UserMainCode with a static method **getDayOfWeek** which accepts the string. The return type is the string.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of a string.

Refer sample output for formatting specifications.

### **Sample Input 1:**

02/04/1985

### **Sample Output 1:**

Tuesday

### **MAIN:**

```
-  
  
import java.text.ParseException;  
  
import java.text.SimpleDateFormat;  
  
import java.util.Date;  
  
import java.util.Scanner;  
  
public class Main {  
  
    public static void main(String[] args) throws ParseException {  
  
        Scanner sc=new Scanner(System.in);
```

```
String s1=sc.nextLine();

System.out.println(UserMainCode.getDayOfWeek(s1));

}

}
```

## USERMAINCODE:

```
import java.text.SimpleDateFormat;

import java.text.ParseException;

import java.util.Date;

public class UserMainCode {

    public static String getDayOfWeek(String s1) throws ParseException

    {

        SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");

        SimpleDateFormat sdf1=new SimpleDateFormat("EEEEEE");

        Date d=sdf.parse(s1);

        String s=sdf1.format(d);

        return s.toLowerCase();

    }

}
```

## **39.Add Time**

Write a program to read two String variables containing time intervals in hh:mm:ss format.  
Add the two time intervals and

return a string in days:hours:minutes:seconds format where DD is number of days.

Hint: Maximum value for hh:mm:ss is 23:59:59

Include a class UserMainCode with a static method **addTime** which accepts the string values.  
The return type is the string.

Create a Class Main which would be used to accept the two string values and call the static  
method present in

UserMainCode.

### **Input and Output Format:**

Input consists of two string.

Output consists of a string.

Refer sample output for formatting specifications.

### **Sample Input 1:**

12:45:30

13:50:45

### **Sample Output 1:**

1:2:36:15

### **Sample Input 2:**

23:59:59

23:59:59

### **Sample Output 2:**

1:23:59:58

## MAIN:

```
-  
  
import java.util.*;  
  
import java.io.IOException;  
  
import java.text.*;  
  
public class Main {  
  
    public static void main(String[] args) throws IOException, ParseException {  
  
        Scanner s = new Scanner(System.in);  
  
        String s1=s.next();  
  
        String s2=s.next();  
  
        System.out.println(UserMainCode.addTime(s1,s2));  
  
    }  
  
}
```

## USERMAINCODE:

```
-  
  
import java.util.*;  
  
import java.io.IOException;  
  
import java.text.*;
```

```

public class UserMainCode {

    public static String addTime(String s1, String s2) throws IOException,
ParseException{

    SimpleDateFormat sdf=new SimpleDateFormat("HH:mm:ss");

    sdf.setTimeZone(TimeZone.getTimeZone("UTC"));

    sdf.setTimeZone(TimeZone.getTimeZone("s1"));

    sdf.setTimeZone(TimeZone.getTimeZone("s2"));

    Date d1=sdf.parse(s1);

    Date d2=sdf.parse(s2);

    long add=d1.getTime()+d2.getTime();

    String s=sdf.format(add);

    Calendar cal=Calendar.getInstance();

    cal.setTime(sdf.parse(s));

    int FindDay=cal.get(Calendar.DAY_OF_MONTH);

    if(FindDay>1)

        FindDay=FindDay-1;

    String op=FindDay+":" +s;

    return op;

}

}

```

## 40.ISBN Validation

Write a program to read a string and validate the given ISBN as input.

Validation Rules:

1. An ISBN (International Standard Book Number) is a ten digit code which uniquely identifies a book.
2. To verify an ISBN you calculate 10 times the first digit, plus 9 times the second digit, plus 8 times the third ..all the way until you add 1 times the last digit.

If the final number leaves no remainder when divided by 11 the code is a valid ISBN.

Example 1:

Input:0201103311

Calculation:  $10*0 + 9*2 + 8*0 + 7*1 + 6*1 + 5*0 + 4*3 + 3*3 + 2*1 + 1*1 = 55$ .

$55 \bmod 11 = 0$

Hence the input is a valid ISBN number

Output: true

Include a class UserMainCode with a static method **validateISBN** which accepts the string.  
The return type is TRUE /

FALSE as per problem.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of TRUE / FALSE.

Refer sample output for formatting specifications.

### **Sample Input 1:**

0201103311

### **Sample Output 1:**

TRUE

### **MAIN:**

-

```
import java.util.*;  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        Scanner s = new Scanner(System.in);  
  
        String ip=s.next();  
  
        System.out.println(UserMainCode.ISBNnumber(ip));  
  
    }  
}
```

### **USERMAINCODE:**

-

```
import java.util.*;  
  
import java.text.*;  
  
public class UserMainCode {  
  
    public static String ISBNnumber(String ip) {
```

```

String b="FALSE";

int sum=0;

for(int i=0,j=ip.length();i<ip.length();i++,j--){
    String s=String.valueOf(ip.charAt(i));
    int n=Integer.parseInt(s);
    sum+=(n*j);
}

//System.out.println(sum);

if(sum%11==0)
    b="TRUE";
}

return b;
}
}

```

## 41.Date Format

Write a program to read two String variables in DD-MM-YYYY. Compare the two dates and return the older date in

'MM/DD/YYYY' format.

Include a class UserMainCode with a static method **findOldDate** which accepts the string values. The return type is the

string.

Create a Class Main which would be used to accept the two string values and call the static method present in

UserMainCode.

## **Input and Output Format:**

Input consists of two string.

Output consists of a string.

Refer sample output for formatting specifications.

### **Sample Input 1:**

05-12-1987

8-11-2010

### **Sample Output 1:**

12/05/1987

### **MAIN:**

```
-  
  
import java.text.ParseException;  
  
import java.text.SimpleDateFormat;  
  
import java.util.*;  
  
public class Main {  
  
  
    public static void main(String[] args) throws ParseException {  
        Scanner s = new Scanner(System.in);  
  
        String s1=s.next();  
  
        String s2=s.next();  
  
        System.out.println(UserMainCode.findOldDate(s1,s2));  
    }  
}
```

## USERMAINCODE:

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

public class UserMainCode {

    public static String findOldDate(String s1, String s2) throws ParseException
    {
        SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
        SimpleDateFormat sdf1=new SimpleDateFormat("MM-dd-yyyy");
        Date d1=sdf.parse(s1);
        Date d2=sdf.parse(s2);
        Calendar cal=Calendar.getInstance();
        cal.setTime(d1);
        long y=cal.getTimeInMillis();
        cal.setTime(d2);
        long y1=cal.getTimeInMillis();
        String s3=sdf1.format(d1);
        String s4=sdf1.format(d2);
        if(y<y1)
            return s3;
    }
}
```

```
    else  
  
        return s4;  
  
    }  
  
}  
  
-  
-
```

## 42. Interest Calculation

Write a program to calculate amount of the account holders based on the below mentioned prototype:

1. Read account details from the User. The details would include id, DOB (date of birth) and amount in the given order. The datatype for id is string, DOB is string and amount is integer.
2. You decide to build two hashmaps. The first hashmap contains employee id as key and DOB as value, and the second hashmap contains same employee ids as key and amount as value.
3. Rate of interest as on 01/01/2015:
  - a. If the age greater than or equal to 60 then interest rate is 10% of Amount.
  - b. If the age less than or equal to 60 and greater than or equal to 30 then interest rate is 7% of Amount.
  - v. If the age less than 30 interest rate is 4% of Amount.
4. Revised Amount = principle Amount + interest rate.
5. You decide to write a function **calculateInterestRate** which takes the above hashmaps as input and returns the treemap as output. Include this function in class UserMainCode.

Create a Class Main which would be used to read employee details in step 1 and build the two hashmaps. Call the static

method present in UserMainCode.

**Input and Output Format:**

Input consists of account details. The first number indicates the size of the account. The next three values indicate the user id, DOB and amount. The Employee DOB format is “dd-mm-yyyy”

Output consists of the user id and the amount for each user one in a line.

Refer sample output for formatting specifications.

**Sample Input 1:**

4

SBI-1010

20-01-1987

10000

SBI-1011

03-08-1980

15000

SBI-1012

05-11-1975

20000

SBI-1013

02-12-1950

30000

**Sample Output 1:**

SBI-1010:10400

SBI-1011:16050

SBI-1012:21400

SBI-1013:33000

## MAIN

```
-  
  
import java.util.HashMap;  
  
import java.util.Iterator;  
  
import java.util.Scanner;  
  
import java.util.TreeMap;  
  
public class Main {  
  
    public static void main(String []args){  
  
        Scanner sc=new Scanner(System.in);  
  
        int s=Integer.parseInt(sc.nextLine());  
  
        HashMap<String, String> hm=new HashMap<String, String>();  
  
        HashMap<String, Integer> hm1=new HashMap<String, Integer>();  
  
        for(int i=0;i<s;i++)  
  
        {  
  
            String id=sc.nextLine();  
  
            hm.put(id, sc.nextLine());  
  
            hm1.put(id, Integer.parseInt(sc.nextLine()));  
  
        }  
  
        TreeMap<String, Integer> tm=new TreeMap<String, Integer>();  
  
        tm=UserMainCode.calculateInterestRate(hm,hm1);  
  
        Iterator<String> it=tm.keySet().iterator();  
  
        while(it.hasNext())  
  
        {  
  
            String n=it.next();  
  
            int fac=tm.get(n);  
  
        }  
}
```

```
System.out.println(n+":"+fac);

}

}

}
```

## USERMAINCODE

```
import java.text.DecimalFormat;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.HashMap;

import java.util.Iterator;

import java.util.HashMap;

import java.util.TreeMap;

public class UserMainCode

{

    public static TreeMap<String, Integer> calculateInterestRate

(HashMap<String, String>hm,HashMap<String, Integer>hm1)

    {

        int year=0,amount=0;

        double dis=0;

        String now="01/01/2015";

        TreeMap<String, Integer>tm=new TreeMap<String, Integer>();

        Iterator<String> it=hm.keySet().iterator();
```

```
while(it.hasNext())
{
String id=it.next();
String dor=hm.get(id);
amount=hm1.get(id);

SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");
SimpleDateFormat sdf1=new SimpleDateFormat("dd/MM/yyyy");

try
{
Date d=sdf.parse(dor);
Date d1=sdf1.parse(now);
sdf.setLenient(false);

int y=d.getYear();
int y1=d1.getYear();

int m=d.getMonth();
int m1=d1.getMonth();

int day=d.getDay();
int day1=d1.getDay();

year=y1-y;

if(m>m1)
year--;
else if(m==m1)
{if(day<day1)
year--;
}
}
```

```
if(year>=60)

dis=0.1*amount+amount;

else if(year<60 && year>=30 )

dis=0.07*amount+amount;

else

dis=0.04*amount+amount;

tm.put(id,(int)dis);

}

catch(Exception e)

{

e.printStackTrace();

}

}

return tm;
```

## 43. Discount Rate Calculation

Write a program to calculate discount of the account holders based on the transaction amount and registration date using below mentioned prototype:

1. Read account details from the User. The details would include id, DOR (date of registration) and transaction amount in the

given order. The datatype for id is string, DOR is string and transaction amount is integer.

2. You decide to build two hashmaps. The first hashmap contains employee id as key and DOR as value, and the second hashmap contains same employee ids as key and amount as value.

3. Discount Amount as on 01/01/2015:

a. If the transaction amount greater than or equal to 20000 and registration greater than or equal to 5 year then discount rate is 20% of transaction amount.

b. If the transaction amount greater than or equal to 20000 and registration less than to 5 year then discount rate is 10% of transaction amount.

c. If the transaction amount less than to 20000 and registration greater than or equal to 5 year then discount rate is 15% of transaction amount.

d. If the transaction amount less than to 20000 and registration less than to 5 year then discount rate is 5% of transaction amount.

4. You decide to write a function **calculateDiscount** which takes the above hashmaps as input and returns the treemap as output. Include this function in class UserMainCode.

Create a Class Main which would be used to read employee details in step 1 and build the two hashmaps. Call the static method present in UserMainCode.

#### **Input and Output Format:**

Input consists of transaction details. The first number indicates the size of the employees. The next three values indicate the user id, user DOR and transaction amount. The DOR (Date of Registration) format is “dd-mm-yyyy”

Output consists of a string which has the user id and discount amount one in a line for each user.

Refer sample output for formatting specifications.

#### **Sample Input 1:**

4

A-1010

20-11-2007

25000

B-1011

04-12-2010

30000

C-1012

11-11-2005

15000

D-1013

02-12-2012

10000

**Sample Output 1:**

A-1010:5000

B-1011:3000

C-1012:2250

D-1013:500

**MAIN:**

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.TreeMap;
import java.util.Scanner;

public class Main{
    public static void main(String []args){
        Scanner sc=new Scanner(System.in);
        int s=Integer.parseInt(sc.nextLine());
        HashMap<String, String> hm=new HashMap<String, String>();
        HashMap<String, Integer> hm1=new HashMap<String, Integer>();
        for(int i=0;i<s;i++)
            hm.put(sc.nextLine(), hm1.get(sc.nextLine()));
        hm1.put(sc.nextLine(), s);
    }
}
```

```

{
String id=sc.nextLine();

hm.put(id, sc.nextLine());

hm1.put(id,Integer.parseInt(sc.nextLine()));

}

TreeMap<String, Integer>tm=new TreeMap<String, Integer>();

tm=UserMainCode.calculateDiscount(hm,hm1);

Iterator<String> it=tm.keySet().iterator();

while(it.hasNext())

{

String n=it.next();

int fac=tm.get(n);

System.out.println(n+":"+fac);

}

}

}

```

### USERMAINCODE

```

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.HashMap;

import java.util.*;

public class UserMainCode

{

public static TreeMap<String, Integer> calculateDiscount

(HashMap<String, String>hm,HashMap<String, Integer>hm1)

```

```
{  
  
int amount=0;  
  
double dis=0;  
  
String now="01/01/2015";  
  
TreeMap<String, Integer>tm=new TreeMap<String, Integer>();  
  
Iterator<String> it=hm.keySet().iterator();  
  
while(it.hasNext())  
  
{  
  
String id=it.next();  
  
String dor=hm.get(id);  
  
amount=hm1.get(id);  
  
SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");  
  
sdf.setLenient(false);  
  
try{  
  
Date d=new Date();  
  
Date d1=new Date();  
  
String a=dor;  
  
String b="01-01-2014";  
  
d=sdf.parse(a);  
  
d1=sdf.parse(b);  
  
long t=d.getTime();  
  
long t1=d1.getTime();  
  
long t3=t1-t;  
  
long l1=(24 * 60 * 60 * 1000);  
  
long l=l1*365;
```

```
long year1=t3/l;

//System.out.println("Result="+year1);

if(year1>=5 && amount>=20000)
dis=0.2*amount;

else if(year1<5 && amount>=20000)
dis=0.1*amount;

else if(year1>=5 && amount<20000)
dis=0.15*amount;

else
dis=0.05*amount;

tm.put(id,(int)dis);

}

catch(Exception e)
{
e.printStackTrace();
}

}

return tm;
}
```



## ZeeZee Bank

Account.java

```
public class Account {  
    private long accountNumber;  
    private double balanceAmount;  
  
    public Account(long accountNumber, double balanceAmount) {  
        this.accountNumber = accountNumber;  
        this.balanceAmount = balanceAmount;  
    }  
  
    public long getAccountNumber() {  
        return accountNumber;  
    }  
  
    public void setAccountNumber(long accountNumber) {  
        this.accountNumber = accountNumber;  
    }  
  
    public double getBalanceAmount() {  
        return balanceAmount;  
    }  
  
    public void setBalanceAmount(double balanceAmount) {  
        this.balanceAmount = balanceAmount;  
    }  
  
    public void deposit(double depositAmount) {  
        balanceAmount += depositAmount;  
    }  
  
    public boolean withdraw(double withdrawAmount) {  
        if (withdrawAmount <= balanceAmount) {  
            balanceAmount -= withdrawAmount;  
            return true;  
        }  
        return false;  
    }  
}
```

Main.java

```
import java.text.DecimalFormat;
```

```

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        DecimalFormat decimalFormat = new DecimalFormat("0.00");

        System.out.println("Enter the account number:");
        long accountNumber = scanner.nextLong();

        System.out.println("Enter initial balance:");
        double balanceAmount = scanner.nextDouble();

        Account account = new Account(accountNumber, balanceAmount);

        System.out.println("Enter the amount to be deposited:");
        double depositAmount = scanner.nextDouble();
        account.deposit(depositAmount);
        double availableBalance = account.getBalanceAmount();

        System.out.println("Available balance is:" + decimalFormat.format(availableBalance));

        System.out.println("Enter the amount to be withdrawn:");
        double withdrawAmount = scanner.nextDouble();
        boolean isWithdrawn = account.withdraw(withdrawAmount);
        availableBalance = account.getBalanceAmount();

        if (!isWithdrawn) {
            System.out.println("Insufficient balance");
        }

        System.out.println("Available balance is:" + decimalFormat.format(availableBalance));
    }
}

```

### Numerology number

```

Main.java
import java.util.Scanner;

public class Main {
    private static int getSum(long num) {
        char[] chars = Long.toString(num).toCharArray();

```

```
int sum = 0;

for (char ch : chars) {
    sum += Character.digit(ch, 10);
}

return sum;
}

private static int getNumerology(long num) {
    String string = String.valueOf(num);

    while (string.length() != 1) {
        string = String.valueOf(getSum(Long.parseLong(string)));
    }

    return Integer.parseInt(string);
}

private static int getOddCount(long num) {
    int oddCount = 0;

    for (char ch : Long.toString(num).toCharArray()) {
        if (Character.digit(ch, 10) % 2 != 0) {
            ++oddCount;
        }
    }

    return oddCount;
}

private static int getEvenCount(long num) {
    int evenCount = 0;

    for (char ch : Long.toString(num).toCharArray()) {
        if (Character.digit(ch, 10) % 2 == 0) {
            ++evenCount;
        }
    }

    return evenCount;
}

public static void main(String[] args) {
```

```

Scanner scanner = new Scanner(System.in);

System.out.println("Enter the number");
long num = scanner.nextLong();

System.out.println("Sum of digits");
System.out.println(getSum(num));

System.out.println("Numerology number");
System.out.println(getNumerology(num));

System.out.println("Number of odd numbers");
System.out.println(getOddCount(num));

System.out.println("Number of even numbers");
System.out.println(getEvenCount(num));
}
}

```

### **Substitution Cipher Technique**

```

Main.java
import java.util.*;

public class Main {
    public static void main(String[] args) {
        StringBuilder stringBuilder = new StringBuilder();
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter the encrypted text:");
        String text = scanner.nextLine();
        char[] chars = text.toCharArray();
        boolean flag = false;

        for (char ch : chars) {
            if (Character.isLetter(ch)) {
                flag = true;

                if (Character.isLowerCase(ch)) {
                    int sub = (int) ch - 7;

                    if (sub < 97) {
                        ch = (char) (122 - (97 - sub) + 1);
                    } else {

```

```

        ch = (char) sub;
    }
} else if (Character.isUpperCase(ch)) {
    int sub = (int) ch - 7;

    if (sub < 65) {
        ch = (char) (90 - (65 - sub) + 1);
    } else {
        ch = (char) sub;
    }
}

stringBuilder.append(ch);
} else if (Character.isWhitespace(ch)) {
    stringBuilder.append(ch);
}
}

if (flag) {
    System.out.println("Decrypted text:");
    System.out.println(stringBuilder.toString());
} else {
    System.out.println("No hidden message");
}
}
}
}

```

### Bank Account - Interface

Account.java

```

public class Account {
    private String accountNumber;
    private String customerName;
    private double balance;

    public Account(String accountNumber, String customerName, double balance) {
        this.accountNumber = accountNumber;
        this.customerName = customerName;
        this.balance = balance;
    }

    public String getAccountNumber() {
        return accountNumber;
    }
}

```

```
public void setAccountNumber(String accountNumber) {
    this.accountNumber = accountNumber;
}

public String getCustomerName() {
    return customerName;
}

public void setCustomerName(String customerName) {
    this.customerName = customerName;
}

public double getBalance() {
    return balance;
}

public void setBalance(double balance) {
    this.balance = balance;
}
}
```

#### CurrentAccount.java

```
public class CurrentAccount extends Account implements MaintenanceCharge {
    public CurrentAccount(String accountNumber, String customerName, double balance) {
        super(accountNumber, customerName, balance);
    }

    @Override
    public float calculateMaintenanceCharge(float noOfYears) {
        return (100.0f + noOfYears) + 200.0f;
    }
}
```

#### MaintenanceCharge.java

```
public interface MaintenanceCharge {
    float calculateMaintenanceCharge(float noOfYears);
}
```

#### SavingsAccount.java

```
public class SavingsAccount extends Account implements MaintenanceCharge {
    public SavingsAccount(String accountNumber, String customerName, double balance) {
        super(accountNumber, customerName, balance);
    }
}
```

```
@Override
public float calculateMaintenanceCharge(float noOfYears) {
    return (50.0f * noOfYears) + 50.0f;
}
}

UserInterface.java
import java.text.DecimalFormat;
import java.util.Scanner;

public class UserInterface {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        DecimalFormat decimalFormat = new DecimalFormat("0.0");

        System.out.println("1. Savings Account");
        System.out.println("2. Current Account");
        System.out.println("Enter your choice:");
        int choice = scanner.nextInt();

        System.out.println("Enter the Account number");
        String accountNumber = scanner.next();

        System.out.println("Enter the Customer Name");
        String customerName = scanner.next();

        System.out.println("Enter the Balance amount");
        double balance = scanner.nextDouble();

        System.out.println("Enter the number of years");
        int noOfYears = scanner.nextInt();

        System.out.println("Customer Name " + customerName);
        System.out.println("Account Number " + accountNumber);
        System.out.println("Account Balance " + decimalFormat.format(balance));

        switch (choice) {
            case 1: {
                SavingsAccount savingsAccount = new SavingsAccount(accountNumber,
customerName, balance);
                System.out.println("Maintenance Charge for Savings Account is Rs " +
decimalFormat.format(savingsAccount.calculateMaintenanceCharge(noOfYears)));
                break;
            }
        }
    }
}
```

```

        case 2: {
            CurrentAccount currentAccount = new CurrentAccount(accountNumber,
customerName, balance);
            System.out.println("Maintenance Charge for Current Account is Rs " +
decimalFormat.format(currentAccount.calculateMaintenanceCharge(noOfYears)));
        }
    }
}
}

```

## Batting Average

```

UserInterface.java
package com.ui;

import com.utility.Player;

import java.util.ArrayList;
import java.util.Scanner;

public class UserInterface {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Player player = new Player();
        player.setScoreList(new ArrayList<>());
        boolean flag = true;

        while (flag) {
            System.out.println("1. Add Runs Scored");
            System.out.println("2. Calculate average runs scored");
            System.out.println("3. Exit");
            System.out.println("Enter your choice");
            int choice = scanner.nextInt();

            switch (choice) {
                case 1: {
                    System.out.println("Enter the runs scored");
                    int score = scanner.nextInt();
                    player.addScoreDetails(score);
                    break;
                }
                case 2: {
                    System.out.println("Average runs secured");
                }
            }
        }
    }
}

```

```
        System.out.println(player.getAverageRunScored());
        break;
    }
    case 3: {
        System.out.println("Thank you for use the application");
        flag = false;
        break;
    }
}
}
}
}
```

Player.java

```
package com.utility;
```

```
import java.util.List;
```

```
public class Player {
```

```
    private List<Integer> scoreList;
```

```
    public List<Integer> getScoreList() {
```

```
        return scoreList;
```

```
}
```

```
    public void setScoreList(List<Integer> scoreList) {
```

```
        this.scoreList = scoreList;
```

```
}
```

```
    public double getAverageRunScored() {
```

```
        if (scoreList.isEmpty()) {
```

```
            return 0.0;
```

```
}
```

```
        int size = scoreList.size();
```

```
        int totalScore = 0;
```

```
        for (int score : scoreList) {
```

```
            totalScore += score;
```

```
}
```

```
        return (double) totalScore / (double) size;
```

```
}
```

```
    public void addScoreDetails(int score) {  
        scoreList.add(score);  
    }  
}
```

### Grade Calculation

```
Main.java  
import java.util.Scanner;  
  
public class Main {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
  
        System.out.println("Enter the number of Threads:");  
        int n = scanner.nextInt();  
  
        GradeCalculator[] gradeCalculators = new GradeCalculator[n];  
        Thread[] threads = new Thread[n];  
  
        for (int i = 0; i < n; ++i) {  
            System.out.println("Enter the String:");  
            String string = scanner.next();  
            String[] strings = string.split(":");  
            int[] marks = new int[5];  
  
            String studName = strings[0];  
  
            for (int j = 1; j < 6; ++j) {  
                marks[j - 1] = Integer.parseInt(strings[j]);  
            }  
  
            gradeCalculators[i] = new GradeCalculator(studName, marks);  
            threads[i] = new Thread(gradeCalculators[i]);  
            threads[i].start();  
            threads[i].interrupt();  
        }  
  
        for (int i = 0; i < n; ++i) {  
            System.out.println(gradeCalculators[i].getStudName() + ":" +  
gradeCalculators[i].getResult());  
        }  
    }  
}
```

```
Gradecalculator.java
public class GradeCalculator extends Thread {
    private String studName;
    private char result;
    private int[] marks;

    public GradeCalculator(String studName, int[] marks) {
        this.studName = studName;
        this.marks = marks;
    }

    public String getStudName() {
        return studName;
    }

    public void setStudName(String studName) {
        this.studName = studName;
    }

    public char getResult() {
        return result;
    }

    public void setResult(char result) {
        this.result = result;
    }

    public int[] getMarks() {
        return marks;
    }

    public void setMarks(int[] marks) {
        this.marks = marks;
    }

    @Override
    public void run() {
        int totalMarks = 0;

        for (int mark : marks) {
            totalMarks += mark;
        }
    }
}
```

```

if (totalMarks <= 500 && totalMarks >= 400) {
    result = 'A';
} else if (totalMarks < 400 && totalMarks >= 300) {
    result = 'B';
} else if (totalMarks < 300 && totalMarks >= 200) {
    result = 'C';
} else if (totalMarks < 200 && totalMarks >= 0) {
    result = 'E';
}
}
}

```

### **Employees eligible for promotionCoding exercise**

Main.java

```

import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Scanner;
import java.util.stream.Collectors;
import java.util.stream.IntStream;

class Employee implements Comparable<Employee> {
    private final String id;
    private final LocalDate joiningDate;
    private boolean isEligible;

    public Employee(String id, LocalDate joiningDate) {
        this.id = id;
        this.joiningDate = joiningDate;
    }

    public void setEligible(LocalDate now) {
        isEligible = joiningDate.until(now, ChronoUnit.YEARS) >= 5;
    }

    public boolean getEligible() {
        return isEligible;
    }

    public String getId() {
        return id;
    }
}

```

```

}

@Override
public String toString() {
    return id;
}

@Override
public int compareTo(Employee employee) {
    return this.id.compareTo(employee.getId());
}
}

public class Main {
    public static void main(String[] args) throws Exception {
        Scanner scanner = new Scanner(System.in);
        DateTimeFormatter dateTimeFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
        LocalDate now = LocalDate.parse("01/01/2019", dateTimeFormatter);
        int n = scanner.nextInt();
        ArrayList<Employee> employees = new ArrayList<>();

        IntStream.rangeClosed(1, 4).forEach(i -> {
            String id = scanner.next();
            String joiningDateStr = scanner.next();

            try {
                LocalDate joiningDate = LocalDate.parse(joiningDateStr, dateTimeFormatter);
                Employee employee = new Employee(id, joiningDate);
                employee.setIsEligible(now);
                employees.add(employee);
            } catch (Exception ignore) {
                System.out.println("Invalid date format");
                System.exit(0);
            }
        });

        List<Employee> filteredEmployees =
        employees.stream().filter(Employee::getIsEligible).collect(Collectors.toList());

        if (filteredEmployees.isEmpty()) {
            System.out.println("No one is eligible");
        } else {
            Collections.sort(filteredEmployees);
            filteredEmployees.forEach(System.out::println);
        }
    }
}

```

```
    }
}
}
```

### Check Number Type

NumberType.java

```
public interface NumberType {
    boolean checkNumber(int num);
}
```

NumberTypeUtility.java

```
import java.util.Scanner;

public class NumberTypeUtility {
    public static NumberType idOdd() {
        return (num) -> num % 2 != 0;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int num = scanner.nextInt();

        if (idOdd().checkNumber(num)) {
            System.out.println(num + " is odd");
        } else {
            System.out.println(num + " is not odd");
        }
    }
}
```

### Retrieve Flight details based on source and destination

Main.java

```
import java.util.*;
public class Main{
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the source");
        sc.next();
        String source=sc.nextLine();
        System.out.println("Enter the destination");
        String dest=sc.nextLine();
        FlightManagementSystem obj=new FlightManagementSystem();
```

```

ArrayList<Flight> res=obj.viewFlightsBySourceDestination(source,dest);
if(res!=null)
    System.out.println(res);
else
    System.out.println("No flights available for the given source and destination");
}
}

```

```

DB.java
import java.io.FileInputStream;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;

public class DB {

    private static Connection con = null;
    private static Properties props = new Properties();

    //ENSURE YOU DON'T CHANGE THE BELOW CODE WHEN YOU SUBMIT
    public static Connection getConnection() throws ClassNotFoundException,
SQLException {
        try{

            FileInputStream fis = null;
            fis = new FileInputStream("database.properties");
            props.load(fis);

            // load the Driver Class
            Class.forName(props.getProperty("DB_DRIVER_CLASS"));

            // create the connection now
            con =
DriverManager.getConnection(props.getProperty("DB_URL"),props.getProperty("DB_USERNAME"),
props.getProperty("DB_PASSWORD"));
        }
        catch(IOException e){
            e.printStackTrace();
        }
        return con;
    }
}

```

```
}

FlightManagementSystem.java
import java.util.*;
import java.sql.*;
public class FlightManagementSystem{
    public ArrayList<Flight> viewFlightsBySourceDestination(String source, String destination){
        DB db=new DB();
        ArrayList<Flight> list=new ArrayList<Flight>();
        try{
            int f=0;
            Connection con=db.getConnection();
            Statement st=con.createStatement();
            String sql= "select * from Flight where source= "+source+" and destination=
"+destination+"";
            ResultSet rs=st.executeQuery(sql);
            while(rs.next()){
                f=1;
                Flight x=new Flight(rs.getInt(1), rs.getString(2),rs.getString(3), rs.getInt(4),
rs.getDouble(5));
                list.add(x);
            }
            con.close();
            if(f==1)
                return list;
            else
                return null;
        }
        catch(SQLException e){
            System.out.println("SQL Error. Contact Administrator.");
            return null;
        }
        catch(Exception e){
            System.out.println("Exception. Contact Administrator.");
            return null;
        }
    }
}
Flight.java
```

```
public class Flight {

    private int flightId;
    private String source;
```

```
private String destination;
private int noOfSeats;
private double flightFare;
public int getFlightId() {
    return flightId;
}
public void setFlightId(int flightId) {
    this.flightId = flightId;
}
public String getSource() {
    return source;
}
public void setSource(String source) {
    this.source = source;
}
public String getDestination() {
    return destination;
}
public void setDestination(String destination) {
    this.destination = destination;
}
public int getNoOfSeats() {
    return noOfSeats;
}
public void setNoOfSeats(int noOfSeats) {
    this.noOfSeats = noOfSeats;
}
public double getFlightFare() {
    return flightFare;
}
public void setFlightFare(double flightFare) {
    this.flightFare = flightFare;
}
public Flight(int flightId, String source, String destination,
             int noOfSeats, double flightFare) {
    super();
    this.flightId = flightId;
    this.source = source;
    this.destination = destination;
    this.noOfSeats = noOfSeats;
    this.flightFare = flightFare;
}
public String toString(){
```

```
        return ("Flight ID : "+getFlightId());
    }
```

```
}
```

### Perform Calculation

```
import java.util.Scanner;

public class Calculator {

    public static void main (String[] args) {

        Scanner sc=new Scanner(System.in);

        int a = sc.nextInt();

        int b= sc.nextInt();

        Calculate Perform_addition = performAddition();

        Calculate Perform_subtraction = performSubtraction();

        Calculate Perform_product = performProduct();

        Calculate Perform_division = performDivision();

        System.out.println("The sum is "+Perform_addition.performCalculation(a,b));

        System.out.println("The difference is
"+Perform_subtraction.performCalculation(a,b));

        System.out.println("The product is "+Perform_product.performCalculation(a,b));

        System.out.println("The division value is
"+Perform_division.performCalculation(a,b));

    }

    public static Calculate performAddition(){

        Calculate Perform_calculation = (int a,int b)->a+b;

        return Perform_calculation;
```

```
}

public static Calculate performSubtraction(){

    Calculate Perform_calculation = (int a,int b)->a-b;

    return Perform_calculation;

}

public static Calculate performProduct(){

    Calculate Perform_calculation = (int a,int b)->a*b;

    return Perform_calculation;

}

public static Calculate performDivision(){

    Calculate Perform_calculation = (int a,int b)->{

        float c = (float)a;

        float d = (float)b;

        return (c/d);

    };

    return Perform_calculation;

}

}

public interface Calculate {

    float performCalculation(int a,int b);

}
```

```

public class InPatient extends Patient {
    InPatient(String patientId, String patientname, long mobileNumber,
    String gender) {
        super(patientId, patientname, mobileNumber, gender);
    }
    InPatient() {
    }
    private double roomRent;
    public double getrent()
    {
        return roomRent;
    }
    public void setrent(double rent)
    {
        roomRent=rent;
    }
    public double calculateTotalBill(int no,double medi)
    {
        return (roomRent*no)+medi;
    }
}

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Scanner;

public class Main {
    public static void main(String [] args) throws IOException {
        Scanner sc=new Scanner(System.in);
        BufferedReader br=new BufferedReader(new
        InputStreamReader(System.in));
        OutPatient o1=new OutPatient();
        InPatient o2=new InPatient();
        System.out.println("1.In Patient\n2.Out Patient");
        System.out.println("Enter the choice");
        int a=sc.nextInt();
        //sc.hasNextLine();
        System.out.println("Enter the details\nPatient Id");
        String pid=br.readLine();
        System.out.println("Patient Name");
        String pname=br.readLine();
        System.out.println("Phone Number");
        long mob=sc.nextLong();
        System.out.println("Gender");
        String gen=br.readLine();
        if(a==1)
        {
            System.out.println("Room Rent");
            double rent=sc.nextDouble();
        }
    }
}

```

```

        System.out.println("Medicinal Bill");
        double med=sc.nextDouble();
        System.out.println("Number of Days of Stay");
        int no=sc.nextInt();
        c2.setrent(rent);
        System.out.println("Amount to be paid
"+c2.calculateTotalBill111(no,med));
    }
    else
    {
        System.out.println("Consultancy Fee");
        double con=sc.nextDouble();
        System.out.println("Medicinal Bill");
        double med=sc.nextDouble();
        System.out.println("Scan Pay");
        int scan=sc.nextInt();
        c1.setcon(con);
        System.out.println("Amount to be paid
"+c1.calculateTotalBill111(scan,med));
    }
}
}

public class OutPatient extends Patient {

    /*OutPatient(String patientId, String patientname, long
mobileNumber, String gender) {
    super(patientId, patientname, mobileNumber, gender);
    // TODO Auto-generated constructor stub
}*/
    OutPatient()
    {
        super();
    }
    private double consultingFee;
    public double getcon()
    {
        return consultingFee;
    }
    public void setcon(double con)
    {
        consultingFee=con;
    }
    public double calculateTotalBill111(int scan,double medi)
    {
        return (consultingFee+scan+medi);
    }
}

```

```
public class Patient {  
    private String patientId,patientname,gender;  
    private long mobileNumber;  
    Patient(String patientId, String patientname, long  
    mobileNumber, String gender)  
    {  
        this.patientId=patientId;  
        this.patientname=patientname;  
        this.gender=gender;  
        this.mobileNumber=mobileNumber;  
    }  
    Patient()  
    {  
    }  
    public String getpaid(){  
        return patientId;  
    }  
    public String getpaname(){  
        return patientname;  
    }  
    public String getpagen(){  
        return gender;  
    }  
    public long getpanob(){  
        return mobileNumber;  
    }  
  
    public void setpaid(String id){  
        patientId=id;  
    }  
    public void setpaname(String name){  
        patientname=name;  
    }  
    public void setpagen(String gen){  
        gender=gen;  
    }  
    public void setpanob(long mob){  
        mobileNumber=mob;  
    }  
}
```

### Payment Inheritance

#### **Bill.java**

```
public class Bill {  
  
    public String processPayment(Payment obj) {  
  
        String message = "Payment not done and your due amount is "+obj.getDueAmount();  
    }  
}
```

```

if(obj instanceof Cheque ) {

    Cheque cheque = (Cheque) obj;

    if(cheque.payAmount())

        message = "Payment done successfully via cheque";

}

else if(obj instanceof Cash ) {

    Cash cash = (Cash) obj;

    if(cash.payAmount())

        message = "Payment done successfully via cash";

}

else if(obj instanceof Credit ) {

    Credit card = (Credit) obj;

    if(card.payAmount())

        message = "Payment done successfully via creditcard. Remaining amount in your
"+card.getCardType()+" card is "+card.getCreditCardAmount();

}

return message;
}
}

```

### **Cash.java**

```

public class Cash extends Payment{

    private int cashAmount;

    public int getCashAmount() {

```

```
        return cashAmount;  
    }  
  
    public void setCashAmount(int cashAmount) {  
        this.cashAmount = cashAmount;  
    }  
  
    @Override  
  
    public boolean payAmount() {  
        return getCashAmount() >= getDueAmount();  
    }  
}
```

### **Cheque.java**

```
import java.text.ParseException;  
  
import java.text.SimpleDateFormat;  
  
import java.util.Calendar;  
  
import java.util.Date;  
  
import java.util.GregorianCalendar;  
  
public class Cheque extends Payment {  
  
    private String chequeNo;  
  
    private int chequeAmount;  
  
    private Date dateOfIssue;  
  
    public String getChequeNo() {  
        return chequeNo;  
    }
```

```
public void setChequeNo(String chequeNo) {
    this.chequeNo = chequeNo;
}

public int getChequeAmount() {
    return chequeAmount;
}

public void setChequeAmount(int chequeAmount) {
    this.chequeAmount = chequeAmount;
}

public Date getDateOfIssue() {
    return dateOfIssue;
}

public void setDateOfIssue(Date dateOfIssue) {

    this.dateOfIssue = dateOfIssue;
}

@Override

public boolean payAmount() {
    int months = findDifference(getDateOfIssue());
    return (getChequeAmount() >= getDueAmount() & months <= 6);
}

private int findDifference(Date date) {
    Calendar myDate = new GregorianCalendar();
    myDate.setTime(date);
    return (2020 - myDate.get(Calendar.YEAR)) * 12 + (0-myDate.get(Calendar.MONTH));
}

public void generateDate(String date) {

    try {
        Date issueDate = new SimpleDateFormat("dd-MM-yyyy").parse(date);
        setDateOfIssue(issueDate);
    }
    catch (ParseException e) {
        e.printStackTrace();
    }
}
```

```
    }  
}
```

### Credit.java

```
public class Credit extends Payment {  
  
    private int creditCardNo;  
  
    private String cardType;  
  
    private int creditCardAmount;  
  
    public int getCreditCardNo(){  
  
        return creditCardNo;  
  
    }  
  
    public void setCreditCardNo(int creditCardNo) {  
  
        this.creditCardNo = creditCardNo;  
  
    }  
  
    public String getCardType() {  
  
        return cardType;  
  
    }  
  
    public void setCardType(String cardType) {  
  
        this.cardType = cardType;  
  
    }  
  
    public int getCreditCardAmount() {  
  
        return creditCardAmount;  
  
    }  
  
    public void setCreditCardAmount(int creditCardAmount) {
```

```
this.creditCardAmount = creditCardAmount;

}

@Override

public boolean payAmount() {

int tax = 0;

boolean isDeducted = false;

switch(cardType) {

case "silver":

setCreditCardAmount(10000);

tax = (int) (0.02*getDueAmount())+getDueAmount();

if(tax <= getCreditCardAmount()) {

setCreditCardAmount(getCreditCardAmount()-tax);

isDeducted = true;

}

break;

case "gold":

setCreditCardAmount(50000);

tax = (int) (0.05*getDueAmount())+getDueAmount();

if(tax <= getCreditCardAmount()) {

setCreditCardAmount(getCreditCardAmount()-tax);

isDeducted = true;

}

}
```

```
break;

case "platinum":

setCreditCardAmount(100000);

tax = (int) (0.1*getDueAmount())+getDueAmount();

if(tax <= getCreditCardAmount()) {

setCreditCardAmount(getCreditCardAmount()-tax);

isDeducted = true;

}

break;

}

return isDeducted;

}

}
```

### **Main.java**

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Bill bill = new Bill();

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the due amount:");

        int dueAmount = sc.nextInt();

        System.out.println("Enter the mode of payment(cheque/cash/credit):");
```

```
String mode = sc.next();

switch (mode) {

    case "cash":

        System.out.println("Enter the cash amount:");

        int cashAmount = sc.nextInt();

        Cash cash = new Cash();

        cash.setCashAmount(cashAmount);

        cash.setDueAmount(dueAmount);

        System.out.println(bill.processPayment(cash));

        break;

    case "cheque":

        System.out.println("Enter the cheque number:");

        String number = sc.next();

        System.out.println("Enter the cheque amount:");

        int chequeAmount = sc.nextInt();

        System.out.println("Enter the date of issue:");

        String date = sc.next();

        Cheque cheque = new Cheque();

        cheque.setChequeAmount(chequeAmount);

        cheque.setChequeNo(number);

        cheque.generateDate(date);

        cheque.setDueAmount(dueAmount);
```

```
System.out.println(bill.processPayment(cheque));
break;

case "credit":

System.out.println("Enter the credit card number.");

int creditNumber = sc.nextInt();

System.out.println("Enter the card type(silver,gold,platinum)");

String cardType = sc.next();

Credit credit = new Credit();

credit.setCardType(cardType);

credit.setCreditCardNo(creditNumber);

credit.setDueAmount(dueAmount);

System.out.println(bill.processPayment(credit));

default:

break;

}

sc.close();

}

}
```

### **Payment.java**

```
public class Payment {

    private int dueAmount;

    public int getDueAmount() {

        return dueAmount;
    }
}
```

```

    }

    public void setDueAmount(int dueAmount) {

        this.dueAmount = dueAmount;

    }

    public boolean payAmount() {

        return false;

    }

}

```

## HUNGER EATS

```

package com.utility;
import java.util.*;
import com.bean.FoodProduct;
public class Order{
    private double discountPercentage;
    private List<FoodProduct> foodList=new ArrayList<FoodProduct>();

    public double getDiscountPercentage() {
        return discountPercentage;
    }
    public void setDiscountPercentage(double discountPercentage) {
        this.discountPercentage = discountPercentage;
    }
    public List<FoodProduct> getFoodList() {
        return foodList;
    }
    public void setFoodList(List<FoodProduct> foodList) {
        this.foodList = foodList;
    }

    public void findDiscount(String bankName)
    {
        if(bankName.equals("HDFC")) {
            discountPercentage=15.0;
        }
        else if(bankName.equals("ICICI")) {
    }
}

```

```

        discountPercentage=25.0;
    }
else if(bankName.equals("CUB")) {
    discountPercentage=30.0;
}
else if(bankName.equals("SBI")) {
    discountPercentage=50.0;
}
else if(bankName.equals("OTHERS")) {
    discountPercentage=0.0;
}

}

public void addToCart(FoodProduct foodProductObject)
{
List<FoodProduct> f=getFoodList();
f.add(foodProductObject);
setFoodList(f);

}

public double calculateTotalBill()
{
    double bill = 0;
    List<FoodProduct> f=getFoodList();
    for(int i=0;i<f.size();i++)
    {
//        System.out.println(f.get(i).getCostPerUnit());
//        System.out.println(f.get(i).getQuantity());
        bill+=f.get(i).getQuantity()*f.get(i).getCostPerUnit()*1.0;

    }
//    System.out.println(bill);
//    System.out.println(dis);
    bill=bill-((bill*discountPercentage)/100);
    return bill;
}

}

package com.ui;

import java.util.Scanner;

```

```
import com.utility.Order;
import com.bean.FoodProduct;

public class UserInterface{

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int itemno;
        String bank;

        System.out.println("Enter the number of items");
        itemno=sc.nextInt();
        System.out.println("Enter the item details");

        Order o=new Order();

        for(int i=0;i<itemno;i++)
        {
            FoodProduct fd=new FoodProduct();
            System.out.println("Enter the item id");
            fd.setFoodId(sc.nextInt());
            System.out.println("Enter the item name");
            fd.setFoodName(sc.next());
            System.out.println("Enter the cost per unit");
            fd.setCostPerUnit(sc.nextDouble());
            System.out.println("Enter the quantity");
            fd.setQuantity(sc.nextInt());
            o.addToCart(fd);

        }

        System.out.println("Enter the bank name to avail offer");
        bank=sc.next();
        o.findDiscount(bank);

        System.out.println("Calculated Bill Amount:"+o.calculateTotalBill());

    }
}
```

```
}

package com.bean;

public class FoodProduct {

    private int foodId;
    private String foodName;
    private double costPerUnit;
    private int quantity;

    public int getFoodId() {
        return foodId;
    }

    public void setFoodId(int foodId) {
        this.foodId = foodId;
    }

    public String getFoodName() {
        return foodName;
    }

    public void setFoodName(String foodName) {
        this.foodName = foodName;
    }

    public double getCostPerUnit() {
        return costPerUnit;
    }

    public void setCostPerUnit(double costPerUnit) {
        this.costPerUnit = costPerUnit;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

}
```

### Singapore

```
import java.util.*;

public class tourism {
    static String name;
    static String place;
    static int days;
    static int tickets;
```

```

static double price = 0.00;
static double total = 0.00;
public static void main(String[] args){
    Scanner in = new Scanner(System.in);
    System.out.println("Enter the passenger name");
    name = in.nextLine();
    System.out.println("Enter the place name");
    place=in.nextLine();
    if(place.equalsIgnoreCase("beach"))

||place.equalsIgnoreCase("pilgrimage")||place.equalsIgnoreCase("heritage")||place.equalsIgnoreCase("Hills")||place.equalsIgnoreCase("palls")||place.equalsIgnoreCase("adventure")){
        System.out.println("Enter the number of days");
        days = in.nextInt();
        if(days>0){
            System.out.println("Enter the number of Tickets");
            tickets = in.nextInt();
            if(tickets>0){
                if(place.equalsIgnoreCase("beach")){
                    price = tickets*270;
                    if(price>1000){
                        total = 85*price/100;
                        System.out.printf("Price:%.2f",total);
                    }
                    else {
                        System.out.printf("Price:%.2f",price);
                    }
                }
                else if(place.equalsIgnoreCase("prilgrimage")){
                    price = tickets*350;
                    if(price>1000){
                        total = 85*price/100;
                        System.out.printf("Price:%.2f",total);
                    }
                    else {
                        System.out.printf("Price:%.2f",price);
                    }
                }
                else if(place.equalsIgnoreCase("heritage")){
                    price = tickets*430;
                    if(price>1000){
                        total = 85*price/100;
                        System.out.printf("Price:%.2f",total);
                    }
                }
            }
        }
    }
}

```

```
else {
    System.out.printf("Price:%.2f",price);
}
}

else if(place.equalsIgnoreCase("hills")){
    price = tickets*780;
    if(price>1000){
        total = 85*price/100;
        System.out.printf("Price:%.2f",total);
    }
    else {
        System.out.printf("Price:%.2f",price);
    }
}

else if(place.equalsIgnoreCase("palls")){
    price = tickets*1200;
    if(price>1000){
        total = 85*price/100;
        System.out.printf("Price:%.2f",total);
    }
    else {
        System.out.printf("Price:%.2f",price);
    }
}

else {
    price = tickets*4500;
    if(price>1000){
        total = 85*price/100;
        System.out.printf("Price:%.2f",total);
    }
    else {
        System.out.printf("Price:%.2f",price);
    }
}

}

else{
    System.out.println(tickets+" is an Invalid no. of tickets");
}
}

else{
    System.out.println(days+" is an Invalid no. of days");
}
}
```

```

        else {
            System.out.println(place+" is an Invalid place");
        }
    }
}

```

### Prime no ending

```

import java.util.*;
public class Main
{
    public static void main (String[] args) {
        int flag=0, k=0, z=0;
        Scanner sc =new Scanner(System.in );
        System.out.println("Enter the first number");
        int f=sc.nextInt();
        System.out.println("Enter the last number");
        int l=sc.nextInt();
        for(int i=f; i<=l; i++)
        {
            for(int j=2; j<i; j++)// this loop increments flag if i is divisible by j
            {
                if(i%j==0)
                {
                    flag++;
                }
            }
            if(i==l && (flag!=0 || i%10!=1))//when last number is not a prime
            {
                while(z==0)
                {
                    for(int a=2; a<i; a++)
                    {
                        if(i%a==0)
                        {
                            flag++;
                        }
                    }
                    if(i%10==1 && flag==0)
                    {
                        System.out.print(", "+i);
                        z++;
                    }
                    flag=0;
                    i++;
                }
            }
        }
    }
}

```

```

        }
    }
    if(i%10==1 && flag==0)//to check for last digit 1 and prime
    {
        if(k==0)
        {
            System.out.print(i);
            k++;
        }
        else
        {
            System.out.print(", "+i);
        }
    }
    flag=0;
}
}

}
}

```

### Query Set

```

public class Query {

    private class DataSet{
        private String theatreId;
        private String theatreName;
        private String location;
        private int noOfScreen;
        private double ticketCost;
        public String getTheatreId() {
            return theatreId;
        }
        public void setTheatreId(String theatreId) {
            this.theatreId = theatreId;
        }
        public String getTheatreName() {
            return theatreName;
        }
        public void setTheatreName(String theatreName) {
            this.theatreName = theatreName;
        }
        public String getLocation() {
            return location;
        }
    }
}

```

```
public void setLocation(String location) {
this.location = location;
}
public int getNoOfScreen() {
return noOfScreen;
}
public void setNoOfScreen(int noOfScreen) {
this.noOfScreen = noOfScreen;
}
public double getTicketCost() {
return ticketCost;
}
public void setTicketCost(double ticketCost) {
this.ticketCost = ticketCost;
}
@Override
public String toString() {
return "Theatre id: " + theatreId + "\nTheatre name: " + theatreName + "\nLocation: " + location
+ "\nNo of Screen: " + noOfScreen + "\nTicket Cost: " + ticketCost+"\n";
}
}

private String queryId;
private String queryCategory;
private DataSet primaryDataset;
private DataSet secondaryDataSet;
public String getQueryId() {
return queryId;
}
public void setQueryId(String queryId) {
this.queryId = queryId;
}
public String getQueryCategory() {
return queryCategory;
}
public void setQueryCategory(String queryCategory) {
this.queryCategory = queryCategory;
}
public DataSet getPrimaryDataset() {
return primaryDataset;
}
public void setPrimaryDataset(DataSet primaryDataset) {
this.primaryDataset = primaryDataset;
}
```

```

public DataSet getSecondaryDataSet() {
    return secondaryDataSet;
}
public void setSecondaryDataSet(DataSet secondaryDataSet) {
    this.secondaryDataSet = secondaryDataSet;
}
@Override
public String toString() {
    return "Primary data set\n" + primaryDataset
        + "Secondary data set\n" + secondaryDataSet + "Query id: " + queryId + "\nQuery category=" +
        queryCategory;
}
}

}

```

```

import java.util.Scanner;
public class TestApplication {
    public static void main(String[] args) {
        Query query = new Query();
        Scanner sc = new Scanner(System.in);
        Query.DataSet primary = query.new DataSet();
        Query.DataSet secondary = query.new DataSet();
        System.out.println("Enter the Details of primary data set");
        System.out.println("Enter the theatre id");
        String theatreid = sc.nextLine();
        primary.setTheatreId(theatreid);
        sc.nextLine();
        System.out.println("Enter the theatre name");
        String theatrename = sc.nextLine();
        primary.setTheatreName(theatrename);
        sc.nextLine();
        System.out.println("Enter the location");
        String location = sc.nextLine();
        primary.setLocation(location);
        sc.nextLine();
        System.out.println("Enter the no of screens");
        int screens = sc.nextInt();
        primary.setNoOfScreen(screens);
        System.out.println("Enter the ticket cost");
        double cost = sc.nextDouble();
        primary.setTicketCost(cost);
    }
}

```

```

System.out.println("ENter the details of secondary data set");
System.out.println("Enter the theatre id");
theatreid = sc.nextInt();
secondary.setTheatreId(theatreid);
sc.nextLine();
System.out.println("Enter the theatre name");
theatrename = sc.next();
secondary.setTheatreName(theatrename);
sc.nextLine();
System.out.println("Enter the location");
location = sc.next();
secondary.setLocation(location);
sc.nextLine();
System.out.println("Enter the no of screens");
screens = sc.nextInt();
secondary.setNoOfScreen(screens);
System.out.println("Enter the ticket cost");
cost = sc.nextDouble();
secondary.setTicketCost(cost);
System.out.println("Enter the query id");
String queryid = sc.next();
query.setQueryId(queryid);
sc.nextLine();
System.out.println("Enter the query category");
String querycategory = sc.next();
query.setQueryCategory(querycategory);
sc.nextLine();
query.setPrimaryDataset(primary);
query.setSecondaryDataSet(secondary);

System.out.println(query);
}
}

```

### Extract book

```

import java.util.Scanner;

class ExtractBook {

    public static int extractDepartmentCode(String input) {
        return Integer.parseInt(input.substring(0, 3));
    }
}

```

```

public static String extractDepartmentName(int code) {

    switch (code) {
        case 101:
            return "Accounting";
        case 102:
            return "Economics";
        case 103:
            return "Engineering";
    }

    throw new Error(code + " is invalid department code");
}

public static int extractDate(String input) {
    String yearStr = input.substring(3, 7);
    try {
        int year = Integer.parseInt(yearStr);
        if (year > 2020 || year < 1900) {
            throw new NumberFormatException();
        }
        return year;
    } catch (NumberFormatException e) {
        throw new Error(yearStr + " is invalid year");
    }
}

public static int extractNumberOfPages(String input) {
    String pagesStr = input.substring(7, 12);
    try {
        int pages = Integer.parseInt(pagesStr);
        if (pages < 10) {
            throw new NumberFormatException();
        }
        return pages;
    } catch (NumberFormatException e) {
        throw new Error(pagesStr + " are invalid pages");
    }
}

public static String extractBookId(String input) {
    String id = input.substring(12, 18);
    if (!Character.isAlphabetic(id.charAt(0)))

```

```

        throw new NumberFormatException();
    try {
        Integer.parseInt(id.substring(1));
    } catch (NumberFormatException e) {
        throw new Error(id + " is invalid book id");
    }
    return id;
}

public static void parseAndPrint(String str) {
    if (str.length() != 18) {
        System.out.println(str + " is an invalid input");
        return;
    }

    try {
        int dCode = extractDepartmentCode(str);
        String dString = extractDepartmentName(dCode);
        int year = extractDate(str);
        int pages = extractNumberOfPages(str);
        String bookId = extractBookId(str);

        System.out.println("Department Code: " + dCode);
        System.out.println("Department Name: " + dString);
        System.out.println("Year of Publication: " + year);
        System.out.println("Number of Pages: " + pages);
        System.out.println("Book Id: " + bookId);

    } catch (Error e) {
        System.out.println(e.getMessage());
    }
}

public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    String input = sc.nextLine();
    parseAndPrint(input);
    sc.close();
}
}

```

```
import java.util.*;
class FDScheme {

    private int schemeNo;
    private double depositAmt;
    private int period;
    private float rate;
    public FDScheme(int schemeNo, double depositAmt, int period) {
        super();
        this.schemeNo = schemeNo;
        this.depositAmt = depositAmt;
        this.period = period;
        calculateInterestRate();
    }
    public int getSchemeNo() {
        return schemeNo;
    }
    public void setSchemeNo(int schemeNo) {
        this.schemeNo = schemeNo;
    }
    public double getDepositAmt() {
        return depositAmt;
    }
    public void setDepositAmt(double depositAmt) {
        this.depositAmt = depositAmt;
    }
    public int getPeriod() {
        return period;
    }
    public void setPeriod(int period) {
        this.period = period;
    }
    public float getRate() {
        return rate;
    }
    public void setRate(float rate) {
        this.rate = rate;
    }

    public void calculateInterestRate()
    {
        if(period>=1 && period<=90)
        {
            this.rate=(float) 5.5;
```

```

        }
        else if(period>=91 && period<=180)
        {
            this.rate=(float) 6.25;
        }
        else if(period>=181 && period<=365)
        {
            this.rate=(float) 7.5;
        }
        System.out.println("Interest rate for "+period+" days is "+this.rate);
    }
}
public class Main{

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter Scheme no");
        int no=sc.nextInt();
        sc.nextLine();
        System.out.println("Enter Deposit amount");
        double amt=sc.nextDouble();
        System.out.println("enter period of deposit");
        int prd=sc.nextInt();
        FDScheme obj=new FDScheme(no,amt,prd);
    }
}

```

### Annual Salary

```

import java.io.*;
public class Main
{
    public static void main(String[] args)throws IOException
    {
        // Scanner sc=new Scanner(System.in);
        //Fill the code
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter the Employee Name");
        String name=br.readLine();
        System.out.println("Enter percentage of salary");
        double percent=Double.parseDouble(br.readLine());
        if(percent>0&&percent<20)
        {

```

```

System.out.println("Enter the Year of Experience");
int time=Integer.parseInt(br.readLine());

if(time>0&&time<15)
{
    double permonth=12000+(2000*(time));
    double dayshift=permonth*6;
    double nightshift=((permonth*percent)/100)+permonth)*6;
    double annualIncome=dayshift+nightshift;

    String str="The annual salary of "+name+" is";
    System.out.println(str+" "+annualIncome);

}
else{
    System.out.println((int)time+" is an invalid year of experience");
}

}
else
System.out.println((int)percent+" is an invalid percentage");
}
}

```

### Amity Passenger

```

import java.util.*;
public class PassengerAmenity {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);
System.out.println("Enter the number of passengers");
int no=sc.nextInt();
sc.nextLine();
int count=0;

if(no>0)
{
String name[]=new String[no];
String seat[]=new String[no];
String arr[]=new String[no];

for(int i=0;i<no;i++)

```

```

{
System.out.println("Enter the name of the passenger "+(i+1));
String str=sc.nextLine();

name[i]=str.toUpperCase();

System.out.println("Enter the seat details of the passenger "+(i+1));
seat[i]=sc.nextLine();

if(seat[i].charAt(0)>='A' && seat[i].charAt(0)<='S')
{

int r=Integer.parseInt(seat[i].substring(1,seat[i].length()));

if(r>=10 && r<=99)
{
count++;
}

else
{
System.out.println(r+" is invalid seat number");
break;
}
}

else
{
System.out.println(seat[i].charAt(0)+" is invalid coach");
break;
}

arr[i]=name[i]+" "+seat[i];
}

if(count==seat.length)
{

Arrays.sort(seat);

for(int i=seat.length-1;i>=0;i--)
{
for(int j=0;j<arr.length;j++)
{

```

```

        if(arr[j].contains(seat[i]))
        {
            System.out.println(arr[j]);
        }
    }

}
}

else
{
    System.out.println(no+" is invalid input");
}
}

}

```

### Change the Case

```

import java.util.*;

public class ChangeTheCase {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String a = sc.next();
        if(a.length() < 3) {
            System.out.println("String length of " + a + " is too short");
            return;
        }
        else if(a.length() > 10) {
            System.out.println("String length of " + a + " is too long");
            return;
        }

        char[] arr = a.toCharArray();
        char[] arr1 = new char[arr.length];
        int j = 0;
        for(int i = 0; i < a.length(); i++) {
            if((arr[i]<65 || ((arr[i]>90) && (arr[i]<97)) || arr[i]>122)) {
                arr1[j++] = arr[i];
            }
        }
    }
}
```

```
}

if(j!=0) {
    System.out.print("String should not contain ");
    for(int i = 0; i<=j; i++) {
        System.out.print(arr1[i]);
    }
    return;
}

char b = sc.next().charAt(0);
int present = 0;
for(int i = 0; i<a.length(); i++) {
    if(arr[i] == Character.toUpperCase(b)) {
        arr[i] = Character.toLowerCase(b);
        present = 1;
    }
    else if(arr[i] == Character.toLowerCase(b)) {
        arr[i] = Character.toUpperCase(b);
        present = 1;
    }
}

if(present == 0) {
    System.out.println("Character " + b + " is not found");
}
else {
    for(int i = 0; i <a.length(); i++) {
        System.out.print(arr[i]);
    }
}
}
```

## **Club Member**

```
import java.util.Scanner;  
  
public class ClubMember {  
    private int memberId;  
    private String memberName;  
    private String memberType;  
    private double membershipFees;
```

```
public ClubMember(int memberId, String memberName, String memberType) {
```

```
super();
this.memberId = memberId;
this.memberName = memberName;
this.memberType = memberType;
calculateMembershipFees();
}
public int getMemberId() {
    return memberId;
}
public void setMemberId(int memberId) {
    this.memberId = memberId;
}
public String getMemberName() {
    return memberName;
}
public void setMemberName(String memberName) {
    this.memberName = memberName;
}
public String getMemberType() {
    return memberType;
}
public void setMemberType(String memberType) {
    this.memberType = memberType;
}
public double getMembershipFees() {
    return membershipFees;
}
public void setMembershipFees(double membershipFees) {
    this.membershipFees = membershipFees;
}

public void calculateMembershipFees() {
    if(!(memberType == "Gold"))
    {
        this.membershipFees=(double) 50000.0;
    }
    else if(!(memberType=="Premium"))
    {
        this.membershipFees=(double) 75000.0;
    }
    System.out.println("Member Id is "+this.memberId);
    System.out.println("Member Name is "+this.memberName);
    System.out.println("Member Type is "+this.memberType);
```

```
System.out.println("Membership Fees is "+this.membershipFees);

}

}

public class Main {
public static void main(String[] args) {
Scanner sc=new Scanner(System.in);
System.out.println("Enter Member Id");
int id=sc.nextInt();
sc.nextLine();
System.out.println("Enter Name");
String name=sc.next();
System.out.println("Enter Member Type");
String type=sc.next();
ClubMember club=new ClubMember(id, name, type);
//club.calculateMembershipFees();
}
}
```

## CLUB MEMBER DETAILS

### ClubMember.java\*

```
public class ClubMember{  
    private int memberId;  
    private String memberName;  
    private String memberType;  
    private double membershipFees;  
  
    public void setMemberId(int memberId){  
        this.memberId = memberId;  
    }  
  
    public int getMemberId(){  
        return memberId;  
    }  
  
    public void setMemberName(String memberName){  
        this.memberName = memberName;  
    }  
  
    public String getMemberName(){  
        return memberName;  
    }  
  
    public void setMemberType(String memberType){  
        this.memberType = memberType;  
    }  
  
    public String getMemberType(){  
        return memberType;  
    }  
  
    public void setMembershipFees(double membershipFees){
```

```
this.membershipFees = membershipFees;  
}  
  
public double getMembershipFees(){  
    return membershipFees;  
}  
  
public ClubMember(int memberId, String memberName, String memberType){  
    this.memberId = memberId;  
    this.memberName = memberName;  
    this.memberType = memberType;  
}  
  
public void calculateMembershipFees (){  
    if (memberType.equals("Gold")) membershipFees = 50000.0;  
    else if (memberType.equals("Premium")) membershipFees = 75000.0;  
}  
}
```

### Main.java\*

```
import java.util.Scanner;  
  
public class Main{  
  
    public static void main (String[] args) {  
        Scanner sc=new Scanner(System.in);  
        System.out.println("Enter Member Id");  
        int memberId = sc.nextInt();  
        sc.nextLine();  
        System.out.println("Enter Name");  
        String memberName = sc.nextLine();
```

```
System.out.println("Enter Member Type");
String memberType = sc.next();

ClubMember clubMemberObj = new ClubMember(memberId,memberName,memberType);
clubMemberObj.calculateMembershipFees();

System.out.println("Member Id is " + clubMemberObj.getMemberId());
System.out.println("Member Name is " + clubMemberObj.getMemberName());
System.out.println("Member Type is " + clubMemberObj.getMemberType());
System.out.println("Membership Fees is " + clubMemberObj.getMembershipFees());
}

}
```

## CreditCardValidator

### CreditCard.java\*

```
package com.cts.entity;

public class CreditCard {

    private String number;

    public CreditCard() {

    }

    public CreditCard(String number) {
        super();
        this.number = number;
    }

    public String getNumber() {
        return number;
    }

    public void setNumber(String number) {
        this.number = number;
    }
}
```

### CreditCardService.java\*

```
package com.cts.services;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
```

```
import java.util.List;
import java.nio.file.*;

import com.cts.entity.CreditCard;

public class CreditCardService {

    //check whether the card is blocklisted and card contains only 16 digits

    public String validate(CreditCard card,String fileName) throws IOException
    {

        String msg=null;
        if(validateAgainstBlocklist(card, fileName))
        {
            msg="Card is blocked";
        }
        else if(validateNumber(card.getNumber()))
        {
            msg="card is not having 16 digits";
        }
        else
        {
            msg="valid card";
        }
        return msg;
    }

    // Validate a credit card against a blocklist.

    public boolean validateAgainstBlocklist(CreditCard card, String fileName) throws IOException {

        //write your code here
        boolean bol = true;
        String str = "";
        str = new String(Files.readAllBytes(Paths.get(fileName)));
        String dig[] = str.split(",");
        String str2 = dig[0];
        String str3 = dig[1];
        if(card.getNumber().equalsIgnoreCase(str2) || card.getNumber().equalsIgnoreCase(str3))

```

```
{  
    bol=true;  
}  
  
else{  
    bol=false;  
}  
  
}  
  
return bol;  
}  
  
// Validate the card number length  
public boolean validateNumber(String number) {  
    int len = number.length();  
    boolean bol=true;  
    if(len!=16)  
    {  
        bol=true;  
    }  
    else{  
        bol=false;  
    }  
  
    return bol;  
}  
  
// Get the blocklisted no's from the file and return list of numbers  
public List<String> getBlockListNumbers(String fileName) throws IOException {  
  
    List<String> li = new ArrayList<String>();  
    String data = "";  
    data = new String(Files.readAllBytes(Paths.get(fileName)));  
    String dig1[] = data.split(",");  
    for(int i=0;i<dig1.length;i++)  
    {  
        li.add(dig1[i]);  
    }  
}
```

```
    }

    return li;
}

}
```

### **SkeletonValidator.java\***

```
package com.cts.skeletonvalidator;

import java.lang.reflect.Method;
import java.util.logging.Level;
import java.util.logging.Logger;

public class SkeletonValidator {

    public SkeletonValidator() {
        validateClassName("com.cts.entity.CreditCard");
        validateClassName("com.cts.services.CreditCardService");
        validateMethodSignature(
            "validate:String,validateAgainstBlocklist:boolean,validateNumber:boolean,getBlockListNumbers>List","com.cts.services.CreditCardService");
    }

    private static final Logger LOG = Logger.getLogger("SkeletonValidator");

    protected final boolean validateClassName(String className) {

        boolean iscorrect = false;
        try {
            Class.forName(className);
            iscorrect = true;
            LOG.info("Class Name " + className + " is correct");
        } catch (ClassNotFoundException e) {
```

```

LOG.log(Level.SEVERE, "You have changed either the " + "class name/package. Use the correct package "
        + "and class name as provided in the skeleton");

} catch (Exception e) {

LOG.log(Level.SEVERE, "There is an error in validating the " + "Class Name. Please manually verify that the "
        + "Class name is same as skeleton before uploading");

}

return iscorrect;

}

protected final void validateMethodSignature(String methodWithExcptn, String className) {

Class cls = null;

try {

    String[] actualmethods = methodWithExcptn.split(",");
    boolean errorFlag = false;
    String[] methodSignature;
    String methodName = null;
    String returnType = null;

    for (String singleMethod : actualmethods) {
        boolean foundMethod = false;
        methodSignature = singleMethod.split(":");
        methodName = methodSignature[0];
        returnType = methodSignature[1];
        cls = Class.forName(className);
        Method[] methods = cls.getMethods();
        for (Method findMethod : methods) {
            if (methodName.equals(findMethod.getName())) {
                foundMethod = true;
                if (!(findMethod.getReturnType().getSimpleName().equals(returnType))) {
                    errorFlag = true;
                    LOG.log(Level.SEVERE, " You have changed the " + "return type in " + methodName

```

```

+ "" method. Please stick to the " + "skeleton provided");

} else {
    LOG.info("Method signature of " + methodName + " is valid");
}

}

if (!foundMethod) {
    errorFlag = true;
    LOG.log(Level.SEVERE, " Unable to find the given public method " + methodName
        + ". Do not change the " + "given public method name. " +
        "Verify it with the skeleton");
}

}

if (!errorFlag) {
    LOG.info("Method signature is valid");
}

}

} catch (Exception e) {
    LOG.log(Level.SEVERE," There is an error in validating the " + "method structure. Please
    manually verify that the "+ "Method signature is same as the skeleton before uploading");
}

}

}

```

### **CreditCardValidatorMain.java\***

```

package com.cts;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

```

```
import com.cts.entity.CreditCard;
import com.cts.services.CreditCardService;
import com.cts.skeletonvalidator.SkeletonValidator;

public class CreditCardValidatorMain {

    public static void main(String[] args) throws IOException {
        // CODE SKELETON - VALIDATION STARTS
        // DO NOT CHANGE THIS CODE
        BufferedReader b = new BufferedReader(new InputStreamReader(System.in));

        new SkeletonValidator();

        // CODE SKELETON - VALIDATION ENDS

        // Please start your code from here

        String cardNumber = b.readLine();
        CreditCard creditCard = new CreditCard();
        creditCard.setNumber(cardNumber);
        //Write your code here read card numnber and create CreditCard object based on cardnumber
        CreditCardService creditCardService = new CreditCardService();

        String validationMessage=creditCardService.validate(creditCard, "resources/blacklist.csv");
        System.out.println(validationMessage);
    }
}
```

## ESHOPPING

### Main.java\*

```
package com.cts.eshopping.main;

import com.cts.eshopping.orderservice.CartService;
import com.cts.eshopping.skeletonvalidator.SkeletonValidator;
import com.cts.eshopping.vo.OrderLineItem;

public class Main {

    public static void main(String ag[]) {

        // CODE SKELETON - VALIDATION STARTS
        // DO NOT CHANGE THIS CODE
        SkeletonValidator validator = new SkeletonValidator();
        // CODE SKELETON - VALIDATION ENDS

        // Please start your code from here

        OrderLineItem it1 = new OrderLineItem("AM33","Book",200,3);
        OrderLineItem it2 = new OrderLineItem("AM345","Watch",1000,2);
        CartService cs  = new CartService();

        OrderLineItem[] arr = {it1, it2};
        double amt = cs.calculateOrderTotalAmount(arr);
        System.out.println(cs.calculateDiscount(amt));

    }
}
```

**CartService.java\*/orderService**

```
package com.cts.eshopping.orderservice;

import com.cts.eshopping.vo.OrderLineItem;

/**
 *
 */
public class CartService {

    /**
     * Method to calculate total purchase amount for all the order line items
     *
     * @param orderLineItems
     * @return totalOrderAmount
     */
    public double calculateOrderTotalAmount(OrderLineItem[] orderLineItems) {

        double totalOrderAmount = 0;
        int qt = 0;
        double cost = 0.0;

        for(int i=0;i<orderLineItems.length;i++){
            qt = orderLineItems[i].quantity;
            cost = orderLineItems[i].itemCostPerQuantity;
            totalOrderAmount += (qt*cost);
        }

        return totalOrderAmount; // TODO change this return value
    }

    /**
     * Method to calculate discount based on order total amount
     *
     * @param totalOrderAmount
     */
}
```

```

* @return discount
*/
public double calculateDiscount(double totalOrderAmount) {
    double discount = 0.0;

    if(totalOrderAmount<1000){
        discount = (totalOrderAmount*10)/100;
    }
    else if(totalOrderAmount>=1000 && totalOrderAmount<10000){
        discount = (totalOrderAmount*20)/100;
    }
    else if(totalOrderAmount>=10000){
        discount = (totalOrderAmount*30)/100;
    }

    return discount; // TODO change this return value
}

/***
 * Method to verify if the order line item is flagged as Bulk Order or not
 *
 * @param linelitem
 * @return boolean
 */
public boolean isBulkOrder(OrderLineItem linelitem) {
    boolean result=false;

    if(linelitem.quantity>5){
        result = true;
    }
    else if(linelitem.quantity<=5 && linelitem.quantity>=1){
        result=false;
    }

    return result; // TODO change this return value
}

```

```

/**
 * Count the number of line items which are ordered in bulk
 *
 * @param orderLineItems
 * @return
 */
public int countOfBulkOrderLineItems(OrderLineItem[] orderLineItems) {
    int count = 0;

    for(int i=0;i<orderLineItems.length;i++){
        if(isBulkOrder(orderLineItems[i])){
            count++;
        }
    }

    return count; // TODO change this return value
}

}

```

### **SkeletonValidator.java\***

```

package com.cts.eshopping.skeletonvalidator;

import java.lang.reflect.Method;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * @author 222805
 *
 * This class is used to verify if the Code Skeleton is intact and not modified by participants thereby ensuring smooth
 * auto evaluation
 *
 */

```

```

public class SkeletonValidator {

    public SkeletonValidator() {
        validateClassName("com.cts.eshopping.orderservice.CartService");
        validateClassName("com.cts.eshopping.vo.OrderLineItem");
        validateMethodSignature(
            "calculateOrderTotalAmount:double,calculateDiscount:double,isBulkOrder:boolean,countOfBulkOrderLineItems:int",
            "com.cts.eshopping.orderservice.CartService");
    }

    private static final Logger LOG = Logger.getLogger("SkeletonValidator");

    protected final boolean validateClassName(String className) {
        boolean iscorrect = false;
        try {
            Class.forName(className);
            iscorrect = true;
            LOG.info("Class Name " + className + " is correct");
        } catch (ClassNotFoundException e) {
            LOG.log(Level.SEVERE, "You have changed either the " + "class name/package. Use the correct package "
                + "and class name as provided in the skeleton");
        } catch (Exception e) {
            LOG.log(Level.SEVERE,"There is an error in validating the " + "Class Name. Please manually
            verify that the " + "Class name is same as skeleton before uploading");
        }
        return iscorrect;
    }
}

```

```

protected final void validateMethodSignature(String methodWithExcptn, String className) {

    Class cls = null;

    try {

        String[] actualmethods = methodWithExcptn.split(",");
        boolean errorFlag = false;
        String[] methodSignature;
        String methodName = null;
        String returnType = null;

        for (String singleMethod : actualmethods) {

            boolean foundMethod = false;
            methodSignature = singleMethod.split(":");

            methodName = methodSignature[0];
            returnType = methodSignature[1];
            cls = Class.forName(className);
            Method[] methods = cls.getMethods();
            for (Method findMethod : methods) {
                if (methodName.equals(findMethod.getName())) {
                    foundMethod = true;
                    if (!(findMethod.getReturnType().getName().equals(returnType))) {
                        errorFlag = true;
                        LOG.log(Level.SEVERE, " You have changed the " + "return type in '" + methodName
                                + "' method. Please stick to the " + "skeleton provided");
                }
            }
            if (!foundMethod) {
                errorFlag = true;
            }
        }
    }
}

```

```

        LOG.log(Level.SEVERE, " Unable to find the given public method " + methodName
+ ". Do not change the " + "given public method name. " + "Verify it with the skeleton");
    }

}

if (!errorFlag) {
    LOG.info("Method signature is valid");
}

}

} catch (Exception e) {
    LOG.log(Level.SEVERE," There is an error in validating the " + "method structure. Please
manually verify that the "+ "Method signature is same as the skeleton before uploading");
}

}

}

```

### **OrderLineItem.java\***

```

package com.cts.eshopping.vo;

/**
 * @author Value Object - OrderLineItem
 *
 */
public class OrderLineItem {

    public String itemId;
    public String itemName;
    public double itemCostPerQuantity;
    public int quantity;

    public String getItemId(){
        return itemId;
    }
}

```

```
public void setItemId(String itemId){  
    this.itemId = itemId;  
}  
  
public String getItemName(){  
    return itemName;  
}  
  
public void setItemName(String itemName){  
    this.itemName = itemName;  
}  
  
public double getItemCostPerQuantity(){  
    return itemCostPerQuantity;  
}  
  
public void setItemCostPerQuantity(double itemCostPerQuantity){  
    this.itemCostPerQuantity = itemCostPerQuantity;  
}  
  
public int getQuantity(){  
    return quantity;  
}  
  
public void setItemId(int quantity){  
    this.quantity = quantity;  
}  
  
public OrderLineItem(String itemId, String itemName, double itemCostPerQuantity, int quantity){  
    this.itemId = itemId;  
    this.itemName = itemName;  
    this.itemCostPerQuantity=itemCostPerQuantity;  
    this.quantity = quantity;  
}  
}
```

## Fixed Deposit Details

### FDScheme.java\*

```
import java.util.*;  
  
class FDScheme{  
  
    private int schemeNo;  
  
    private double depositAmt;  
  
    private int period;  
  
    private float rate;  
  
    public FDScheme(int schemeNo, double depositAmt, int period){  
  
        super();  
  
        this.schemeNo=schemeNo;  
  
        this.depositAmt=depositAmt;  
  
        this.period=period;  
  
        calculateInterestRate();  
  
    }  
  
    public int getSchemeNo(){  
  
        return schemeNo;  
  
    }  
  
    public void setSchemeNo(int schemeNo)  
  
    {  
  
        this.schemeNo=schemeNo;  
  
    }  
  
    public double getDepositAmt(){  
  
        return depositAmt;  
  
    }  
  
    public void setDepositAmt(double depositAmt)  
  
    {  
  
        this.depositAmt=depositAmt;  
  
    }  
  
    public int getPeriod()  
  
    {  
  
        return period;  
  
    }  
  
    public void setPeriod(int period){
```

```
this.period=period;
}

public float getRate(){
    return rate;
}

public void setRate(float rate){
    this.rate=rate;
}

public void calculateInterestRate()
{
    if(period>=1 && period<=90)
    {
        this.rate=(float)5.5;
    }
    else if(period>=91 && period<=180)
    {
        this.rate=(float)6.25;
    }
    else if(period>=181 && period<=365)
    {
        this.rate=(float)7.5;
    }
    System.out.println("Interest rate for"+period+"days is"+this.rate);
}
}
```

### Main.java\*

```
import java.util.Scanner;

public class Main{

    public static void main (String[] args) {
        Scanner sc=new Scanner(System.in);
```

```
System.out.println("Enter Scheme no");
int no=sc.nextInt();
sc.nextLine();
System.out.println("Enter Deposit amount");
double amt=sc.nextDouble();
System.out.println("enter period of deposit");
int prd=sc.nextInt();
FDScheme obj=new
FDScheme(no,amt,prd);
}
}
```

## GPA CALCULATION

### UserInterface.java\*

```
package com.ui;

import com.utility.*;
import java.util.*;

public class UserInterface {

    public static void main(String []args)
    {

        GPACalculator gpa = new GPACalculator();

        gpa.setGradePointList(new ArrayList<Integer>());

        int option=0;

        double gpa1=0;

        Scanner sc = new Scanner(System.in);

        do

        {

            System.out.println("1. Add Grade\n2. Calculate GPA\n3. Exit");

            System.out.println("Enter your choice");

            option = Integer.valueOf(sc.nextLine());

            switch(option)

            {

                case 1: System.out.println("Enter the obtained grade");

                    char grade = sc.nextLine().charAt(0);

                    gpa.addGradePoint(grade);

                    break;

                case 2 : gpa1 = gpa.calculateGPAScored();

                    if(gpa1 > 0)

                    {

                        System.out.println("GPA Scored");

                        System.out.println(gpa1);

                    }

                    else

                    {

                        System.out.println("No GradePoints available");
                    }
            }
        }
    }
}
```

```

        }

        break;

    case 3 : break;

}

}while(option!=3);

System.out.println("Thank you for using the Application");

}

}

```

### **GPACalculator.java\***

```

package com.utility;

import java.util.*;

public class GPACalculator {

    private List<Integer> gradePointList;

    public List<Integer> getGradePointList() {
        return gradePointList;
    }

    public void setGradePointList(List<Integer> gradePointList) {
        this.gradePointList = gradePointList;
    }

    /*This method should add equivalent grade points based on the grade obtained by the student passed as argument into gradePointList

```

Grade	S	A	B	C	D	E
Grade Point	10	9	8	7	6	5

For example if the grade obtained is A, its equivalent grade points is 9 has to be added into the gradePointList\*/

```

public void addGradePoint(char gradeObtained) {

    if(gradeObtained == 'S')
    {
        gradePointList.add(10);
    }
    else if(gradeObtained == 'A')
    {
        gradePointList.add(9);
    }
    else if(gradeObtained == 'B')
    {
        gradePointList.add(8);
    }
    else if(gradeObtained == 'C')
    {
        gradePointList.add(7);
    }
    else if(gradeObtained == 'D')
    {
        gradePointList.add(6);
    }
    else
    {
        gradePointList.add(5);
    }
}

/* This method should return the GPA of all grades scored in the semester
GPA can be calculated based on the following formula
GPA= (gradePoint1 + gradePoint2 + ... + gradePointN) / (size of List)

```

For Example:

if the list contains the following marks [9,10,8,5]

GPA =  $(9 + 10 + 8 + 5) / (4)$ = 8.0 \*/

```
public double calculateGPAScored() {  
  
    double gpa=-1;  
    double total=0,value=0,size=0;  
  
    size = gradePointList.size();  
    if(size < 1)  
    {  
        return 0;  
    }  
    // fill the code  
    Iterator i = gradePointList.iterator();  
    while(i.hasNext())  
    {  
        value = (Integer)i.next();  
        total += value;  
    }  
    gpa = total/size;  
  
    return gpa;  
}  
}
```

## HUNGER EATS

### FoodProduct.java\*

```
package com.bean;

public class FoodProduct {

    private int foodId;
    private String foodName;
    private double costPerUnit;
    private int quantity;

    public int getFoodId() {
        return foodId;
    }

    public void setFoodId(int foodId) {
        this.foodId = foodId;
    }

    public String getFoodName() {
        return foodName;
    }

    public void setFoodName(String foodName) {
        this.foodName = foodName;
    }

    public double getCostPerUnit() {
        return costPerUnit;
    }

    public void setCostPerUnit(double costPerUnit) {
        this.costPerUnit = costPerUnit;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }
}
```

```
}
```

```
}
```

### **UserInterface.java\***

```
package com.ui;

import java.util.Scanner;

import com.utility.Order;

import com.bean.FoodProduct;

public class UserInterface {

    public static void main(String[] args) {

        // fill the code
        Scanner sc=new Scanner(System.in);
        int itemno;
        String bank;
        System.out.println("Enter the number of items");
        itemno=sc.nextInt();
        System.out.println("Enter the item details");
        Order z=new Order();
        for(int i=0;i<itemno;i++){
            FoodProduct fd=new FoodProduct();
            System.out.println("Enter the item id");
            fd.setFoodId(sc.nextInt());
            System.out.println("Enter the item name");
            fd.setFoodName(sc.next());
            System.out.println("Enter the cost per unit");
            fd.setCostPerUnit(sc.nextDouble());
            System.out.println("Enter the quantity");
            fd.setQuantity(sc.nextInt());
            z.addToCart(fd);
        }
    }
}
```

```
        System.out.println("Enter the bank name to avail offer");
        bank=sc.next();
        z.findDiscount(bank);
        System.out.println("Calculated Bill Amount:"+z.calculateTotalBill());
    }

}
```

### **Order.java\***

```
package com.utility;

import java.util.*;
import com.bean.FoodProduct;

public class Order {

    private double discountPercentage;
    private List<FoodProduct> foodList=new ArrayList<FoodProduct>();

    public double getDiscountPercentage() {
        return discountPercentage;
    }

    public void setDiscountPercentage(double discountPercentage) {
        this.discountPercentage = discountPercentage;
    }

    public List<FoodProduct> getFoodList() {
        return foodList;
    }

    public void setFoodList(List<FoodProduct> foodList) {
        this.foodList = foodList;
    }
}
```

```
//This method should set the discount percentage based on bank passed as argument
```

```
public void findDiscount(String bankName) {
```

```
    // fill the code
```

```
    if(bankName.equals("HDFC")){
```

```
        discountPercentage=15.0;
```

```
    }
```

```
    else if(bankName.equals("ICICI")){
```

```
        discountPercentage=25.0;
```

```
    }
```

```
    else if(bankName.equals("CUB")){
```

```
        discountPercentage=30.0;
```

```
    }
```

```
    else if(bankName.equals("SBI")){
```

```
        discountPercentage=50.0;
```

```
    }
```

```
    else if(bankName.equals("OTHERS")){
```

```
        discountPercentage=0.0;
```

```
    }
```

```
}
```

```
//This method should add the FoodProduct Object into Food List
```

```
public void addToCart(FoodProduct foodProductObject) {
```

```
    // fill the code
```

```
    List<FoodProduct> f=getFoodList();
```

```
    f.add(foodProductObject);
```

```
    setFoodList(f);
```

```
}
```

```
//method should return the total bill amount after discount
```

```
// based on the bank name  
public double calculateTotalBill() {  
  
    // fill the code  
    double bill=0;  
    List<FoodProduct> f=getFoodList();  
    for(int i=0;i<f.size();i++){  
        //  
        // System.out.println(f.get(i).getCostPerUnit());  
        //  
        // System.out.println(f.get(i).getQuantity());  
        bill+=f.get(i).getQuantity()*f.get(i).getCostPerUnit()*1.0;  
    }  
    bill=bill-((bill*discountPercentage)/100);  
  
    return bill;  
}  
  
}
```

## INSURANCE PREMIUM GENERATOR

### PropertyDetails.java\*

```
package com.cts.insurance.entity;

public class PropertyDetails {

    private Integer builtUpArea;
    private Integer builtYear;
    private Integer reconstructionCost;
    private Integer householdValuation;
    private String burglaryCoverReqd;
    private String politicalUnrestCoverReqd;
    private Integer sumAssured;

    public PropertyDetails() {

    }

    public Integer getBuiltUpArea() {
        return builtUpArea;
    }

    public void setBuiltUpArea(Integer builtUpArea) {
        this.builtUpArea = builtUpArea;
    }

    public Integer getBuiltYear() {
        return builtYear;
    }

    public void setBuiltYear(Integer builtYear) {
        this.builtYear = builtYear;
    }

    public Integer getReconstructionCost() {
```

```
        return reconstructionCost;
    }

    public void setReconstructionCost(Integer reconstructionCost) {
        this.reconstructionCost = reconstructionCost;
    }

    public Integer getHouseholdValuation() {
        return householdValuation;
    }

    public void setHouseholdValuation(Integer householdValuation) {
        this.householdValuation = householdValuation;
    }

    public String getBurglaryCoverReqd() {
        return burglaryCoverReqd;
    }

    public void setBurglaryCoverReqd(String burglaryCoverReqd) {
        this.burglaryCoverReqd = burglaryCoverReqd;
    }

    public String getPoliticalUnrestCoverReqd() {
        return politicalUnrestCoverReqd;
    }

    public void setPoliticalUnrestCoverReqd(String politicalUnrestCoverReqd) {
        this.politicalUnrestCoverReqd = politicalUnrestCoverReqd;
    }

    public Integer getSumAssured() {
        return sumAssured;
    }
```

```

public void setSumAssured(Integer sumAssured) {
    this.sumAssured = sumAssured;
}

public PropertyDetails(Integer builtUpArea, Integer builtYear, Integer reconstructionCost, Integer
householdValuation,
String burglaryCoverReqd, String politicalUnrestCoverReqd) {
super();
this.builtUpArea = builtUpArea;
this.builtYear = builtYear;
this.reconstructionCost = reconstructionCost;
this.householdValuation = householdValuation;
this.burglaryCoverReqd = burglaryCoverReqd;
this.politicalUnrestCoverReqd = politicalUnrestCoverReqd;
}

}


```

#### **Constants.java\***

```

package com.cts.insurance.misc;

public class Constants {
    public final static String YES = "Yes";
    public final static String NO = "No";
    public final static double MIN_PREMIUM_AMOUNT = 5000;
    public final static int MIN_HOUSEHOLD_VALUATION = 0;
}

```

#### **CalculatePremiumService.java\***

```

package com.cts.insurance.services;

```

```

import com.cts.insurance.entity.PropertyDetails;
import com.cts.insurance.misc.Constants;

import java.time.LocalDate;

public class CalculatePremiumService {

    public boolean checkOwnerDetails(String name, String mobile) {
        //name cannot have numbers or special characters; minimum length of name=2
        //mobile number begins with any digit between 6 and 9; length=10
        return name.matches("^([a-zA-Z]{2,})$") && mobile.matches("^[6-9][0-9]{9}$");
    }

    public double getPremiumAmount(PropertyDetails propertyDetails) {
        double amountToBePaid = 0;
        double additionalAmount1 = 0;
        double additionalAmount2 = 0;
        /*invoke validatePropertyParameters(propertyDetails) and check the response
         * if true ,calculate premium amount to be paid by calling
         * the methods calculatePremiumByPropertyAge(propertyDetails),
         * calculatePremiumForBurglaryCoverage(propertyDetails, amountToBePaid) and
         * calculatePremiumForPoliticalUnrestCoverage(propertyDetails, amountToBePaid)
         *
         * return the premium amount rounded off to zero decimal places
         * else return 0;
        */
        if(!validatePropertyParameters(propertyDetails)) {
            return 0;
        }

        amountToBePaid = calculatePremiumByPropertyAge(propertyDetails);
        additionalAmount1 = calculatePremiumForBurglaryCoverage(propertyDetails, amountToBePaid);
        additionalAmount2 = calculatePremiumForPoliticalUnrestCoverage(propertyDetails,
amountToBePaid);

        return Math.round(amountToBePaid + additionalAmount1 + additionalAmount2);
    }
}

```

```

}

public boolean validatePropertyParameters(PropertyDetails propertyDetails) {
    /*
     * conditions to be checked
     * builtUpArea between 400 and 15,000 sq. ft.
     * reconstructionCost between Rs.1,000 and Rs.10,000
     * householdValuation either same as Constants.MIN_HOUSEHOLD_VALUATION
     * between Rs.1,00,000 and Rs.15,00,000
     * builtYear between 2000 and current year
     */
    int builtUpArea = propertyDetails.getBuiltUpArea();
    if(!(builtUpArea>=400 && builtUpArea<=15000)) return false;
    int reconstructionCost = propertyDetails.getReconstructionCost();
    if(!(reconstructionCost>=1000 && reconstructionCost<=10000)) return false;
    int householdValuation = propertyDetails.getHouseholdValuation();
    if(!((householdValuation==Constants.MIN_HOUSEHOLD_VALUATION) || (householdValuation >= 100000 && householdValuation <= 1500000))) {
        return false;
    }
    int builtYear = propertyDetails.getBuiltYear();
    if(!(builtYear>=2000 && builtYear<=LocalDate.now().getYear())) {
        return false;
    }
    return true;
}

public double calculatePremiumByPropertyAge(PropertyDetails propertyDetails) {
    //Write your code here based on business rules
    //Use Constants.MIN_PREMIUM_AMOUNT
    int sumAssured =
propertyDetails.getBuiltUpArea()*propertyDetails.getReconstructionCost()+propertyDetails.getHouseholdValuation());
    int propertyAge = LocalDate.now().getYear()-propertyDetails.getBuiltYear();
    propertyDetails.setSumAssured(sumAssured);
    double premium = 0;
    if(propertyAge>15) {

```

```

        premium = Constants.MIN_PREMIUM_AMOUNT+(propertyDetails.getSumAssured()*0.35);

    }

    else if(propertyAge>=6) {

        premium = Constants.MIN_PREMIUM_AMOUNT+(propertyDetails.getSumAssured()*0.2);

    }

    else {

        premium = Constants.MIN_PREMIUM_AMOUNT+(propertyDetails.getSumAssured()*0.1);

    }

    return premium;

}

public double calculatePremiumForBurglaryCoverage(PropertyDetails propertyDetails, double amount) {

    //write your code here based on business rules

    if(propertyDetails.getBurglaryCoverReqd().equalsIgnoreCase(Constants.YES)) {

        return amount*.01;

    }

    return 0;

}

public double calculatePremiumForPoliticalUnrestCoverage(PropertyDetails propertyDetails, double amount) {

    //Write your code here based on business rules

    //Ex:-propertyDetails.getPoliticalUnrestCoverReqd().equalsIgnoreCase(Constants.YES) to check
condition

    if(propertyDetails.getPoliticalUnrestCoverReqd().equalsIgnoreCase(Constants.YES)) {

        return amount*.01;

    }

    return 0;

}

}

```

**SkeletonValidator.java\***

```
package com.cts.insurance.skeleton;

import java.lang.reflect.Method;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * @author
 *
 * This class is used to verify if the Code Skeleton is intact and not modified by participants thereby ensuring smooth
auto evaluation
 *
 */
public class SkeletonValidator {

    public SkeletonValidator() {
        validateClassName("com.cts.insurance.entity.PropertyDetails");
        validateClassName("com.cts.insurance.misc.Constants");
        validateClassName("com.cts.insurance.services.CalculatePremiumService");
        validateClassName("com.cts.insurance.InsurancePremiumGeneratorApp");
        validateMethodSignature(
            "checkOwnerDetails:boolean,getPremiumAmount:double,validatePropertyParameters:boolean,calculatePremiumByPropertyAge:double,calculatePremiumForBurglaryCoverage:double,calculatePremiumForPoliticalUnrestCoverage:double","com.cts.insurance.services.CalculatePremiumService");
    }

    private static final Logger LOG = Logger.getLogger("SkeletonValidator");

    protected final boolean validateClassName(String className) {

        boolean iscorrect = false;
        try {
            Class.forName(className);
            iscorrect = true;
        }
        catch (Exception e) {
            iscorrect = false;
        }
        return iscorrect;
    }
}
```

```

        LOG.info("Class Name " + className + " is correct");

    } catch (ClassNotFoundException e) {
        LOG.log(Level.SEVERE, "You have changed either the " + "class name/package. Use the
correct package "+ "and class name as provided in the skeleton");

    } catch (Exception e) {
        LOG.log(Level.SEVERE,
                "There is an error in validating the " + "Class Name. Please manually verify that the "
                + "Class name is same as skeleton before uploading");
    }

    return iscorrect;
}

protected final void validateMethodSignature(String methodWithExcptn, String className) {
    Class cls = null;
    try {

        String[] actualmethods = methodWithExcptn.split(",");
        boolean errorFlag = false;
        String[] methodSignature;
        String methodName = null;
        String returnType = null;

        for (String singleMethod : actualmethods) {
            boolean foundMethod = false;
            methodSignature = singleMethod.split(":");
            methodName = methodSignature[0];
            returnType = methodSignature[1];
            cls = Class.forName(className);
            Method[] methods = cls.getMethods();
            for (Method findMethod : methods) {

```

```

        if (methodName.equals(findMethod.getName())) {
            foundMethod = true;
            if (!(findMethod.getReturnType().getSimpleClassName().equals(returnType))) {
                errorFlag = true;
                LOG.log(Level.SEVERE, " You have changed the " + "return type in " + methodName+ "" " method. Please stick to the " + "skeleton provided");
            }
        } else {
            LOG.info("Method signature of " + methodName + " is valid");
        }
    }

    if (!foundMethod) {
        errorFlag = true;
        LOG.log(Level.SEVERE, " Unable to find the given public method " + methodName + ". Do not change the " + "given public method name. " + "Verify it with the skeleton");
    }
}

if (!errorFlag) {
    LOG.info("Method signature is valid");
}
}

} catch (Exception e) {
    LOG.log(Level.SEVERE," There is an error in validating the " + "method structure. Please manually verify that the "+ "Method signature is same as the skeleton before uploading");
}

}
}

```

**InsurancePremiumGeneratorApp.java\***

```
package com.cts.insurance;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

import com.cts.insurance.entity.PropertyDetails;
import com.cts.insurance.misc.Constants;
import com.cts.insurance.services.CalculatePremiumService;
import com.cts.insurance.skeleton.SkeletonValidator;

public class InsurancePremiumGeneratorApp {

    public static void main(String[] args) throws IOException {

        // CODE SKELETON - VALIDATION STARTS
        // DO NOT CHANGE THIS CODE

        SkeletonValidator validator = new SkeletonValidator();

        // CODE SKELETON - VALIDATION ENDS

        // Please start your code from here
        String name = "";
        String mobile = "";
        Integer builtUpArea = 0;
        Integer builtYear=0;
        Integer reconstructionCost = 0;
        Integer householdValuation = Constants.MIN_HOUSEHOLD_VALUATION;
        String burglaryCoverReqd = "";
        String politicalUnrestCoverReqd = "";

        //writer the code for creating BufferedReader object here
    }
}
```

```

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
CalculatePremiumService premiumService = new CalculatePremiumService();

System.out.println("Enter the name");
//read name
name = br.readLine();

System.out.println("Enter the mobile");
//read mobile
mobile = br.readLine();

//validate name and mobile format; continue if true
if(premiumService.checkOwnerDetails(name, mobile)) {

System.out.println("Enter the Built-Up Area(In sq.ft.)between 400 and 15,000");
//read builtUpArea
builtUpArea = Integer.parseInt(br.readLine());

System.out.println("Enter the year the house was built");
//read builtYear
builtYear = Integer.parseInt(br.readLine());

System.out.println("Enter the Re-construction cost(per sq.ft.)between 1,000 and 10,000");
//read reconstructionCost
reconstructionCost = Integer.parseInt(br.readLine());

System.out.println(
"Do you want to include valuation of HouseHold Articles? Please provide yes/no");

//read response
String response = br.readLine();

//if (response is "yes" case insensitive)
if(response.equalsIgnoreCase("yes")) {

System.out.println("Enter the Household valuation between Rs.1,00,000 and Rs.15,00,000");
//read householdValuation
householdValuation = Integer.parseInt(br.readLine());
}

System.out.println("Do you want to include Burglary cover? Please provide yes/no");

```

```
//read burglaryCoverReqd  
burglaryCoverReqd = br.readLine();  
  
System.out.println("Do you want to include Political unrest cover? Please provide yes/no");  
  
//read politicalUnrestCoverReqd  
politicalUnrestCoverReqd = br.readLine();  
  
//create PropertyDetails Object  
  
PropertyDetails propertyDetails = new PropertyDetails(builtUpArea, builtYear, reconstructionCost,  
householdValuation, burglaryCoverReqd, politicalUnrestCoverReqd);  
  
double premiumAmount = premiumService.getPremiumAmount(propertyDetails);  
  
if(premiumAmount==0.0) {  
  
    System.out.println("Incorrect figures provided");  
  
}else {  
  
    System.out.println("Sum Insured: Rs." +propertyDetails.getSumAssured()+"\nInsurance  
Premium for the property of " + name + ": Rs." + premiumAmount);  
  
}  
  
}  
  
}
```

## NUMEROLOGY NUMBER

### Main.java\*

```
import java.util.Scanner;

public class Main {
    private static int getSum(long num) {
        char[] chars = Long.toString(num).toCharArray();
        int sum = 0;

        for (char ch : chars) {
            sum += Character.digit(ch, 10);
        }

        return sum;
    }

    private static int getNumerology(long num) {
        String string = String.valueOf(num);

        while (string.length() != 1) {
            string = String.valueOf(getSum(Long.parseLong(string)));
        }

        return Integer.parseInt(string);
    }

    private static int getOddCount(long num) {
        int oddCount = 0;

        for (char ch : Long.toString(num).toCharArray()) {
            if (Character.digit(ch, 10) % 2 != 0) {
                ++oddCount;
            }
        }
    }
}
```

```
        return oddCount;
    }

private static int getEvenCount(long num) {
    int evenCount = 0;

    for (char ch : Long.toString(num).toCharArray()) {
        if (Character.digit(ch, 10) % 2 == 0) {
            ++evenCount;
        }
    }

    return evenCount;
}

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    System.out.println("Enter the number");
    long num = scanner.nextLong();

    System.out.println("Sum of digits");
    System.out.println(getSum(num));

    System.out.println("Numerology number");
    System.out.println(getNumerology(num));

    System.out.println("Number of odd numbers");
    System.out.println(getOddCount(num));

    System.out.println("Number of even numbers");
    System.out.println(getEvenCount(num));
}
```

## OIL STORES

### Oil.java\*

```
import java.util.Scanner;

public class Oil{

    //Fill the code here

    private String name;
    private int pack;
    private char category;
    private float cost;

    public Oil(String name,int pack,char category,float cost){

        this.name=name;
        this.pack=pack;
        this.category=category;
        this.cost=cost;
    }

    public void setName(String name){

        this.name=name;
    }

    public String getName(){

        return name;
    }

    public void setPack(int pack){

        this.pack=pack;
    }

    public int getPack(){

        return pack;
    }

    public void setCategory(char category){

        this.category=category;
    }

    public char getCategory(){

        return category;
    }
}
```

```
}

public void setCost(float cost){

    this.cost=cost;
}

public float getCost(){

    return cost;
}

public float calculateTotalCost(float qty){

    float price=((qty*1000)/pack)*cost;

    return price;
}

}
```

### Main.java\*

```
import java.util.Scanner;

public class Main{

    public static void main (String[] args) {

        Scanner sc=new Scanner(System.in);

        //Fill the code

        System.out.println("Enter oil name");

        String n=sc.next();

        System.out.println("Enter pack capacity");

        int pc=sc.nextInt();

        System.out.println("Enter category");

        char cat=sc.next().charAt(0);

        System.out.println("Enter cost");

        float c=sc.nextFloat();

        Oil obj=new Oil(n,pc,cat,c);

        obj.setName(n);

        obj.setPack(pc);

        obj.setCategory(cat);
    }
}
```

```
obj.setCost(c);
System.out.println("Enter Quantity to purchase");
float qty=sc.nextFloat();
System.out.println("Oil cost rs."+obj.calculateTotalCost(qty));
}
}
```

## PAYMENT-INHERITENCE

**Bill.java\***

```
public class Bill {

    public String processPayment(Payment payObj) {
        String result="Payment not done and your due amount is "+payObj.getDueAmount();
        // Fill your code
        if(payObj instanceof Cheque)
        {
            Cheque cheque=(Cheque)payObj;
            if(cheque.payAmount())
                result="Payment done successfully via cheque";
        }
        else if(payObj instanceof Cash)
        {
            Cash cash=(Cash)payObj;
            if(cash.payAmount())
                result="Payment done successfully via cash";
        }
        else if(payObj instanceof Credit)
        {
            Credit credit=(Credit)payObj;
            if(credit.payAmount())
                result="Payment done successfully via credit card. Remaining amount in your "+credit.getCardType()+" card
is "+credit.getCreditCardAmount();
        }
        return result;
    }
}
```

**Cash.java\***

```
public class Cash extends Payment
{
    private int cashAmount;

    public int getCashAmount() {
        return cashAmount;
    }

    public void setCashAmount(int cashAmount) {
        this.cashAmount = cashAmount;
    }

    public boolean payAmount()
    {
        return getCashAmount() >= getDueAmount();
    }
}
```

**Cheque.java**

```
import java.util.*;
import java.util.GregorianCalendar;
import java.text.ParseException;
import java.util.Calendar;
import java.util.Date;
import java.text.SimpleDateFormat;

public class Cheque extends Payment
{
    private String chequeNo;
    private int chequeAmount;
    private Date dateOfIssue;

    public String getChequeNo() {
```

```

        return chequeNo;
    }

    public void setChequeNo(String chequeNo) {
        this.chequeNo = chequeNo;
    }

    public int getChequeAmount() {
        return chequeAmount;
    }

    public void setChequeAmount(int chequeAmount) {
        this.chequeAmount = chequeAmount;
    }

    public Date getDateOfIssue() {
        return dateOfIssue;
    }

    public void setDateOfIssue(Date dateOfIssue) {
        this.dateOfIssue = dateOfIssue;
    }

    private int findDifference(Date date){
        Calendar myDate=new GregorianCalendar();
        myDate.setTime(date);
        return (2020-myDate.get(Calendar.YEAR))*12+(0-myDate.get(Calendar.MONTH));
    }

    @Override
    public boolean payAmount()
    {
        int months=findDifference(getDateOfIssue());
        return (getChequeAmount()>=getDueAmount()&months<=6);
    }

// Fill your code

    public void generateDate(String date)
    {
        try{
            Date issueDate=new SimpleDateFormat("dd-MM-yyyy").parse(date);

```

```
        setDateOfIssue(issueDate);

    }

    catch(ParseException e)
    {
        e.printStackTrace();
    }

}

}
```

### Credit.java\*

```
public class Credit extends Payment
{
    private int creditCardNo;

    private String cardType; //(silver,gold,platinum) String,
    private int creditCardAmount;

    public int getCreditCardNo() {
        return creditCardNo;
    }

    public void setCreditCardNo(int creditCardNo) {
        this.creditCardNo = creditCardNo;
    }

    public String getCardType() {
        return cardType;
    }

    public void setCardType(String cardType) {
        this.cardType = cardType;
    }

    public int getCreditCardAmount() {
        return creditCardAmount;
    }

    public void setCreditCardAmount(int creditCardAmount) {
```

```
        this.creditCardAmount = creditCardAmount;  
    }  
  
    public Credit(int creditCardNo, String cardType) {  
        super();  
  
        // Fill your code  
  
    }  
  
    public Credit()  
    {  
  
    }  
  
    // Fill your code  
  
    @Override  
  
    public boolean payAmount()  
    {  
        int tax=0;  
  
        boolean isDeducted=false;  
  
        switch(cardType)  
        {  
            case "silver": setCreditCardAmount(10000);  
                tax=(int)(0.02*getDueAmount())+getDueAmount();  
                if(tax<=getCreditCardAmount())  
                {  
                    setCreditCardAmount(getCreditCardAmount()-tax);  
                    isDeducted=true;  
                }  
                break;  
  
            case "gold": setCreditCardAmount(50000);  
                tax=(int)(0.05*getDueAmount())+getDueAmount();  
                if(tax<=getCreditCardAmount())  
        }
```

```

    {
        setCreditCardAmount(getCreditCardAmount()-tax);
        isDeducted=true;
    }
    break;

case "platinum": setCreditCardAmount(100000);
    tax=(int)(0.1*getDueAmount())+getDueAmount();
    if(tax<=getCreditCardAmount())
    {
        setCreditCardAmount(getCreditCardAmount()-tax);
        isDeducted=true;
    }
    break;
}

return isDeducted;
}
}

```

### **Payment.java\***

```

public class Payment {

    private int dueAmount;

    public int getDueAmount() {
        return dueAmount;
    }

    public void setDueAmount(int dueamount) {
        this.dueAmount = dueamount;
    }

    public boolean payAmount() {

        return false;
    }
}

```

**Main.java\***

```
import java.util.*;  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        // Fill your code  
        Bill bill=new Bill();  
        Scanner sc=new Scanner(System.in);  
        System.out.println("Enter the due amount:");  
        int dueAmount=sc.nextInt();  
        System.out.println("Enter the mode of payment(cheque/cash/credit):");  
        String mode=sc.next();  
        switch(mode){  
            case "cash" : System.out.println("Enter the cash amount:");  
                int cashAmount=sc.nextInt();  
                Cash cash=new Cash();  
                cash.setCashAmount(cashAmount);  
                cash.setDueAmount(dueAmount);  
                System.out.println(bill.processPayment(cash));  
                break;  
            case "cheque" : System.out.println("Enter the cheque number:");  
                String number=sc.next();  
                System.out.println("Enter the cheque amount:");  
                int chequeAmount=sc.nextInt();  
                System.out.println("Enter the date of issue:");  
                String date=sc.next();  
                Cheque cheque=new Cheque();  
                cheque.setChequeAmount(chequeAmount);  
                cheque.setChequeNo(number);  
                cheque.generateDate(date);  
                cheque.setDueAmount(dueAmount);  
                System.out.println(bill.processPayment(cheque));  
        }  
    }  
}
```

```
        break;

    case "credit" : System.out.println("Enter the credit card number:");

        int creditNumber = sc.nextInt();

        System.out.println("Enter the card type(silver,gold,platinum):");

        String cardType=sc.next();

        Credit credit=new Credit();

        credit.setCardType(cardType);

        credit.setCreditCardNo(creditNumber);

        credit.setDueAmount(dueAmount);

        System.out.println(bill.processPayment(credit));

        break;

    default:

        break;

    }

    sc.close();

}

}

}
```

## POWER PROGRESS

### Main.java\*

```
import java.util.*;  
  
public class Main  
{  
    public static void main(String[] args)  
    {  
        Scanner sc=new Scanner(System.in);  
        //Fill the code  
        int m=sc.nextInt();  
        if(m<=0){  
            System.out.println(""+m+" is an invalid");  
            return;  
        }  
        int n=sc.nextInt();  
        if(n<=0){  
            System.out.println(""+n+" is an invalid");  
            return;  
        }  
        if(m>=n){  
            System.out.println(""+m+" is not less than "+n);  
            return;  
        }  
  
        for(int i=1;i<=n;i++){  
            System.out.print((int)Math.pow(m,i)+" ");  
        }  
    }  
}
```

## PRIME NUMBERS ENDING WITH 1

### Main.java\*

```
import java.util.Scanner;

public class Main {

    public static void main(String args[]) {

        Scanner sc = new Scanner(System.in);

        // fill the code

        int low, high;
        int last=0;
        int flag = 0;

        System.out.println("Enter the first number");
        low = sc.nextInt();

        System.out.println("Enter the last number");
        high = sc.nextInt();

        if (low > high || low < 0 || high < 0 || low == high)
            ;
        else {
            int i = low;
            high = high + 30;

            while (i <= high-1) {

                int x = i % 10;

                for (int j = 2; j <= i / 2; j++) {
                    if (i % j != 0 && x == 1) {
                        flag = 1;
                    } else {
                        flag = 0;
                        break;
                    }
                }

                if (flag == 1 )
                    System.out.println(i);
                i++;
            }
        }
    }
}
```

## SINGAPORE TOURISM

### Main.java\*

```
import java.util.*;  
  
public class Main  
{  
    public static void main(String[] args)  
    {  
        Scanner sc=new Scanner(System.in);  
        Map<String,Integer> map=new HashMap<>();  
        map.put("BEACH",270);  
        map.put("PILGRIMAGE",350);  
        map.put("HERITAGE",430);  
        map.put("HILLS",780);  
        map.put("FALLS",1200);  
        map.put("ADVENTURES",4500);  
        System.out.println("Enter the Passenger Name");  
        String pname=sc.next();  
        System.out.println("Enter the Place");  
        String name=sc.next();  
        if(!map.containsKey(name.toUpperCase()))  
        {  
            System.out.println(name+" is an invalid place");  
        }  
        else  
        {  
            System.out.println("Enter the no of Days");  
            int nod=sc.nextInt();  
            if(nod<=0)  
            {  
                System.out.println(nod+" is an invalid days");  
            }  
            else  
            {  
                System.out.println("Enter the no of Tickets");  
            }  
        }  
    }  
}
```

```
int not=sc.nextInt();
if(not<=0)
{
    System.out.println(not+" is an invalid tickets");
}
else
{
    double d=(double)map.get(name.toUpperCase());
    double totalcost=d*(double)not*(double)nod;
    if(totalcost>=1000)
    {
        totalcost=totalcost-((totalcost*15)/100);
    }
    System.out.printf("Bill Amount is %.2f", totalcost);
}

}

//Fill the code
}
```

## SUBSTITUTION CYPHER TECHNIQUE

### Main.java\*

```
import java.util.Scanner;

public class Main {

    public static void main (String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the Encrypted text: ");
        String code= sc.nextLine();
        int shift = 7;
        int f=0;

        String decryptMessage = " ";
        for(int i = 0; i < code.length(); i++){
            char alpha = code.charAt(i);
            if(alpha >='a' && alpha <= 'z'){
                f=1;
                alpha=(char)(alpha - shift);
                if(alpha < 'a'){
                    alpha = (char)(alpha - 'a'+'z'+1);
                }
                decryptMessage=decryptMessage+alpha;
            }
            else if (alpha >='A' && alpha <= 'Z'){
                f=1;
                alpha=(char)(alpha - shift);
                if(alpha < 'A'){
                    alpha = (char)(alpha - 'A'+'Z'+1);
                }
                decryptMessage=decryptMessage+alpha;
            }
            else if(alpha == ' '){
                decryptMessage=decryptMessage+alpha;
            }
        }
    }
}
```

```
    }

}

if(decryptMessage.length() == 0 || f == 0){

    System.out.println("No hidden Message");

    System.exit(0);

}

System.out.println("Decrpted Text:\n"+decryptMessage);

}
```

## ZEE ZEE BANK

### Account.java\*

```
public class Account {  
    long accountNumber;  
    double balanceAmount;  
  
    public Account(long accno, double bal){  
        super();  
        this.accountNumber=accno;  
        this.balanceAmount=bal;  
    }  
    public long getAccountNumber(){  
        return accountNumber;  
    }  
    public void setAccountNumber(long accno){  
        this.accountNumber=accno;  
    }  
    public double getBalanceAmount(){  
        return balanceAmount;  
    }  
    public void setBalanceAmount(double bal) {  
        this.balanceAmount=bal;  
    }  
    public void deposit(double depositAmt){  
        float total=(float)(balanceAmount+depositAmt);  
        balanceAmount=total;  
    }  
    public boolean withdraw(double withdrawAmt){  
        float total;  
        if(withdrawAmt>balanceAmount){  
            System.out.println("Insufficient balance");  
  
            return false;  
        }  
    }  
}
```

```
 }else{
    total=(float)(balanceAmount-withdrawAmt);
    setBalanceAmount(total);
    return true;
}
}
```

### Main.java\*

```
import java.util.Scanner;

public class Main{
    static Account ac=new Account(0, 0);
    public static void main (String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the account number:");
        ac.setAccountNumber(sc.nextLong());
        System.out.println("Enter the available amount in the account:");
        ac.setBalanceAmount(sc.nextDouble());
        System.out.println("Enter the amount to be deposited:");
        ac.deposit(sc.nextDouble());
        System.out.printf("Available balance is:%.2f",ac.getBalanceAmount());
        System.out.println();
        System.out.println("Enter the amount to be withdrawn:");
        ac.withdraw(sc.nextDouble());
        System.out.printf("Available balance is:%.2f",ac.getBalanceAmount());
        //Fill the code
    }
}
```

## THE NEXT RECHARGE DATE

### Main.java\*

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Main {

    public static void main(String [] args)throws Exception {
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Recharged date");
        String date=br.readLine();
        String currentDate="29/10/2019";
        if(Main.isValidFormat(date)&&(Main.dateCompare(date,currentDate))){
            System.out.println("Validity days");
            int days=Integer.parseInt(br.readLine());
            if(days>0)
                System.out.println(Main.futureDate(date,days));
            else
                System.out.println(days+ "is not a valid days");
        }
        else
            System.out.println(date+ "is not a valid date");
    }

    public static boolean isValidFormat(String date){
        String regex="^(3[01]|12[0-9]|0[1-9])/([1-2]|0[1-9])/[0-9]{4}$";
        Pattern pattern=Pattern.compile(regex);
        Matcher matcher=pattern.matcher((CharSequence)date);
        return matcher.matches();
    }
}
```

```
}

public static boolean dateCompare(String date1,String date2) throws ParseException{
    SimpleDateFormat sdfformat=new SimpleDateFormat("dd/MM/yyyy");
    Date d1=sdfformat.parse(date1);
    Date d2=sdfformat.parse(date2);
    if((d1.compareTo(d2)<0)|| (d1.compareTo(d2)==0))
        return true;
    else
        return false;
}

public static String futureDate(String date,int days){
    Calendar c=Calendar.getInstance();
    SimpleDateFormat sdfformat=new SimpleDateFormat("dd/MM/yyyy");
    try{
        Date mydate=sdfformat.parse(date);
        c.setTime(mydate);
        c.add(Calendar.DATE, days);
    }catch(ParseException e){
        e.printStackTrace();
    }
    String toDate=sdfformat.format(c.getTime());
    return toDate;
}
}
```

## A New You Spa

### DiamondMembers.java\*

```
public class DiamondMembers extends Members{

    // Fill the code

    public DiamondMembers(String customerId,String customerName,long mobileNumber,String memberType,String emailId){

        super(customerId,customerName,numberOfMembers,memberType,emailId);

        /*this.customerId = customerId;
        this.customerName = customerName;
        this.mobileNumber = mobileNumber;
        this.memberType = memberType;
        this.emailId = emailId;*/

    }

    public boolean validateCustomerId(){

        // Fill the code

        boolean b=true;

        String s1 = this.customerId.toUpperCase();

        String regex="[DIAMOND]{7}[0-9]{3}";

        if(s1.matches(regex)){

            b=true;

        }

        else{

            b=false;

        }

        return b;

    }

    public double calculateDiscount(double purchaseAmount){

        // Fill the code

    }

}
```

```
        double discount=purchaseAmount*0.45;
        double updateamount=purchaseAmount-discount;
        return updateamount;
    }

}
```

### **GoldMembers.java\***

```
public class GoldMembers extends Members {
    public GoldMembers(String customerId,String customerName,long mobileNumber,String memberType,String emailId){
        super(customerId,customerName,mobileNumber,memberType,emailId);
    }
    // Fill the code
    public boolean validateCustomerId(){
        boolean b=true;
        String s1 = this.customerId.toUpperCase();
        String regex="[GOLD]{4}[0-9]{3}";
        if(s1.matches(regex)){
            b=true;
        }
        else{
            b=false;
        }
        return b;
    }
    // Fill the code
}
public double calculateDiscount(double purchaseAmount){
    // Fill the code
    double discount=purchaseAmount*0.15;
    double updateamount=purchaseAmount-discount;
    return updateamount;
}
```

## **Members.java\***

```
abstract public class Members {  
    protected String customerId;  
    protected String customerName;  
    protected long mobileNumber;  
    protected String memberType;  
    protected String emailId;  
  
    abstract public double calculateDiscount(double purchaseAmount);  
  
    public String getCustomerId() {  
        return customerId;  
    }  
    public void setCustomerId(String customerId) {  
        this.customerId = customerId;  
    }  
    public String getCustomerName() {  
        return customerName;  
    }  
    public void setCustomerName(String customerName) {  
        this.customerName = customerName;  
    }  
    public long getMobileNumber() {  
        return mobileNumber;  
    }  
    public void setMobileNumber(long mobileNumber) {  
        this.mobileNumber = mobileNumber;  
    }  
  
    public String getMemberType() {  
        return memberType;  
    }  
    public void setMemberType(String memberType) {
```

```

        this.memberType = memberType;
    }

    public String getEmailId() {
        return emailId;
    }

    public void setEmailId(String emailId) {
        this.emailId = emailId;
    }

    public Members(String customerId, String customerName, long mobileNumber, String memberType, String
emailId) {
        this.customerId = customerId;
        this.customerName = customerName;
        this.mobileNumber = mobileNumber;
        this.memberType = memberType;
        this.emailId = emailId;
    }

}

```

### **PlatinumMembers.java\***

```

public class PlatinumMembers extends Members {

    // Fill the code

    public PlatinumMembers(String customerId, String customerName, long mobileNumber, String
memberType, String emailId) {
        super(customerId, customerName, mobileNumber, memberType, emailId);
        /*customerId = customerId;
        customerName = customerName;
        mobileNumber = mobileNumber;
        memberType = memberType;
        emailId = emailId;
    */

}

```

```

public boolean validateCustomerId(){
    // Fill the code
    boolean b=true;
    String s1 = this.customerId.toUpperCase();
    String regex="[PLATINUM]{8}[0-9]{3}";
    if(s1.matches(regex)){
        b=true;
    }
    else{
        b=false;
    }

    return b;
}

public double calculateDiscount(double purchaseAmount){
    // Fill the code
    double discount=purchaseAmount*0.3;
    double updateamount=purchaseAmount-discount;
    return updateamount;
}

}

```

### **UserInterface.java\***

```

import java.util.Scanner;

public class UserInterface {

    public static void main(String[] args){

        Scanner sc=new Scanner(System.in);
        System.out.println("Enter Customer Id");
    }
}

```

```

String cid=sc.nextLine();
System.out.println("Enter Customer name");
String cname=sc.nextLine();
System.out.println("Enter mobile number");
long mob=sc.nextLong();
sc.nextLine();
System.out.println("Enter Member type");
String mem=sc.nextLine();
System.out.println("Enter Email Id");
String email=sc.nextLine();
System.out.println("Enter amount Purchased");
double amount=sc.nextDouble();
DiamondMembers d=new DiamondMembers(cid,cname,mob,mem,email);
GoldMembers g=new GoldMembers(cid,cname,mob,mem,email);
PlatinumMembers p=new PlatinumMembers(cid,cname,mob,mem,email);

double res=0.0;
if(d.validateCustomerId()){
    res= d.calculateDiscount(amount);
    System.out.println("Name :" +d.getCustomerName());
    System.out.println("Id :" +d.getCustomerId());
    System.out.println("Email Id :" +d.getEmailId());
    System.out.println("Amount to be paid :" +res);
}

} else if(g.validateCustomerId()){
    res= g.calculateDiscount(amount);
    System.out.println("Name :" +g.getCustomerName());
    System.out.println("Id :" +g.getCustomerId());
    System.out.println("Email Id :" +g.getEmailId());
    System.out.println("Amount to be paid :" +res);
}

} else if(p.validateCustomerId()){
    res= p.calculateDiscount(amount);
}

```

```
System.out.println("Name :" + p.getCustomerName());  
System.out.println("Id :" + p.getCustomerId());  
System.out.println("Email Id :" + p.getEmailId());  
System.out.println("Amount to be paid :" + res);  
  
} else{  
    System.out.println("Provide a valid Customer Id");  
}  
  
// Fill the code  
}
```

## BATTING AVERAGE

### UserInterface.java\*

```
package com.ui;

import com.utility.Player;

import java.util.ArrayList;

import java.util.Scanner;

public class UserInterface {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        Player player=new Player();

        player.setScoreList(new ArrayList<>());

        boolean flag=true;

        while(flag)

        {

            System.out.println("1. Add Runs Scored");

            System.out.println("2. Calculate average runs scored");

            System.out.println("3. Exit");

            System.out.println("Enter your choice");

            int choice=sc.nextInt();

            switch(choice)

            {

                case 1:

                    System.out.println("Enter the runs scored");

                    int runScored=sc.nextInt();

                    player.addScoreDetails(runScored);

                    break;

                }

                case 2:

                    System.out.println("Average runs secured");

                    System.out.println(player.getAverageRunScored());

                    break;

            }

        }

    }

}
```

```
        case 3: {
            System.out.println("Thank you for using the Application");
            flag=false;
            break;
        }
    }
}

}
```

### **Player.java\***

```
package com.utility;

import java.util.List;

public class Player {

    private List<Integer> scoreList;

    public List<Integer> getScoreList() {
        return scoreList;
    }

    public void setScoreList(List<Integer> scoreList) {
        this.scoreList = scoreList;
    }

    //This method should add the runScored passed as the argument into the scoreList
    public void addScoreDetails(int runScored) {
```

```
// fill the code  
scoreList.add(runScored);  
  
}  
  
}
```

/\* This method should return the average runs scored by the player

Average runs can be calculated based on the sum of all runScored available in the scoreList divided by the number of elements in the scoreList.

For Example:

```
List contains[150,50,50]  
average runs secured=(150+50+50)/3=83.3333333333333  
so this method should return 83.3333333333333  
If list is empty return 0  
*/  
  
public double getAverageRunScored() {  
  
    // fill the code  
    if(scoreList.isEmpty()) {  
        return 0.0;  
    }  
    int size=scoreList.size();  
    int totalScore=0;  
    for(int score : scoreList)  
    {  
        totalScore+=score;  
    }  
    return (double) totalScore / (double) size;  
}  
  
}
```

## Change The Cash

### Main.java\*

```
import java.util.*;  
  
public class Main{  
  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        String a = sc.next();  
        if(a.length() < 3) {  
            System.out.println("String length of " + a + " is too short");  
            return;  
        }  
        else if(a.length() > 10) {  
            System.out.println("String length of " + a + " is too long");  
            return;  
        }  
  
        char[] arr = a.toCharArray();  
        char[] arr1 = new char[arr.length];  
        int j = 0;  
        for(int i = 0; i < a.length(); i++) {  
            if((arr[i]<65 || ((arr[i]>90) && (arr[i]<97)) || arr[i]>122)) {  
                arr1[j++] = arr[i];  
            }  
        }  
        if(j!=0) {  
            System.out.print("String should not contain ");  
            for(int i = 0; i<j; i++) {  
                System.out.print(arr1[i]);  
            }  
            return;  
        }  
        char b = sc.next().charAt(0);  
    }  
}
```

```
int present = 0;
for(int i = 0; i<a.length(); i++) {
    if(arr[i] == Character.toUpperCase(b)) {
        arr[i] = Character.toLowerCase(b);
        present = 1;
    }
}
if(present == 0) {
    System.out.println("Character " + b + " is not found");
}
else {
    for(int i = 0; i <a.length(); i++) {
        System.out.print(arr[i]);
    }
}
}
```

## Check Number Type

### NumberType.java\*

```
public interface NumberType
{
    public boolean checkNumberType(int n);
}
```

### NumberTypeUtility.java\*

```
import java.util.Scanner;

public class NumberTypeUtility
{
    public static NumberType isOdd()
    {
        return n -> n%2 != 0;
    }

    public static void main (String[] args)
    {

        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        if(isOdd().checkNumberType(n))
        {
            System.out.println(n+" is odd");
        }
        else
        {
            System.out.println(n+" is not odd");
        }
    }
}
```

## Cheque Payment Process

### PaymentDao.java\*

```
package com.cts.paymentProcess.dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

import com.cts.paymentProcess.model.Payment;
import com.cts.paymentProcess.util.DatabaseUtil;

public class PaymentDao {

    private Connection connection;

    public List<Payment> getAllRecord(){

        connection=DatabaseUtil.getConnection();
        PreparedStatement statement=null;
        ResultSet resultSet=null;
        List<Payment> paymentList=new ArrayList<Payment>();
        try {
            statement=connection.prepareStatement("select * from cheque_payments");
            resultSet=statement.executeQuery();
            while(resultSet.next()){
                Payment payment =new Payment();
                payment.setCustomerNumber(resultSet.getInt("customerNumber"));
                payment.setChequeNumber(resultSet.getString("chequeNumber"));
                payment.setPaymentDate(resultSet.getDate("paymentDate"));
                payment.setAmount(resultSet.getInt("amount"));
                paymentList.add(payment);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return paymentList;
    }
}
```

```
        }

    } catch (SQLException e) {

        e.printStackTrace();

    }finally{

        try{

            resultSet.close();

            statement.close();

        }catch(Exception e){

            e.printStackTrace();

        }

    }

}

return paymentList;

}
```

### **Payment.java\***

```
package com.cts.paymentProcess.model;

import java.util.Date;

public class Payment {

    private int customerNumber;

    private String chequeNumber;

    private Date paymentDate;

    private int amount;

    public int getCustomerNumber() {

        return customerNumber;

    }

}
```

```
public void setCustomerNumber(int customerNumber) {
    this.customerNumber = customerNumber;
}

public String getChequeNumber() {
    return chequeNumber;
}

public void setChequeNumber(String chequeNumber) {
    this.chequeNumber = chequeNumber;
}

public Date getPaymentDate() {
    return paymentDate;
}

public void setPaymentDate(Date paymentDate) {
    this.paymentDate = paymentDate;
}

public int getAmount() {
    return amount;
}

public void setAmount(int amount) {
    this.amount = amount;
}

@Override
public String toString() {

    return String.format("%15s%15s%15s%15s", customerNumber, chequeNumber, paymentDate,
amount);
}

}
```

**PaymentService.java\***

```
package com.cts.paymentProcess.service;

import java.util.*;
import java.util.Calendar;
import java.util.List;
import java.util.stream.Collectors;

import com.cts.paymentProcess.dao.PaymentDao;
import com.cts.paymentProcess.model.Payment;

public class PaymentService {

    private PaymentDao paymentDao=new PaymentDao();

    public List<Payment> findCustomerByNumber(int customerNumber){

        List<Payment> list=paymentDao.getAllRecord();

        List<Payment> list2 = new ArrayList<>();
        list2 = list.stream().filter(x->x.getCustomerNumber()==customerNumber).collect(Collectors.toList());

        return list2;
    }

    public List<Payment> findCustomerByYear(int year){

        List<Payment> list=paymentDao.getAllRecord();

        List<Payment> list2 = new ArrayList<>();
        list2 = list.stream().filter(x->x.getPaymentDate().getYear()==(year-1900)).sorted(Comparator.comparingInt(Payment::getAmount)).collect(Collectors.toList());

        return list2;
    }
}
```

**SkeletonValidator.java\***

```
package com.cts.paymentProcess.skeletonValidator;

import java.lang.reflect.Method;
import java.util.logging.Level;
import java.util.logging.Logger;

public class SkeletonValidator {

    public SkeletonValidator(){

        validateClassName("com.cts.paymentProcess.dao.PaymentDao");
        validateMethodSignature("getAllRecord:java.util.List","com.cts.paymentProcess.dao.PaymentDao");

        validateClassName("com.cts.paymentProcess.model.Payment");
        validateMethodSignature("toString:java.lang.String","com.cts.paymentProcess.model.Payment");

        validateClassName("com.cts.paymentProcess.service.PaymentService");

        validateMethodSignature("findCustomerByNumber:java.util.List,findCustomerByYear:java.util.List","com.cts.
paymentProcess.service.PaymentService");

        validateClassName("com.cts.paymentProcess.util.DatabaseUtil");

        validateMethodSignature("getConnection:java.sql.Connection","com.cts.paymentProcess.util.DatabaseUtil")
    ;

    }

    private static final Logger LOG = Logger.getLogger("SkeletonValidator");

    protected final boolean validateClassName(String className) {
```

```

boolean iscorrect = false;

try {
    Class.forName(className);
    iscorrect = true;
    LOG.info("Class Name " + className + " is correct");
}

} catch (ClassNotFoundException e) {
    LOG.log(Level.SEVERE, "You have changed either the " + "class name/package. Use the
correct package "
        + "and class name as provided in the skeleton");

} catch (Exception e) {
    LOG.log(Level.SEVERE,
        "There is an error in validating the " + "Class Name. Please manually verify
that the "
        + "Class name is same as skeleton before uploading");
}

return iscorrect;
}

protected final void validateMethodSignature(String methodWithExcptn, String className) {
    Class cls = null;
    try {
        String[] actualmethods = methodWithExcptn.split(",");
        boolean errorFlag = false;
        String[] methodSignature;
        String methodName = null;
        String returnType = null;

        for (String singleMethod : actualmethods) {
            boolean foundMethod = false;
            methodSignature = singleMethod.split(":");

```

```

methodName = methodSignature[0];
returnType = methodSignature[1];
cls = Class.forName(className);
Method[] methods = cls.getMethods();
for (Method findMethod : methods) {
    if (methodName.equals(findMethod.getName())) {
        foundMethod = true;
        if (!(findMethod.getReturnType().getName().equals(returnType))) {
            errorFlag = true;
            LOG.log(Level.SEVERE, " You have changed the " + "return
type in '" + methodName
                                         + "' method. Please stick to the " + "skeleton
provided");
    }
}
} else {
    LOG.info("Method signature of " + methodName + " is
valid");
}
}

if (!foundMethod) {
    errorFlag = true;
    LOG.log(Level.SEVERE, " Unable to find the given public method " +
methodName
                                         + ". Do not change the " + "given public method name. " +
"Verify it with the skeleton");
}
}

if (!errorFlag) {
    LOG.info("Method signature is valid");
}
}

} catch (Exception e) {
    LOG.log(Level.SEVERE,

```

```
" There is an error in validating the " + "method structure. Please manually  
verify that the "  
+ "Method signature is same as the skeleton before  
uploading");  
}  
}  
}
```

### **DatabaseUtil.java\***

```
package com.cts.paymentProcess.util;  
  
import java.io.FileInputStream;  
import java.io.IOException;  
import java.io.InputStream;  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.SQLException;  
import java.util.Properties;  
import java.util.ResourceBundle;  
  
public class DatabaseUtil {  
  
    private DatabaseUtil() {  
    }  
  
    private static Connection con = null;  
    private static Properties props = new Properties();  
  
    //ENSURE YOU DON'T CHANGE THE BELOW CODE WHEN YOU SUBMIT  
    public static Connection getConnection() {  
        try{
```

```

FileInputStream fis = null;
fis = new FileInputStream("resource/database.properties");
props.load(fis);

// load the Driver Class
try {
    Class.forName(props.getProperty("DB_DRIVER_CLASS"));
} catch (ClassNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

// create the connection now
try {
    con =
DriverManager.getConnection(props.getProperty("DB_URL"),props.getProperty("DB_USERNAME"),props.getProperty("DB_PASSWORD"));
} catch (SQLException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}

catch(IOException e){
    e.printStackTrace();
}

return con;
}

}

```

**App.java(Main)\***

```
package com.cts.paymentProcess;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.List;
import java.util.Scanner;

import com.cts.paymentProcess.model.Payment;
import com.cts.paymentProcess.service.PaymentService;
import com.cts.paymentProcess.skeletonValidator.SkeletonValidator;

public class App {

    public static void main(String[] args) throws ParseException {

        new SkeletonValidator();

        Payment payment=null;
        Scanner scanner=new Scanner(System.in);

        do{
            System.out.println("Select Option:");
            System.out.println("1.Customer list\n2.Yearly Customer List\n3.Exit");
            int choice=scanner.nextInt();
            switch(choice){

                case 1:System.out.println("Enter customer number");
                int number=scanner.nextInt();
                List<Payment> numberList=new PaymentService().findCustomerByNumber(number);
                if(numberList.size()==0){
                    System.out.println("\nNo Records Found\n");
                }else{


```

```
System.out.format("%15s%15s%15s%15s\n","Customer Number","Cheque Number","Payment Date","Amount");

    numberList.stream()

        .forEach(System.out::println);

    }

    break;

    case 2:System.out.println("Enter year");

    int year=scanner.nextInt();

    List<Payment> yearList=new PaymentService().findCustomerByYear(year);

    if(yearList.size()==0){

        System.out.println("\nNo Records Found\n");

    }else{

        System.out.format("%15s%15s%15s%15s\n","Customer Number","Cheque Number","Payment Date","Amount");

        yearList.stream()

            .forEach(System.out::println);

        }

        break;

        case 3:System.exit(0);

        default:System.out.println("\nWrong Choice\n");

    }

}

}

}

}
```

## Employee Eligibility for Promotion

**Main.java\***

```
import java.time.format.DateTimeFormatter;

import java.time.temporal.ChronoUnit;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.time.*;

import java.util.TreeMap;
import java.util.Date;
import java.util.Map;

import java.util.Map.Entry;

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        //System.out.println("In-time");
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
        LocalDate currTime = LocalDate.of(2019, 01, 01);
        String fdt=currTime.format(formatter);
        //
        System.out.println(fdt);
        int no = sc.nextInt();
        Map<String, String> m = new TreeMap<>();
        for (int i = 1; i <= no; i++) {
            String id = sc.next();
            String date = sc.next();
            m.put(id, date);
        }
    }
}
```

```

int count = 0;
int val = 0;

for (Entry<String, String> entry : m.entrySet()) {
    if (entry.getValue().matches("(0[1-9]|1[1-2][0-9]|3[0-1])/"
        "(0[1-9]|1[0-2])/[0-9]{4}"))
    {
        val++;
        LocalDate InTime = LocalDate.parse(entry.getValue(), formatter);
        Period in = Period.between(InTime, currTime);
        long lin = in.get(ChronoUnit.YEARS);
        if (lin >= 5)
        {
            count++;
            System.out.println(entry.getKey());
        }
    }
    else
    {
        System.out.println("Invalid date format");
        break;
    }
}
if (count == 0 && val == no)
    System.out.println("No one is eligible");
}
}

```

## Exam Scheduler

### AssessmentDao.java\*

```
package com.cts.cc.dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.time.Duration;
import java.time.LocalDate;
import java.time.LocalTime;
import java.time.Period;
import java.util.List;
import java.sql.*;

import com.cts.cc.model.Assessment;
import com.cts.cc.util.DatabaseUtil;

public class AssessmentDAO {

    public int uploadAssessments(List<Assessment> assessments) throws Exception {
        if(assessments==null || assessments.isEmpty()) {
            throw new Exception("List is Empty");
        }
        int rowsCount = 0;

        //Write your logic here...

        try{
            Connection con = DatabaseUtil.getConnection();
            for(Assessment a:assessments)
            {
                PreparedStatement st = con.prepareStatement("insert into assessment values(?,?,?,?,?,?)");
                st.setString(1,a.getExamCode());
                st.setString(2,a.getExamTitle());
                st.setString(3,a.getExamDate().toString());
                st.setString(4,a.getExamTime().toString());
            }
        }
    }
}
```

```

        st.setString(5,a.getExamDuration().toString());
        st.setString(6,a.getEvalDays().toString());
        int rs=st.executeUpdate();
        if(rs!=-1)
            rowsCount=rowsCount+1;
    }

} catch(SQLException e){

}

return rowsCount;
}

public Assessment findAssessment(String code) throws Exception {
    Assessment assessment = null;
    Connection conn = DatabaseUtil.getConnection();
    String sql = "SELECT * FROM assessment where code=?";
    PreparedStatement ps = conn.prepareStatement(sql);
    ps.setString(1, code);
    ResultSet rs = ps.executeQuery();
    if(rs.next()) {
        assessment = new Assessment();
        assessment.setExamCode(rs.getString(1));
        assessment.setExamTitle(rs.getString(2));
        assessment.setExamDate(LocalDate.parse(rs.getString(3)));
        assessment.setExamTime(LocalTime.parse(rs.getString(4)));
        assessment.setExamDuration(Duration.parse(rs.getString(5)));
        assessment.setEvalDays(Period.parse(rs.getString(6)));
    }

    return assessment;
}
}

```

**GenerateAssessmentFunction.java\***

```
package com.cts.cc.functions;

import java.time.Duration;
import java.time.LocalDate;
import java.time.LocalTime;
import java.time.Period;
import java.util.function.Function;

import com.cts.cc.model.Assessment;

public class GenerateAssessmentFunction implements Function<String, Assessment>{

    @Override
    public Assessment apply(String t) {
        //Write your code here...
        String temp[] = t.split(",");
        Assessment a = new
        Assessment(temp[0],temp[1],LocalDate.parse(temp[2]),LocalTime.parse(temp[3]),Duration.parse(temp[4]),Period.p
        arse(temp[5]));
        return a;
    }
}
```

**Assessment.java\***

```
package com.cts.cc.model;

import java.time.Duration;
import java.time.LocalDate;
import java.time.LocalTime;
import java.time.Period;
import java.time.format.DateTimeFormatter;

public class Assessment {
```

```
private String examCode;
private String examTitle;
private LocalDate examDate;
private LocalTime examTime;
private Duration examDuration;
private Period evalDays;

public Assessment(String examCode, String examTitle, LocalDate examDate, LocalTime examTime, Duration
examDuration,
                    Period evalDays) {
    super();
    this.examCode = examCode;
    this.examTitle = examTitle;
    this.examDate = examDate;
    this.examTime = examTime;
    this.examDuration = examDuration;
    this.evalDays = evalDays;
}

public Assessment() {
}

public String getExamCode() {
    return examCode;
}

public void setExamCode(String examCode) {
    this.examCode = examCode;
}

public String getExamTitle() {
    return examTitle;
}
```

```
public void setExamTitle(String examTitle) {
    this.examTitle = examTitle;
}

public LocalDate getExamDate() {
    return examDate;
}

public void setExamDate(LocalDate examDate) {
    this.examDate = examDate;
}

public LocalTime getExamTime() {
    return examTime;
}

public void setExamTime(LocalTime examTime) {
    this.examTime = examTime;
}

public Duration getExamDuration() {
    return examDuration;
}

public void setExamDuration(Duration examDuration) {
    this.examDuration = examDuration;
}

public Period getEvalDays() {
    return evalDays;
}

public void setEvalDays(Period evalDays) {
    this.evalDays = evalDays;
}
```

```
}

public void printDetails() {
    DateTimeFormatter date1=DateTimeFormatter.ofPattern("dd-MMM-y");
    DateTimeFormatter date2=DateTimeFormatter.ofPattern("HH:mm");
    LocalTime t=examTime.plus(examDuration);
    String d=DateTimeFormatter.ofPattern("HH:mm").format(t);
    LocalDate t1=examDate.plus(evalDays);
    String d1=DateTimeFormatter.ofPattern("dd-MMM-y").format(t1);
    System.out.println("Assessment Code: "+examCode);
    System.out.println("Title: "+examTitle);
    System.out.println("Assessment Date: "+examDate.format(date1));
    System.out.println("Start Time: "+examTime.format(date2));
    System.out.println("End Time: "+d);
    System.out.println("Result Date: "+d1);
    //Write your code here...
}
```

```
}
```

#### **DatabaseUtil.java\***

```
package com.cts.cc.util;

import java.io.FileInputStream;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;

public class DatabaseUtil {

    private static Connection con = null;
```

```

private static Properties props = new Properties();

//ENSURE YOU DON'T CHANGE THE BELOW CODE WHEN YOU SUBMIT

public static Connection getConnection() throws ClassNotFoundException, SQLException {
    try{
        FileInputStream fis = null;
        fis = new FileInputStream("resource/connection.properties");
        props.load(fis);

        // load the Driver Class
        Class.forName(props.getProperty("DB_DRIVER_CLASS"));

        // create the connection now
        con =
        DriverManager.getConnection(props.getProperty("DB_URL"),props.getProperty("DB_USERNAME"),props.getProperty("DB_PASSWORD"));
    }
    catch(IOException e){
        e.printStackTrace();
    }
    return con;
}
}

```

### **FileUtil.java\***

```

package com.cts.cc.util;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;
import java.util.function.Function;

```

```

import java.util.stream.Collectors;
import java.util.stream.Stream;
import java.io.*;
import java.util.*;

import com.cts.cc.functions.GenerateAssessmentFunction;
import com.cts.cc.model.Assessment;

public class FileUtils {

    public static List<Assessment> loadData(String fileName) throws IOException {
        List<Assessment> list = null;
        Function<String, Assessment> function = new GenerateAssessmentFunction();
        BufferedReader br=new BufferedReader(new FileReader(fileName));
        String line="";
        list=new ArrayList<Assessment>();
        while((line=br.readLine())!=null)
        {
            list.add(function.apply(line));
        }
        //Write your code here...
    }

    return list;
}
}

```

### **SkeletonValidator.java\***

```

package com.cts.cc;

import java.lang.reflect.Constructor;
import java.lang.reflect.Method;
import java.sql.Connection;
import java.time.Duration;
import java.time.LocalDate;
import java.time.LocalTime;

```

```
import java.time.Period;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

import com.cts.cc.model.Assessment;

public class SkeletonValidator {
    private static final Logger LOG = Logger.getLogger("SkeletonValidator");

    public SkeletonValidator() {
        String assessmentClass = "com.cts.cc.model.Assessment";
        String assessmentDAOClass = "com.cts.cc.dao.AssessmentDAO";
        String functionalClass = "com.cts.cc.functions.GenerateAssessmentFunction";
        String databaseUtilClass = "com.cts.cc.util.DatabaseUtil";
        String fileUtilClass = "com.cts.cc.util.FileUtil";

        Class[] assessmentParams = { String.class, String.class, LocalDate.class, LocalTime.class,
Duration.class,
                                         Period.class };

        String[] assessmentFields = { "examCode", "examTitle", "examDate", "examTime", "examDuration",
"evalDays" };

        testClass(assessmentClass, assessmentParams);
        testClass(assessmentDAOClass, null);
        testClass(functionalClass, null);
        testClass(databaseUtilClass, null);
        testClass(fileUtilClass, null);

        testFields(assessmentClass, assessmentFields);
        testMethods(assessmentClass, "printDetails", null, null);
        testMethods(assessmentDAOClass, "uploadAssessments", new Class[] { List.class }, boolean.class);
        testMethods(assessmentDAOClass, "findAssessment", new Class[] { String.class }, Assessment.class);
        testMethods(functionalClass, "apply", new Class[] { String.class }, Assessment.class);
    }
}
```

```

    testMethods(databaseUtilClass, "getConnection", null, Connection.class);
    testMethods(fileUtilClass, "loadData", new Class[] { String.class }, List.class);
}

public void testClass(String className, Class[] paramTypes) {
    try {
        Class classUnderTest = Class.forName(className);
        LOG.info("Class Name " + className + " is correct");
        Constructor<?> constructor = classUnderTest.getConstructor(paramTypes);
        constructor.equals(constructor);
        LOG.info(className + " Constructor is valid");
    } catch (ClassNotFoundException e) {
        LOG.log(Level.SEVERE, "You have changed either the class name/package. "
            + "Use the correct package and class name as provided in the skeleton");
    } catch (NoSuchMethodException e) {
        LOG.log(Level.SEVERE, " Unable to find the given constructor. "
            + "Do not change the given public constructor. " + "Verify it with the
skeleton");
    } catch (SecurityException e) {
        LOG.log(Level.SEVERE,
            "There is an error in validating the " + className + ". " + "Please verify the
skeleton manually");
    }
}

public void testFields(String className, String[] fields) {
    try {
        Class classUnderTest = Class.forName(className);
        for (String field : fields) {
            classUnderTest.getDeclaredField(field);
        }
        LOG.info("Fields in " + className + " are correct");
    } catch (ClassNotFoundException e) {
        LOG.log(Level.SEVERE, "You have changed either the class name/package. "
            + "Use the correct package and class name as provided in the skeleton");
    }
}

```

```

} catch (NoSuchFieldException e) {
    LOG.log(Level.SEVERE,
        "You have changed one/more field(s). " + "Use the field name(s) as provided
in the skeleton");

} catch (SecurityException e) {
    LOG.log(Level.SEVERE,
        "There is an error in validating the " + className + ". " + "Please verify the
skeleton manually");
}

}

public void testMethods(String className, String methodName, Class[] paramTypes, Class returnType) {
    try {
        Class classUnderTest = Class.forName(className);
        Method method = classUnderTest.getDeclaredMethod(methodName, paramTypes);
        Class retType = method.getReturnType();
        if (returnType != null && !retType.equals(returnType)) {
            LOG.log(Level.SEVERE, " You have changed the " + "return type in '" + methodName
                + "' method. Please stick to the " + "skeleton provided");
            throw new NoSuchMethodException();
        }
        LOG.info(methodName + " signature is valid.");
    } catch (ClassNotFoundException e) {
        LOG.log(Level.SEVERE, "You have changed either the class name/package. "
            + "Use the correct package and class name as provided in the skeleton");
    } catch (NoSuchMethodException e) {
        LOG.log(Level.SEVERE, "You have changed/removed method " + methodName + ". "
            + "Use the method signature as provided in the skeleton");
    } catch (SecurityException e) {
        LOG.log(Level.SEVERE,
            "There is an error in validating the " + className + ". " + "Please verify the
skeleton manually");
    }
}

```

**Main.java\***

```
package com.cts.cc;

import java.util.List;

import com.cts.cc.dao.AssessmentDAO;
import com.cts.cc.model.Assessment;
import com.cts.cc.util.FileUtil;

public class Main {

    public static void main(String[] args) {
        // CODE SKELETON - VALIDATION STARTS
        // DO NOT CHANGE THIS CODE
        new SkeletonValidator();
        // CODE SKELETON - VALIDATION ENDS

        try {
            List<Assessment> assessments = FileUtil.loadData("resource/data.txt");
            AssessmentDAO dao = new AssessmentDAO();
            dao.uploadAssessments(assessments);
            Assessment assessment = dao.findAssessment("ASEJE025");
            assessment.printDetails();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

## Book Details

### Main.java\*

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        //fill the code
        String string = sc.nextLine();
        if(string.length()==18)
        {
            String substr1 = string.substring(0,3);
            int i1 = Integer.parseInt(substr1);
            if(i1>100 && i1<104)
            {
                String substr2 = string.substring(3,7);
                int i2 = Integer.parseInt(substr2);
                if(i2>=1900 && i2<=2020)
                {
                    String substr3 = string.substring(7,12);
                    //System.out.println(substr3);
                    int i3 = Integer.parseInt(substr3);
                    if(i3>=10)
                    {
                        String substr4 = string.substring(12,13);
                        String substr5 = string.substring(13,18);

                        if((substr4.charAt(0)>='A'&&substr4.charAt(0)<='Z')|| (substr4.charAt(0)>='a'&&substr4.charAt(0)<='z'))
                        {

                            if((substr5.charAt(0)>='0'&&substr5.charAt(0)<='9')&&(substr5.charAt(1)>='0'&&substr5.charAt(1)<='9')&&
                            (substr5.charAt(2)>='0'&&substr5.charAt(2)<='9')&&(substr5.charAt(3)>='0'&&substr5.charAt(3)<='9')&&
                            (substr5.charAt(4)>='0'&&substr5.charAt(4)<='9'))
                            {

```

```
String substr6 = string.substring(12,18);

System.out.println("Department Code: "+substr1);

if(i1==101)

System.out.println("Department Name: "+"Accounting");

else if(i1==102)

System.out.println("Department Name: "+"Economics");

else if(i1==103)

System.out.println("Department Name: "+"Engineering");

System.out.println("Year of Publication: "+substr2);

System.out.println("Number of Pages: "+substr3);

System.out.println("Book Id: "+substr6);

}

else

{

    String substr6 = string.substring(12,18);

    System.out.printf("%s is invalid book id",substr6);

    System.out.printf("\n");

}

else

{

    String substr6 = string.substring(12,18);

    System.out.printf("%s is invalid book id",substr6);

    System.out.printf("\n");

}

else

{

    System.out.printf("%s are invalid pages",substr3);

    System.out.printf("\n");

}
```

```
    }

    else

    {

        System.out.printf("%d is invalid year",i2);

        System.out.printf("\n");

    }

}

else

{

    System.out.printf("%d is invalid department code",i1);

    System.out.printf("\n");

}

}

else

{

    System.out.printf("%s is invalid input",string);

    System.out.printf("\n");

}

}

}
```

## Find Membership Category

**Member.java\***

```
public class Member {  
  
    private String memberId;  
    private String memberName;  
    private String category;  
  
    public String getMemberId() {  
        return memberId;  
    }  
    public void setMemberId(String memberId) {  
        this.memberId = memberId;  
    }  
    public String getMemberName() {  
        return memberName;  
    }  
    public void setMemberName(String memberName) {  
        this.memberName = memberName;  
    }  
    public String getCategory() {  
        return category;  
    }  
    public void setCategory(String category) {  
        this.category = category;  
    }  
  
    public Member(String memberId, String memberName, String category) {  
        super();  
        this.memberId = memberId;  
        this.memberName = memberName;  
        this.category = category;  
    }  
}
```

```
}
```

### **ZeeShop.java\***

```
import java.util.List;

public class ZEEShop extends Thread
{
    private String memberCategory;
    private int count;
    private List<Member> memberList;

    public ZEEShop(String memberCategory, List<Member> memberList) {
        super();
        this.memberCategory = memberCategory;
        this.memberList = memberList;
    }

    public String getMemberCategory() {
        return memberCategory;
    }

    public void setMemberCategory(String memberCategory) {
        this.memberCategory = memberCategory;
    }
}
```

```
public int getCount() {
    return count;
}

public void setCount(int count) {
    this.count = count;
}

public List<Member> getMemberList() {
    return memberList;
}

public void setMemberList(List<Member> memberList) {
    this.memberList = memberList;
}

public void run()
{
    synchronized(this)
    {
        for(Member m:memberList)
        {
            if(m.getCategory().equals(memberCategory))
                count++;
        }
    }
}
```

```
}
```

```
}
```

```
}
```

### Main.java\*

```
import java.util.ArrayList;  
import java.util.List;  
import java.util.Scanner;
```

```
public class Main {
```

```
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
  
        List<Member> mList=new ArrayList<Member>();  
        System.out.println("Enter the number of Members:");  
        Scanner sc=new Scanner(System.in);  
        int tot=sc.nextInt();  
        String[] str=new String[tot];  
        for(int i=0;i<str.length;i++)  
        {  
            System.out.println("Enter the Member Details:");  
            str[i]=sc.next();  
        }
```

```
        Member m[]=new Member[tot];  
        for(int i=0;i<m.length;i++)  
        {  
            String s[]=str[i].split(":");
```

```

m[i]=new Member(s[0],s[1],s[2]);
mList.add(m[i]);

}

System.out.println("Enter the number of times Membership category needs to be searched:");
int tot1=sc.nextInt();
ZEEShop t1[]=new ZEEShop[tot1];
for(int i=0;i<tot1;i++)
{
    System.out.println("Enter the Category");
    String s1=sc.next();
    t1[i]=new ZEEShop(s1,mList);
    t1[i].start();

    //System.out.println(s1+" "+t1.getCount());
}

try {
    Thread.currentThread().sleep(2000);
} catch (InterruptedException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

for(ZEEShop s:t1)
{
    System.out.println(s.getMemberCategory()+":"+s.getCount());
}

}

```

## Go Hospitals

### InPatient.java\*

```
public class InPatient extends Patient{  
  
    private double roomRent;  
  
    public InPatient(String patientId, String patientName, long mobileNumber, String gender, double roomRent) {  
        super(patientId,patientName,mobileNumber,gender);  
        this.roomRent=roomRent;  
    }  
  
    public double getRoomRent() {  
        return roomRent;  
    }  
  
    public void setRoomRent(double roomRent) {  
        this.roomRent = roomRent;  
    }  
  
    // Fill the code  
  
    public double calculateTotalBill(int noOfDays,double medicinalBill){  
        // Fill the code  
        double bill_amount;  
        bill_amount=((this.roomRent*noOfDays)+medicinalBill);  
        return bill_amount;  
    }  
}
```

### OutPatient.java\*

```
public class OutPatient extends Patient{
```

```

private double consultingFee;

public OutPatient(String patientId, String patientName, long mobileNumber, String gender, double consultingFee)
{
    super(patientId,patientName,mobileNumber,gender);
    this.consultingFee=consultingFee;
}

public double getConsultingFee() {
    return consultingFee;
}

public void setConsultingFee(double consultingFee) {
    this.consultingFee = consultingFee;
}

// Fill the code

public double calculateTotalBill(double scanPay,double medicinalBill){
    // Fill the code
    double bill_amount;
    bill_amount=this.consultingFee+scanPay+medicinalBill;
    return bill_amount;
}

```

### Patient.java\*

```

public class Patient {

    protected String patientId;
    protected String patientName;

```

```
protected long mobileNumber;
protected String gender;

public Patient(String patientId, String patientName, long mobileNumber, String gender) {
    this.patientId = patientId;
    this.patientName = patientName;
    this.mobileNumber = mobileNumber;
    this.gender = gender;
}

public String getPatientId() {
    return patientId;
}

public void setPatientId(String patientId) {
    this.patientId = patientId;
}

public String getPatientName() {
    return patientName;
}

public void setPatientName(String patientName) {
    this.patientName = patientName;
}

public long getMobileNumber() {
    return mobileNumber;
}

public void setMobileNumber(long mobileNumber) {
    this.mobileNumber = mobileNumber;
}

public String getGender() {
    return gender;
}

public void setGender(String gender) {
    this.gender = gender;
}
```

```
}
```

```
}
```

### **UserInterface.java\***

```
import java.util.Scanner;

public class UserInterface {

    public static void main(String[] args){
        Scanner read=new Scanner(System.in);

        System.out.println("1.In Patient");
        System.out.println("1.Out Patient");
        System.out.println("Enter the choice");
        int ch=read.nextInt();
        System.out.println("Enter the details");
        System.out.println("Patient Id");
        String id=read.nextLine();
        System.out.println("Patient Name");
        String name=read.nextLine();
        read.nextLine();
        System.out.println("Phone Number");
        long num=read.nextLong();
        System.out.println("Gender");
        String gen=read.next();

        if(ch==1){

            System.out.println("Room Rent");
            double rent=read.nextDouble();
            InPatient in=new InPatient(id,name,num,gen,rent);
            System.out.println("Medicinal Bill");
        }
    }
}
```

```
double bill=read.nextDouble();

System.out.println("Number of Days of Stay");

int days=read.nextInt();

System.out.println("Amount to be paid "+in.calculateTotalBill(days,bill));

}

else{

System.out.println("Consultancy Fee");

double fee=read.nextDouble();

OutPatient out=new OutPatient(id,name,num,gen,fee);

System.out.println("Medicinal Bill");

double medbill=read.nextDouble();

System.out.println("Scan Pay");

double pay=read.nextDouble();

System.out.println("Amount to be paid "+out.calculateTotalBill(pay,medbill));

}

// Fill the code

}
```

## Grade Calculation

### GradeCalculator.java\*

```
public class GradeCalculator extends Thread {
```

```
    private String studName;
```

```
    private char result;
```

```
    private int[] marks;
```

```
    public String getStudName(){
```

```
        return studName;
```

```
}
```

```
    public void setStudName(String studName){
```

```
        this.studName = studName;
```

```
}
```

```
    public char getResult(){
```

```
        return result;
```

```
}
```

```
    public void setResult(char result){
```

```
        this.result = result;
```

```
}
```

```
    public int[] getMarks(){
```

```
        return marks;
```

```
}
```

```
    public void setMarks(int[] marks){
```

```
        this.marks = marks;
```

```
}
```

```

public GradeCalculator(String studName, int[] marks){
    this.studName = studName;
    this.marks = marks;
}

public void run(){
    int sum = 0;
    int[] score = getMarks();
    for(int i = 0;i<score.length;i++)
        sum = sum+score[i];
    if((400<=sum)&&(sum<=500))
        System.out.println(getStudName()+":"+ 'A');
    if((300<=sum)&&(sum<=399))
        System.out.println(getStudName()+":"+ 'B');
    if((200<=sum)&&(sum<=299))
        System.out.println(getStudName()+":"+ 'C');
    if(sum<200)
        System.out.println(getStudName()+":"+ 'E');
}
}

```

### Main.java\*

```

import java.util.Scanner;
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class Main {

    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter the number of Threads:");
        int th = Integer.parseInt(br.readLine());
        GradeCalculator obj = null;
        String str = "";

```

```
String details[] = new String[th];
for(int i=0; i<th; i++){
    System.out.println("Enter the String:");
    str = br.readLine();
    details[i]=str;
}

for(int i=0; i<th; i++){
    String sp[] = details[i].split(":");
    int k = 0;
    int arr[] = new int[sp.length];
    for(int j = 1; j<sp.length; j++)
        arr[k++] = Integer.parseInt(sp[j]);
    obj = new GradeCalculator(sp[0],arr);
    obj.start();
    try{
        Thread.sleep(1000);
    }
    catch(Exception e)
    {
        System.out.println(e);
    }
}

}

}
```

## Passanger Amenity

### Main.java\*

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        int num,n,i,count1=0,count2=0,y;
        char alpha,ch;
        String n1,n2;

        Scanner sc = new Scanner(System.in);

        //fill the code

        System.out.println("Enter the number of passengers");

        n=sc.nextInt();

        if(n<=0){

            System.out.println(n+" is invalid input");
            System.exit(0);

        }

        String[] arr1=new String[n];
        String[] arr2=new String[n];

        for(i=0;i<n;i++,count1=0,count2=0){

            System.out.println("Enter the name of the passenger "+(i+1));
            arr1[i] =sc.next();

            System.out.println("Enter the seat details of the passenger "+(i+1));
            arr2[i]= sc.next();

            num =Integer.parseInt(arr2[i].substring(1,(arr2[i].length())));

            alpha= arr2[i].charAt(0);

            if(num>=10 && num<=99){

                count2++;

            }

            for(ch=65;ch<84;ch++){

                if(ch==alpha){

                    count1++;

                }

            }

            if(count1==0){


```

```

        System.out.println(""+alpha+" is invalid coach");
        System.exit(0);
    }

    if(count2==0){

        System.out.println(num+" is invalid seat number");
        System.exit(0);
    }

}

for(i=0;i<n;i++){

    for(int j=i+1;j<n;j++){

        if(arr2[i].charAt(0)==arr2[j].charAt(0)){

if((Integer.parseInt(arr2[i].substring(1,(arr2[i].length()))))<(Integer.parseInt(arr2[j].substring(1,arr2[j].length())))){



n1=arr1[i];
n2=arr2[i];
arr1[i]=arr1[j];
arr2[i]=arr2[j];
arr1[j]=n1;
arr2[j]=n2;

}
}
else
if(arr2[i].charAt(0)<arr2[j].charAt(0))

{
n1=arr1[i];
n2=arr2[i];
arr1[i]=arr1[j];
arr2[i]=arr2[j];
arr1[j]=n1;
arr2[j]=n2;

}
}
}
}

```

```
for(i=0;i<n;i++){  
    String a=arr1[i].toUpperCase();  
    String b=arr2[i];  
    System.out.print(a+" "+b);  
    System.out.println("");  
}  
}  
}
```

## Perform Calculation

### **Calculate.java\***

```
public interface Calculate {  
  
    float performCalculation(int a,int b);  
}
```

### **Calculator.java\***

```
import java.util.Scanner;
```

```
public class Calculator {
```

```
    public static void main (String[] args) {
```

```
        Scanner sc=new Scanner(System.in);
```

```
        int a = sc.nextInt();
```

```
        int b= sc.nextInt();
```

```
        Calculate Perform_addition = performAddition();
```

```
        Calculate Perform_subtraction = performSubtraction();
```

```
        Calculate Perform_product = performProduct();
```

```
        Calculate Perform_division = performDivision();
```

```
System.out.println("The sum is "+Perform_addition.performCalculation(a,b));  
  
System.out.println("The difference is "+Perform_subtraction.performCalculation(a,b));  
  
System.out.println("The product is "+Perform_product.performCalculation(a,b));  
  
System.out.println("The division value is "+Perform_division.performCalculation(a,b));  
  
}
```

```
public static Calculate performAddition(){  
  
    Calculate Perform_calculation = (int a,int b)->a+b;  
  
    return Perform_calculation;  
  
}
```

```
public static Calculate performSubtraction(){  
  
    Calculate Perform_calculation = (int a,int b)->a-b;  
  
    return Perform_calculation;  
  
}
```

```
public static Calculate performProduct(){  
  
    Calculate Perform_calculation = (int a,int b)->a*b;  
  
    return Perform_calculation;
```

```
}
```

```
public static Calculate performDivision(){
```

```
Calculate Perform_calculation = (int a,int b)->{
```

```
    float c = (float)a;
```

```
    float d = (float)b;
```

```
    return (c/d);
```

```
};
```

```
return Perform_calculation;
```

```
}
```

```
}
```

## Query DataSet

### Query.java\*

```
//Write the required business logic as expected in the question description
```

```
public class Query {  
  
    private String queryId;  
    private String queryCategory;  
    private DataSet primaryDataSet;  
    private DataSet secondaryDataSet;  
  
    @Override  
    public String toString()  
    {  
        String g="";  
        g+="Primary data set"+'\n';  
        g+="Theatre id : "+primaryDataSet.getTheatreId()+'\n';  
        g+="Theatre name : "+primaryDataSet.getTheatreName()+'\n';  
        g+="Location :" +primaryDataSet.getLocation()+'\n';  
        g+="No of Screen : "+primaryDataSet.getNoOfScreen()+'\n';  
        g+="Ticket Cost : "+primaryDataSet.getTicketCost()+'\n';  
  
        g+="Secondary data set"+'\n';  
        g+="Theatre id : "+secondaryDataSet.getTheatreId()+'\n';  
        g+="Theatre name : "+secondaryDataSet.getTheatreName()+'\n';  
        g+="Location :" +secondaryDataSet.getLocation()+'\n';  
        g+="No of Screen : "+secondaryDataSet.getNoOfScreen()+'\n';  
        g+="Ticket Cost : "+secondaryDataSet.getTicketCost()+'\n';  
  
        g+="Query id : "+queryId+'\n';  
        g+="Query category : "+queryCategory+'\n';  
  
        return g;  
    }
```

```
public class DataSet{  
    private String theatreId;  
    private String theatreName;  
    private String location;  
    private int noOfScreen;  
    private double ticketCost;  
  
    public double getTicketCost()  
    {  
        return ticketCost;  
    }  
  
    public void setTicketCost(double a)  
    {  
        ticketCost=a;  
    }  
  
    public int getNoOfScreen()  
    {  
        return noOfScreen;  
    }  
  
    public void setNoOfScreen(int a)  
    {  
        noOfScreen=a;  
    }  
    public String getLocation ()  
    {  
        return location;  
    }  
    public void setLocation(String a)  
    {  
        location=a;  
    }  
}
```

```
}

public String getTheatreName ()
{
    return theatreName;
}

public void setTheatreName(String a)
{
    theatreName=a;
}

public String getTheatreId ()
{
    return theatreId;
}

public void setTheatreId(String a)
{
    theatreId=a;
}

public void setSecondaryDataSet(DataSet pD)
{
    this.secondaryDataSet=pD;
}

public DataSet getSecondaryDataSet()
{
    return this.secondaryDataSet;
}

public void setPrimaryDataSet(DataSet pD)
{
    this.primaryDataSet=pD;
}
```

```

public DataSet getPrimaryDataSet()
{
    return this.primaryDataSet;
}

public void setQueryId (String queryId)
{
    this.queryId=queryId;
}

public void setQueryCategory(String queryCategory)
{
    this.queryCategory=queryCategory;
}

public String getQueryId()
{
    return this.queryId;
}

public String getQueryCategory()
{
    return this.queryCategory;
}
}

```

### **TestApplication.java\***

```

import java.util.*;
public class TestApplication {
    //Write the required business logic as expected in the question description
    public static void main (String[] args) {
        Scanner sc= new Scanner (System.in);
        Query q= new Query();
        Query.DataSet pd= q.new DataSet();
        Query.DataSet sd= q.new DataSet();
        System.out.println("Enter the Details for primary data set");
        System.out.println("Enter the theatre id");
    }
}

```

```
pd.setTheatreId(sc.nextLine());
System.out.println("Enter the theatre name");
pd.setTheatreName(sc.nextLine());
System.out.println("Enter the location");
pd.setLocation(sc.nextLine());
System.out.println("Enter the no of screens");
pd.setNoOfScreen(sc.nextInt());
System.out.println("Enter the ticket cost");
pd.setTicketCost(sc.nextDouble());

System.out.println("Enter the Details for secondary data set");
System.out.println("Enter the theatre id");
String id2=sc.next();
// System.out.println(id2);
sd.setTheatreId(id2);
System.out.println("Enter the theatre name");
sc.nextLine();
sd.setTheatreName(sc.nextLine());
System.out.println("Enter the location");
String gll=sc.nextLine();
sd.setLocation(gll);
System.out.println("Enter the no of screens");

// System.out.println(gll);
// String pp=sc.nextLine();
// System.out.println(pp);
sd.setNoOfScreen(sc.nextInt());
System.out.println("Enter the ticket cost");
sd.setTicketCost(sc.nextDouble());
sc.nextLine();
System.out.println("Enter the query id");
q.setQueryId(sc.nextLine());
System.out.println("Enter the query category");
q.setQueryCategory(sc.nextLine());
```

```
q.setSecondaryDataSet(sd);
q.setPrimaryDataSet(pd);
System.out.println(q.toString());
}

}
```

## Retrieve Flight Based On Source And Destination

### Flight.java\*

```
public class Flight {  
  
    private int flightId;  
  
    private String source;  
  
    private String destination;  
  
    private int noOfSeats;  
  
    private double flightFare;  
  
    public int getFlightId() {  
        return flightId;  
    }  
  
    public void setFlightId(int flightId) {  
        this.flightId = flightId;  
    }  
  
    public String getSource() {  
        return source;  
    }  
  
    public void setSource(String source) {  
        this.source = source;  
    }  
  
    public String getDestination() {  
        return destination;  
    }  
  
    public void setDestination(String destination) {  
        this.destination = destination;  
    }  
  
    public int getNoOfSeats() {  
        return noOfSeats;  
    }  
  
    public void setNoOfSeats(int noOfSeats) {  
        this.noOfSeats = noOfSeats;  
    }  
}
```

```
public double getFlightFare() {
    return flightFare;
}

public void setFlightFare(double flightFare) {
    this.flightFare = flightFare;
}

public Flight(int flightId, String source, String destination,
        int noOfSeats, double flightFare) {
    super();
    this.flightId = flightId;
    this.source = source;
    this.destination = destination;
    this.noOfSeats = noOfSeats;
    this.flightFare = flightFare;
}

}
```

}

### **FlightManagement.java\***

```
import java.util.ArrayList;
import java.sql.*;

public class FlightManagementSystem {

    public ArrayList<Flight> viewFlightBySourceDestination(String source, String destination){
        ArrayList<Flight> flightList = new ArrayList<Flight>();
        try{
            Connection con = DB.getConnection();
            String query="SELECT * FROM flight WHERE source= '" + source + "' AND destination= '" + destination + "' ";

```

```

Statement st=con.createStatement();

ResultSet rst= st.executeQuery(query);

while(rst.next()){

    int flightId= rst.getInt(1);

    String src=rst.getString(2);

    String dst=rst.getString(3);

    int noofseats=rst.getInt(4);

    double flightfare=rst.getDouble(5);

    flightList.add(new Flight(flightId, src, dst, noofseats, flightfare));

}

}catch(ClassNotFoundException | SQLException e){

    e.printStackTrace();

}

return flightList;

}

}

```

### **DB.java\***

```

import java.io.FileInputStream;

import java.io.IOException;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.util.Properties;

public class DB {

    private static Connection con = null;

    private static Properties props = new Properties();

```

```

//ENSURE YOU DON'T CHANGE THE BELOW CODE WHEN YOU SUBMIT

public static Connection getConnection() throws ClassNotFoundException, SQLException {
    try{
        FileInputStream fis = null;
        fis = new FileInputStream("database.properties");
        props.load(fis);

        // load the Driver Class
        Class.forName(props.getProperty("DB_DRIVER_CLASS"));

        // create the connection now
        con =
        DriverManager.getConnection(props.getProperty("DB_URL"),props.getProperty("DB_USERNAME"),props.getProperty("DB_PASSWORD"));
    }
    catch(IOException e){
        e.printStackTrace();
    }
    return con;
}
}

```

### **Main.java\***

```

import java.util.Scanner;
import java.util.ArrayList;

public class Main{
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the source");
        String source=sc.next();
        System.out.println("Enter the destination");
        String destination=sc.next();
    }
}

```

```
FlightManagementSystem fms= new FlightManagementSystem();
ArrayList<Flight> flightList=fms.viewFlightBySourceDestination(source,destination);
if(flightList.isEmpty()){
    System.out.println("No flights available for the given source and destination");
    return;
}
System.out.println("Flightid Noofseats Flightfare");
for(Flight flight : flightList){
    System.out.println(flight.getFlightId()+" "+flight.getNoOfSeats()+" "+flight.getFlightFare());
}
}
```

## Silver Health Plan Insurance

### FamilyInsurancePolicy.java

```
public class FamilyInsurancePolicy extends InsurancePolicies{  
  
    public FamilyInsurancePolicy(String clientName, String policyId, int age, long mobileNumber, String emailId) {  
        super(clientName, policyId, age, mobileNumber, emailId);  
  
    }  
  
    public boolean validatePolicyId()  
    {  
        int count=0;  
        if(policyId.contains("FAMILY"));  
        count++;  
        char ch[]=policyId.toCharArray();  
        for(int i=6;i<9;i++)  
        {  
            if(ch[i]>='0' && ch[i]<='9')  
                count++;  
        }  
        if(count==4)  
            return true;  
        else  
            return false;  
    }  
    public double calculateInsuranceAmount(int months, int no_of_members)  
    {  
        double amount=0;  
        if(age>=5 && age<=25)  
            amount=2500*months*no_of_members;  
        else if (age>25 && age<60)  
            amount=5000*months*no_of_members;  
        else if (age>=60)  
            amount=10000*months*no_of_members;  
    }  
}
```

```
    return amount;  
}  
  
}
```

### **IndividualInsurancePolicy.java\***

```
public class IndividualInsurancePolicy extends InsurancePolicies{  
  
    public IndividualInsurancePolicy(String clientName, String policyId, int age, long mobileNumber, String  
emailId) {  
        super(clientName, policyId, age, mobileNumber, emailId);  
  
    }  
  
    public boolean validatePolicyId()  
    {  
        int count=0;  
        if(policyId.contains("SINGLE"));  
        count++;  
        char ch[]=policyId.toCharArray();  
        for(int i=6;i<9;i++)  
        {  
            if(ch[i]>='0' && ch[i]<='9')  
                count++;  
        }  
        if(count==4)  
            return true;  
        else  
            return false;  
    }  
  
    public double calculateInsuranceAmount(int months)  
    {  
        double amount=0;
```

```
if(age>=5 && age<=25)
    amount=2500*months;
else if (age>25 && age<60)
    amount=5000*months;
else if (age>=60)
    amount=10000*months;
return amount;
}

}
```

### **InsurancePolicies.java\***

```
public class InsurancePolicies {
    protected String clientName;
    protected String policyId;
    protected int age;
    protected long mobileNumber;
    protected String emailId;
    public String getClientName() {
        return clientName;
    }
    public void setClientName(String clientName) {
        this.clientName = clientName;
    }
    public String getPolicyId() {
        return policyId;
    }
    public void setPolicyId(String policyId) {
        this.policyId = policyId;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
```

```

        this.age = age;
    }

    public long getMobileNumber() {
        return mobileNumber;
    }

    public void setMobileNumber(long mobileNumber) {
        this.mobileNumber = mobileNumber;
    }

    public String getEmailId() {
        return emailId;
    }

    public void setEmailId(String emailId) {
        this.emailId = emailId;
    }

    public InsurancePolicies(String clientName, String policyId, int age, long mobileNumber, String emailId) {
        super();
        this.clientName = clientName;
        this.policyId = policyId;
        this.age = age;
        this.mobileNumber = mobileNumber;
        this.emailId = emailId;
    }
}

```

### **SeniorCitizenPolicy.java\***

```

public class SeniorCitizenPolicy extends InsurancePolicies{

    public SeniorCitizenPolicy(String clientName, String policyId, int age, long mobileNumber, String emailId) {
        super(clientName, policyId, age, mobileNumber, emailId);
    }

    public boolean validatePolicyId()
    {
        int count=0;
        if(policyId.contains("SENIOR"));

```

```

        count++;

        char ch[]=policyId.toCharArray();

        for(int i=6;i<9;i++)
        {

            if(ch[i]>='0' && ch[i]<='9')

                count++;

        }

        if(count==4)

            return true;

        else

            return false;

    }

}

public double calculateInsuranceAmount(int months, int no_of_members)

{

    double amount=0;

    if(age>=5 && age<60)

        amount=0;

    else if (age>=60)

        amount=10000*months*no_of_members;

    return amount;

}

}

```

### **UserInterface.java\***

```

import java.util.Scanner;

public class UserInterface {

    public static void main(String[] args)
    {

```

```

Scanner sc=new Scanner(System.in);

System.out.println("Enter Client name");

String name=sc.next();

System.out.println("Enter Policy Id");

String id=sc.next();

System.out.println("Enter Client age");

int age=sc.nextInt();

System.out.println("Enter mobile number");

long mnum=sc.nextLong();

System.out.println("Enter Email Id");

String email=sc.next();

InsurancePolicies policy=new InsurancePolicies(name,id,age,mnum,email);

System.out.println("Enter the months");

int month=sc.nextInt();

double amount=0;

if(id.contains("SINGLE"))

{

    IndividualInsurancePolicy g=new IndividualInsurancePolicy(name,id,age,mnum,email);

    if(g.validatePolicyId())

    {

        //System.out.println(g.validatePolicyId());

        amount=g.calculateInsuranceAmount(month);

        System.out.println("Name :" +name);

        System.out.println("Email Id :" +email);

        System.out.println("Amount to be paid :" +amount);

    }

    else

    {

        System.out.println("Provide valid Policy Id");

    }

}

```

```

else if(id.contains("FAMILY"))

{
    FamilyInsurancePolicy g=new FamilyInsurancePolicy(name,id,age,mnum,email);

    if(g.validatePolicyId())

    {

        System.out.println("Enter number of members");

        int num=sc.nextInt();

        amount=g.calculateInsuranceAmount(month,num);

        System.out.println("Name :" +name);

        System.out.println("Email Id :" +email);

        System.out.println("Amount to be paid :" +amount);

    }

    else

    {

        System.out.println("Provide valid Policy Id");

    }

}

else if(id.contains("SENIOR"))

{

    SeniorCitizenPolicy g=new SeniorCitizenPolicy(name,id,age,mnum,email);

    if(g.validatePolicyId())

    {

        System.out.println("Enter number of members");

        int num=sc.nextInt();

        amount=g.calculateInsuranceAmount(month,num);

        System.out.println("Name :" +name);

        System.out.println("Email Id :" +email);

        System.out.println("Amount to be paid :" +amount);

    }

    else

    {

        System.out.println("Provide valid Policy Id");

    }

}

```

```
else
    System.out.println("Provide valid Policy Id");
}

}
```

## Travel Request System

### TraveRequestDao.java\*

```
package com.cts.travelrequest.dao;

import java.io.FileInputStream;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;

import java.sql.*;
import java.util.*;
//import java.util.Properties;

import com.cts.travelrequest.vo.TravelRequest;

class DB{

    private static Connection con = null;
    private static Properties props = new Properties();

    //ENSURE YOU DON'T CHANGE THE BELOW CODE WHEN YOU SUBMIT

    public static Connection getConnection() throws ClassNotFoundException, SQLException {
        try{

            FileInputStream fis = null;
            fis = new FileInputStream("resource/database.properties");
            props.load(fis);

            // load the Driver Class
            Class.forName(props.getProperty("DB_DRIVER_CLASS"));

            // create the connection now
            con = DriverManager.getConnection(props.getProperty("URL"),
                props.getProperty("username"),
                props.getProperty("password"));
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

    public static void closeConnection(){
        try{
            if(con != null)
                con.close();
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }
}
```

```

        con =
DriverManager.getConnection(props.getProperty("DB_URL"),props.getProperty("DB_USERNAME"),props.getProperty("DB_PASSWORD"));

    }

    catch(IOException e){
        e.printStackTrace();
    }

    return con;
}

}

```

```

/**
 * Method to get travel details based on source and destination city      *
 * @return list
 */

public class TravelRequestDao{

    // public PreparedStatement prepareStatement(String query) throws SQLException{}

    public static List<TravelRequest> getTravelDetails(String sourceCity, String destinationCity) {

        List<TravelRequest> travel=new ArrayList<>();

        try{
            Connection con=DB.getConnection();

            String query="Select * from t_travelrequest where sourceCity=? and destinationCity=?;";

            PreparedStatement ps=con.prepareStatement(query);

            ps.setString(1,sourceCity);
            ps.setString(2,destinationCity);

            ResultSet rs=ps.executeQuery();

            while(rs.next()){

                String tid=rs.getString("travelReqId");
                java.sql.Date date=rs.getDate("travelDate");
                String apstat=rs.getString("approvalStatus");
                String sour=rs.getString("sourceCity");
                String des=rs.getString("destinationCity");
                double cost=rs.getDouble("travelCost");

                TravelRequest tr=new TravelRequest(tid,date,apstat,sour,des,cost);

```

```

        travel.add(tr);
    }
}

catch(ClassNotFoundException e){
    e.printStackTrace();
}

catch(SQLException e ){
    e.printStackTrace();
}

return travel; //TODO change this return value
}

/** 
 * Method to calculate total travel cost based approvalStatus
 * @return list
 */
public static double calculateTotalTravelCost(String approvalStatus) {
    double amount=0;
    try{
        Connection con=DB.getConnection();
        String query="select travelCost from t_travelrequest where approvalStatus=?;";
        PreparedStatement ps1=con.prepareStatement(query);
        ps1.setString(1,approvalStatus);
        ResultSet rs1=ps1.executeQuery();
        while(rs1.next()){
            amount+=rs1.getDouble("travelCost");
        }
    }
    catch(ClassNotFoundException e){
        e.printStackTrace();
    }
}

```

```

        catch(SQLException e){
            e.printStackTrace();
        }

        return amount; //TODO change this return value
    }
}

```

### **TravelRequestService.java\***

```

package com.cts.travelrequest.service;

import java.util.List;

import com.cts.travelrequest.dao.TravelRequestDao;
import com.cts.travelrequest.vo.TravelRequest;

public class TravelRequestService {

    /**
     * Method to validate approval status
     *
     * @return status
     */
    public String validateApprovalStatus(String approvalStatus) {
        if(approvalStatus.equalsIgnoreCase("Approved") | |approvalStatus.equalsIgnoreCase("Pending")){
            return "valid";
        }
        return "invalid"; //TODO change this return value
    }

    /**
     * Method to validate source and destination city
     *
     * @return status
     */
}

```

```

*/
public String validateSourceAndDestination(String sourceCity, String destinationCity) {
    if(!sourceCity.equalsIgnoreCase(destinationCity)){
        if(sourceCity.equalsIgnoreCase("Pune")|| sourceCity.equalsIgnoreCase("Mumbai")|||
        sourceCity.equalsIgnoreCase("Chennai")|| sourceCity.equalsIgnoreCase("Bangalore")|||
        sourceCity.equalsIgnoreCase("Hydrabad")){
            if(destinationCity.equalsIgnoreCase("Pune")|||
            destinationCity.equalsIgnoreCase("Mumbai")|||destinationCity.equalsIgnoreCase("Chennai")|||
            destinationCity.equalsIgnoreCase("Bangalore")||| destinationCity.equalsIgnoreCase("Hydrabad")){
                return "valid";
            }
        } else{
            return "invalid";
        }
    } else{
        return "invalid";
    }
}

/**
 * Method to invoke getTravelDetails method of TravelRequestDao class
 *
 * @return listOfTravelRequest
 */
public List<TravelRequest> getTravelDetails(String sourceCity, String destinationCity) {
    if(this.validateSourceAndDestination(sourceCity,destinationCity).contentEquals("valid")){
        return TravelRequestDao.getTravelDetails(sourceCity,destinationCity);
    }
    else{
        return null;
    }
}

```

```

}

/**
 * Method to invoke calculateTotalTravelCost method of TravelRequestDao class
 *
 * @return totalCost
 */
public double calculateTotalTravelCost(String approvalStatus) {
    if(this.validateApprovalStatus(approvalStatus).equals("valid")){
        return TravelRequestDao.calculateTotalTravelCost(approvalStatus);
    }
    else{
        return -1;
    }
}

```

### **SkeletonValidator.java\***

```

package com.cts.travelrequest.skeletonvalidator;

import java.lang.reflect.Method;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * @author t-aarti3
 *      This class is used to verify if the Code Skeleton is intact and not
 *      modified by participants thereby ensuring smooth auto evaluation
 */

public class SkeletonValidator {
    public SkeletonValidator() {
        validateClassName("com.cts.travelrequest.service.TravelRequestService");
        validateClassName("com.cts.travelrequest.vo.TravelRequest");
    }
}

```

```

validateMethodSignature()

"validateApprovalStatus:java.lang.String,validateSourceAndDestination:java.lang.String,getTravelDetails:java
.util.List,calculateTotalTravelCost:double",
"com.cts.travelrequest.service.TravelRequestService");

}

private static final Logger LOG = Logger.getLogger("SkeletonValidator");

protected final boolean validateClassName(String className) {

    boolean iscorrect = false;

    try {

        Class.forName(className);

        iscorrect = true;

        LOG.info("Class Name " + className + " is correct");

    } catch (ClassNotFoundException e) {

        LOG.log(Level.SEVERE, "You have changed either the " + "class name/package. Use the
correct package "
                + "and class name as provided in the skeleton");

    } catch (Exception e) {

        LOG.log(Level.SEVERE,
                "There is an error in validating the " + "Class Name. Please manually verify
that the "
                + "Class name is same as skeleton before uploading");

    }

    return iscorrect;
}

protected final void validateMethodSignature(String methodWithExcptn, String className) {

    Class cls = null;

    try {

        String[] actualmethods = methodWithExcptn.split(",");

```

```

boolean errorFlag = false;
String[] methodSignature;
String methodName = null;
String returnType = null;

for (String singleMethod : actualmethods) {
    boolean foundMethod = false;
    methodSignature = singleMethod.split(":");
    methodName = methodSignature[0];
    returnType = methodSignature[1];
    cls = Class.forName(className);
    Method[] methods = cls.getMethods();
    for (Method findMethod : methods) {
        if (methodName.equals(findMethod.getName())) {
            foundMethod = true;
            if (!(findMethod.getReturnType().getName().equals(returnType))) {
                errorFlag = true;
                LOG.log(Level.SEVERE, " You have changed the " + "return
type in " + methodName
                                         + " method. Please stick to the " + "skeleton
provided");
            }
        } else {
            LOG.info("Method signature of " + methodName + " is
valid");
        }
    }
    if (!foundMethod) {
        errorFlag = true;
        LOG.log(Level.SEVERE, " Unable to find the given public method " +
methodName
                                         + ". Do not change the " + "given public method name. " +
"Verify it with the skeleton");
    }
}

```

```

        }

    }

    if (!errorFlag) {
        LOG.info("Method signature is valid");
    }

}

} catch (Exception e) {
    LOG.log(Level.SEVERE,
        " There is an error in validating the " + "method structure. Please manually
verify that the "
        + "Method signature is same as the skeleton before
uploading");
}

}

}

```

### **TravelRequest.java\***

```

package com.cts.travelrequest.vo;

import java.util.Date;

public class TravelRequest {
    // member variables
    private String travelReqId;
    private Date travelDate;
    private String approvalStatus;
    private String sourceCity;
    private String destinationCity;
    private double travelCost;

    public TravelRequest() {
        super();
        // TODO Auto-generated constructor stub
    }
}

```

```

}

// parameterized constructor

public TravelRequest(String travelReqId, Date travelDate, String approvalStatus, String sourceCity,
                     String destinationCity, double travelCost) {

    super();
    this.travelReqId = travelReqId;
    this.travelDate = travelDate;
    this.approvalStatus = approvalStatus;
    this.sourceCity = sourceCity;
    this.destinationCity = destinationCity;
    this.travelCost = travelCost;
}

// setter, getter

/** 
 * @return the travelReqId
 */
public String getTravelReqId() {
    return travelReqId;
}

/** 
 * @param travelReqId
 *      the travelReqId to set
 */
public void setTravelReqId(String travelReqId) {
    this.travelReqId = travelReqId;
}

/** 
 * @return the travelDate
 */
public Date getTravelDate() {
    return travelDate;
}

/** 

```

```
* @param travelDate
*      the travelDate to set
*/
public void setTravelDate(Date travelDate) {
    this.travelDate = travelDate;
}

/**
* @return the approvalStatus
*/
public String getApprovalStatus() {
    return approvalStatus;
}

/**
* @param approvalStatus
*      the approvalStatus to set
*/
public void setApprovalStatus(String approvalStatus) {
    this.approvalStatus = approvalStatus;
}

/**
* @return the sourceCity
*/
public String getSourceCity() {
    return sourceCity;
}

/**
* @param sourceCity
*      the sourceCity to set
*/
public void setSourceCity(String sourceCity) {
    this.sourceCity = sourceCity;
}

/**
* @return the sourceCity
```

```

        */
    public String getDestinationCity() {
        return destinationCity;
    }

    /**
     * @param destinationCity
     *      the destinationCity to set
     */

    public void setDestinationCity(String destinationCity) {
        this.destinationCity = destinationCity;
    }

    /**
     * @return the travelCost
     */

    public double getTravelCost() {
        return travelCost;
    }

    /**
     * @param travelCost
     *      the travelCost to set
     */

    public void setTravelCost(double travelCost) {
        this.travelCost = travelCost;
    }

}

```

### Main.java\*

```

package com.cts.travelrequest.main;

import java.sql.*;
import java.util.*;
import java.text.SimpleDateFormat;
import com.cts.travelrequest.service.TravelRequestService;

```

```

import com.cts.travelrequest.skeletonvalidator.SkeletonValidator;
import com.cts.travelrequest.vo.TravelRequest;
public class Main {

    public static void main(String[] args) throws SQLException {
        // CODE SKELETON - VALIDATION STARTS
        // DO NOT CHANGE THIS CODE
        new SkeletonValidator();
        // CODE SKELETON - VALIDATION ENDS
        //TravelRequest tr=new TravelRequest();
        //List<TravelRequest> ltr=new ArrayList<>();
        TravelRequestService service = new TravelRequestService();
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter source city:");
        String sourceCity=sc.next();
        System.out.println("Enter destination city:");
        String destinationCity=sc.next();
        System.out.println("Enter approval status to find total travel cost:");
        String status=sc.next();

        if(service.validateSourceAndDestination(sourceCity,destinationCity).equals("valid")){
            List<TravelRequest> ltr=service.getTravelDetails(sourceCity, destinationCity);
            if(ltr.isEmpty()){
                System.out.println("No travel request raised for given source and destination cities");
            }
            else{
                for(TravelRequest t:ltr){
                    SimpleDateFormat sd= new SimpleDateFormat("dd-MMM-YYYY");
                    String d=sd.format(t.getTravelDate());
                    System.out.println(t.getTravelReqId()+"\t| "+d+"\t| "+t.getApprovalStatus()+"\t| "+t.getSourceCity()+"\t| "+t.getDestinationCity()+"\t| "+t.getTravelCost());
                }
            }
        }
    }
}

```

```
else{
    System.out.println("Provide correct source and destination city");
}

if(service.validateApprovalStatus(status).contentEquals("valid")){
    System.out.println(service.calculateTotalTravelCost(status));
}

else{
    System.out.println("Provide valid approval status");
}

}

}
```

## Print Customer Details

```
import java.util.*;
2 import java.io.*;
3 public class Customer{
4     public static void main(String args[]){
5         Scanner sc=new Scanner(System.in);
6         System.out.println("Enter your name:");
7         String name=sc.nextLine();
8         System.out.println("Enter age:");
9         int age=sc.nextInt();
10        System.out.println("Enter gender:");
11        String gender=sc.next();
12        sc.nextLine();
13        System.out.println("Hailing from:");
14        String from=sc.nextLine();
15        System.out.println("Welcome," +name+"!");
16        System.out.println("Age:"+age);
17        System.out.println("Gender:"+gender);
18        System.out.println("City:"+from);
19    }
20}
```

## Finding Range Of The Number

```
import java.util.Scanner;
2 public class Main
3 {
4     public static void main(String[] args)
5     {
6         Scanner sc=new Scanner(System.in);
7         System.out.println("Enter the number");
8         int value=sc.nextInt();
9         if(value>0 && value<=100)
10        {
11            System.out.println("Range of the number "+value+" is 0 to 100");
12        }
13        else if(value>100 && value<=200)
14        {
15            System.out.println("Range of the number "+value+" is 100 to 200");
16        }
17        else if(value>200 && value<=500)
18        {
19            System.out.println("Range of the number "+value+" is 200 to 500");
20        }
21        else{
22            System.out.println("Entered number "+value+" is not in the expected range");
23        }
24    }
25
26}
```

## Road Signaling

```
import java.util.*;
2
3 public class Main{
4
5     public static void main(String[] args){
6
7         Scanner sc=new Scanner(System.in);
```

```

8 System.out.println("Enter the color");
9 String color=sc.nextLine();
10 if(color.equals("green")){
11     System.out.println("go");
12 }
13 else if(color.equals("red")){
14     System.out.println("stop");
15 }
16 else if(color.equals("yellow")){
17     System.out.println("proceed with caution");
18 }
19 else{
20     System.out.println("prepare to go");
21 }
22 }
23 }
```

## Electricity bill calculation

```

import java.util.Scanner;
2
3 public class Main {
4
5     public static void main(String[] args) {
6
7         Scanner sc = new Scanner(System.in);
8         System.out.println("Enter the units consumed");
9         int units = sc.nextInt();
10        if (units <= 20) {
11            System.out.println("No charge");
12        } else if (units > 20 && units < 100) {
13            System.out.println("You have to pay Rs." + (units * 3.50));
14        } else if (units >= 100) {
15            System.out.println("You have to pay Rs." + (units * 5.0));
16        }
17    }
18 }
19 }
20 }
```

## Checking budget of mobile

```

import java.util.Scanner;
2
3 public class Main{
4
5     public static void main(String[] args){
6
7         Scanner sc=new Scanner(System.in);
8         System.out.println("Enter the cost of the mobile");
9         int price=sc.nextInt();
10        if(price<=13000)
11        {
12            System.out.println("Mobile chosen is within the budget");
13        }
14    }
15    else{
16        System.out.println("Mobile chosen is beyond the budget");
17    }
18
19 }
```

21 }

## Finding even number

```
1 import java.util.Scanner;
2 public class Main {
3     public static void main(String args[]){
4
5         Scanner sc = new Scanner(System.in);
6         System.out.println("Enter the number");
7         int num = sc.nextInt();
8
9
10        if (num % 2 == 0){
11            System.out.println(num+" is an even number");
12        }
13        else {
14            System.out.println(num +
15                " is not an even number");
16        }
17    }
18 }
```

## Registration Details

```
import java.util.Scanner;
2
3 public class RegistrationDetails{
4
5     public static void main(String[] args){
6
7         Scanner sc=new Scanner(System.in);
8
9         System.out.println("Enter your name:");
10        String name=sc.nextLine();
11        System.out.println("Enter your age:");
12        int age=sc.nextInt();
13        System.out.println("Enter your phoneno:");
14        String pno=sc.nextLine();
15        System.out.println("Enter your qualification:");
16        String qualification=sc.nextLine();
17        System.out.println("Enter your email id[Please provide valid id, after registering your registration id will be mailed]");
18        String email=sc.nextLine();
19        System.out.println("Enter your noofexperience[if any]:");
20        String noe=sc.nextLine();
21        System.out.println("Dear "+name+", Thanks for registering in our portal, registration id will be mailed to "+noe+" within 2 working days");
22    }
23
24 }
25
```

## Celcius to Farenheit Conversion

```
import java.util.Scanner;
2
3
4 public class CelsiusConversion
5 {
```

```

6  public static void main (String[] args) {
7      Scanner sc=new Scanner(System.in);
8      double cel=sc.nextDouble();
9      double fah=((9*cel)/5)+32;
10     System.out.println(fah);
11 }
12 }
```

## Display Characters

```

import java.util.Scanner;
2
3 public class Ascivalue{
4
5     public static void main(String args[]){
6
7         Scanner in=new Scanner(System.in);
8
9         System.out.println("Enter the digits: ");
10
11        int a=in.nextInt();
12
13        int b=in.nextInt();
14
15        int c=in.nextInt();
16
17        int d=in.nextInt();
18
19
20
21        char q=(char)a;
22
23        char w=(char)b;
24
25        char e=(char)c;
26
27        char r=(char)d;
28
29        System.out.println();
30
31        System.out.print(a);
32
33        System.out.println("-"+q);
34
35        System.out.print(b);
36
37        System.out.println("-"+w);
38
39        System.out.print(c);
40
41        System.out.println("-"+e);
42
43        System.out.print(d);
44
45        System.out.println("-"+r);
46
47    }
48
49 }
```

## Bill Generation

```
1 import java.util.Scanner;
2 public class SnacksDetails
3 {
4     public static void main (String[] args) {
5         Scanner sc=new Scanner(System.in);
6         System.out.println("Enter the no of pizzas bought:");
7         int piz=sc.nextInt();
8         System.out.println("Enter the no of puffs bought:");
9         int puf=sc.nextInt();
10        System.out.println("Enter the no of cool drinks bought:");
11        int cdk=sc.nextInt();
12
13        System.out.println("Bill Details");
14        System.out.println("No of pizzas:"+piz);
15        System.out.println("No of puffs:"+puf);
16        System.out.println("No of cooldrinks:"+cdk);
17        System.out.println("Total price="+ (piz*100+puf*20+cdk*10));
18        System.out.println("ENJOY THE SHOW!!!");
19    }
20 }
```

## Find Season

```
import java.util.Scanner;
2 public class Season
3 {
4     public static void main (String[] args) {
5         Scanner sc = new Scanner(System.in);
6         System.out.println("Enter the month:");
7         int mon=sc.nextInt();
8         if(mon>12||mon<1)
9         {
10             System.out.println("Invalid month");
11         }
12         else if(mon>=3&&mon<=5)
13         {
14             System.out.println("Season:Spring");
15         }
16         else if(mon>=6&&mon<=8)
17         {
18             System.out.println("Season:Summer");
19         }
20         else if(mon>=9&&mon<=11)
21         {
22             System.out.println("Season:Autumn");
23         }
24         else if(mon==12||mon==1||mon==2)
25         {
26             System.out.println("Season:Winter");
27         }
28     }
29 }
```

## Check for Leap Year

```
1 import java.util.Scanner;
2 public class LeapYear
3 {
4     public static void main (String[] args) {
```

```

5 Scanner sc=new Scanner(System.in);
6 System.out.println("Enter the Year");
7
8 int yr=sc.nextInt();
9
10 if(yr>999 && yr<10000)
11 {
12     if(yr%4==0)
13     {
14         if(yr%100==0)
15         {
16             if(yr%400==0)
17             {
18                 System.out.println("Leap Year");
19             }
20         else
21         {
22             System.out.println("Not a Leap Year");
23         }
24     }
25 else
26 {
27     System.out.println("Leap Year");
28 }
29 }
30 else
31 {
32     System.out.println("Not a Leap Year");
33 }
34 }
35 else
36 {
37     System.out.println("Invalid Year");
38 }
39 }
40 }

```

## Highest Placement

```

import java.util.Scanner;
2 public class Placement
3 {
4     public static void main (String[] args) {
5         Scanner sc=new Scanner(System.in);
6         System.out.println("Enter the no of students placed in CSE:");
7         int CSE= sc.nextInt();
8         System.out.println("Enter the no of students placed in ECE:");
9         int ECE=sc.nextInt();
10        System.out.println("Enter the no of students placed in MECH:");
11        int MECH=sc.nextInt();
12
13        if(CSE<0||ECE<0||MECH<0)
14        {
15            System.out.println("Input is Invalid");
16        }
17
18        else if(CSE==ECE && ECE==MECH)
19        {
20            System.out.println("None of the department has got the highest placement");
21        }
22

```

```

23     else if(CSE==ECE && MECH<CSE)
24     {
25         System.out.println("Highest placement");
26         System.out.println("CSE");
27         System.out.println("ECE");
28     }
29
30     else if(CSE==MECH && ECE<MECH)
31     {
32         System.out.println("Highest placement");
33         System.out.println("CSE");
34         System.out.println("MECH");
35     }
36
37     else if(ECE==MECH && CSE<MECH)
38     {
39         System.out.println("Highest placement");
40         System.out.println("ECE");
41         System.out.println("MECH");
42     }
43
44     else if(CSE>ECE&&CSE>MECH)
45     {
46         System.out.println("Highest placement");
47         System.out.println("CSE");
48     }
49
50     else if(ECE>CSE && ECE>MECH)
51     {
52         System.out.println("Highest placement");
53         System.out.println("ECE");
54     }
55
56     else if(MECH>CSE&&MECH>ECE)
57     {
58         System.out.println("Highest placement");
59         System.out.println("MECH");
60     }
61 }
62 }
```

## ***IncrementCalculation***

```

import java.util.*;
2
3 class IncrementCalculation{
4
5     public static void main (String[] args) {
6
7
8
9     Scanner sc = new Scanner (System.in);
10
11    System.out.println("Enter the salary");
12
13    int salary = sc.nextInt();
14
15    System.out.println("Enter the Performance appraisal rating");
16
17    float rating = sc.nextFloat();
18 }
```

```

19     if(salary<1||rating<1.0||rating>5.0){
20         System.out.println("Invalid Input");
21         System.exit(0);
22     }
23
24     else if(rating>=1&&rating<=3){
25         salary=salary+(int)(0.1*salary);
26         System.out.println(salary);
27     }
28
29     else if(rating>3&&rating<=4){
30         salary=salary+(int)(0.25*salary);
31         System.out.println(salary);
32     }
33
34     else /*if(rating>4&&rating<=5)*{
35         salary=salary+(int)(0.3*salary);
36         System.out.println(salary);
37     }
38
39     */
40 }
41
42 }
43
44 }
45
46 }
47
48 }
49
50 }
51 }
52
53 }

```

## Display String

```

import java.util.Scanner;
3
4 public class Main {
5
6     public static void main(String[] args){
7         Scanner sc=new Scanner(System.in);
8         System.out.println("Enter the String");
9         String st=sc.nextLine();
10        System.out.println("Enter the number");
11        int nb=sc.nextInt();
12        if(nb<1){
13            System.out.println(nb+ "is not a valid input");
14        }
15        else{
16            for(int i=0;i<=nb;i++){
17                System.out.println(st);
18            }
19        }
20    }
21
22 }
import java.util.Scanner;
3

```

```

4 public class Main {
5
6     public static void main(String[] args){
7         Scanner sc=new Scanner(System.in);
8         System.out.println("Enter the String");
9         String st=sc.nextLine();
10        System.out.println("Enter the number");
11        int nb=sc.nextInt();
12        if(nb<1){
13            System.out.println(nb+ "is not a valid input");
14        }
15        else{
16            for(int i=0;i<=nb;i++){
17                System.out.println(st);
18            }
19        }
20    }
21
22}

```

## Reverse the number

```

import java.util.Scanner;
2 public class Main
3 {
4 //method for reverse a number
5 public static void reverseNumber(int number)
6 {
7 if (number < 10)
8 {
9 //prints the same number if the number is less than 10
10 System.out.println(number);
11 return;
12 }
13 else
14 {
15 System.out.print(number % 10);
16 reverseNumber(number/10);
17 }
18 }
19 public static void main(String args[])
20 {
21 System.out.print("Enter the number that you want to reverse: ");
22 Scanner sc = new Scanner(System.in);
23 int num = sc.nextInt();
24 System.out.print("The reverse of the given number is: ");
25 //method calling
26 reverseNumber(num);
27 }
28 }
29

```

## Prime Factors

```

1 import java.util.*;
2 class Main {
3     public static void main(String[] args)
4     {
5         Scanner sc=new Scanner(System.in);
6         Scanner scan = new Scanner(System.in);
7         System.out.print("Input a number: ");

```

```

8   int n = scan.nextInt();
9     if (n>0)
10    {
11      while (n%2==0)
12      {
13        System.out.print(2 + " ");
14        n /= 2;
15      }
16
17      for (int i = 3; i <= Math.sqrt(n); i+= 2)
18      {
19        while (n%i == 0)
20        {
21          System.out.print(i + " ");
22          n /= i;
23        }
24      }
25      if (n > 2)
26        System.out.print(n);
27    }
28  }
29}
30

```

## Count of Prime numbers

```

import java.util.Scanner;
2 public class Main
3 {
4   public static void main(String[] args)
5   {
6
7   Scanner sc=new Scanner(System.in);
8
9   System.out.print("Enter starting range");
10  int start = sc.nextInt();
11  System.out.print("\nEnter ending range");
12  int end = sc.nextInt();
13
14  int count;
15
16  for(int i = start ; i <= end ; i++)
17  {
18
19    count = 0;
20    for(int j = 1 ; j <= i ; j++)
21    {
22      if(i % j == 0)
23        count = count+1;
24    }
25
26    System.out.println(i);
27  }
28
29
30 }
31}
32

```

## Palindrome numbers

```
import java.util.*;
2 class Main
3 {
4
5     static int isPalindrome(int n)
6     {
7
8         int rev = 0;
9         for (int i = n; i > 0; i /= 10)
10            rev = rev * 10 + i % 10;
11
12
13
14     return(n == rev) ? 1 : 0;
15 }
16
17
18     static void countPal(int min, int max)
19     {
20         for (int i = min; i <= max; i++)
21             if (isPalindrome(i)==1)
22                 System.out.print(i + " ");
23     }
24
25
26     public static void main(String args[])
27     {
28         System.out.println("Enter the starting range");
29         Scanner sc=new Scanner(System.in);
30         int start=sc.nextInt();
31         System.out.println("Enter the ending range");
32         int end=sc.nextInt();
33         countPal(start, end);
34     }
35 }
36 }
```

## Display Month/Day

```
import java.util.Scanner;
2
3     public class Main {
4
5         public static void main(String[] args){
6             Scanner sc=new Scanner(System.in);
7             int n=0;
8             for(;n!=3;){
9                 System.out.println("1.Display the Months");
10                System.out.println("2.Display the Days");
11                System.out.println("3.Exit");
12                n=sc.nextInt();
13                switch(n)
14                {
15                    case 1:{}
16                        System.out.println("1.January");
17                        System.out.println("2.February");
18                        System.out.println("3.March");
19                        System.out.println("4.April");
20                        System.out.println("5.May");
```

```

21         System.out.println("6.June");
22         System.out.println("7.July");
23         System.out.println("8.August");
24         System.out.println("9.September");
25         System.out.println("10.October");
26         System.out.println("11.November");
27         System.out.println("12.December");
28         break;
29     }
30     case 2:{
31         System.out.println("1.Monday");
32         System.out.println("2.Tuesday");
33         System.out.println("3.Wednesday");
34         System.out.println("4.Thursday");
35         System.out.println("5.Friday");
36         System.out.println("6.Saturday");
37         System.out.println("7.Sunday");
38         break;
39     }
40     case 3: System.exit(0);
41
42 }
43
44 }
45 }
46 }
47

```

## Power of a Number

```

import java.util.*;
2  public class Power
3  {
4      public static void main(String args[]){
5          Scanner scan = new Scanner (System.in);
6          long n, p;
7          System.out.println("Enter the number");
8          n = scan.nextLong();
9          System.out.println("Enter the digit");
10         p = scan.nextLong();
11         if(n < 0 || p < 0) {
12             System.out.println("Invalid input.");
13             return;
14         }
15         if(n == 0 && p == 0) {
16             System.out.println("Invalid input");
17             return;
18         }
19         if (n > 0 && p == 0) {
20             System.out.println("1");
21             return;
22         }
23         if(n == 0 && p > 0) {
24             System.out.println("0");
25             return;
26         }
27         else
28             System.out.println(power(n, p));
29     }
30     private static long power(long n, long p) {
31         long result = n;

```

```

32     for(int i=1; i<p; i++)
33         result *= n;
34     return result;
35   }
36 }
37

```

## Factors of a Number

```

1 import java.util.*;
2 import java.lang.*;
3 public class FindFactor
4 {
5     public static void main (String[] args) {
6         Scanner sc = new Scanner(System.in);
7         int num=Math.abs(sc.nextInt());
8         if(num==0)
9         {
10             System.out.println("No Factors");
11         }
12         else
13         {System.out.print("1");
14             for(int i=2;i<=num;i++)
15             {
16                 if(num%i==0)
17                 {
18                     System.out.print(", "+i);
19                 }
20             }
21         }
22     }
23 }

```

## Lucky Number

```

import java.util.Scanner;
2 public class LuckyNum
3 {
4     public static void main (String[] args) {
5         Scanner sc= new Scanner(System.in);
6         System.out.println("Enter the car no:");
7         int num=sc.nextInt();
8         if(num<1000 || num>9999)
9         {
10             System.out.println(num+" is not a valid car number");
11         }
12         else
13         {int sum=0;
14             while(num>0)
15             {
16                 sum+=num%10;
17                 num/=10;
18             }
19             if(sum%3==0 || sum%5==0 || sum%7==0)
20             {
21                 System.out.println("Lucky Number");
22             }
23             else
24             {
25                 System.out.println("Sorry its not my lucky number");
26             }

```

```
27     }
28 }
29 }
```

## Number Palindrome

```
import java.util.Scanner;
1
2 public class Palindrome {
3
4     public static void main(String[] args) {
5
6         int lastDigit, sum = 0, a;
7         Scanner sc = new Scanner(System.in);
8         int num = sc.nextInt();
9         if (num < 0) {
10             System.out.println("Invalid Input");
11         } else {
12             a = num;
13
14
15             while (a > 0) {
16
17                 lastDigit = a % 10;
18                 sum = (sum * 10) + lastDigit;
19                 a = a / 10;
20
21             }
22
23
24
25             if (sum == num)
26                 System.out.println("Palindrome");
27             else
28                 System.out.println("Not a Palindrome");
29
30         }
31
32     }
33 }
34 }
```

## Find Number for Given Factorial

```
1 import java.util.Scanner;
2 public class FindNumber{
3     public static void main(String args[]){
4         Scanner in=new Scanner(System.in);
5         int count=0;
6         int n=in.nextInt();
7         if(n>0){
8             for(int i=1;i<=n;i++){
9                 if(n%i==0){
10                     n=n/i;
11                     count=count+1;
12                 }
13                 else{
14                     break;
15                 }
16                 if(n==1){
17                     System.out.print(count);
18                 }else{
19                     System.out.print("Sorry.The given number is not a perfect factorial");
20                 }
21             }
22         }
23     }
24 }
```

```

20      }
21  }else if(n<=0)
22  {
23      System.out.print("Invalid Input");
24  }
25 }
26 }
27

```

## List of prime numbers

```

import java.util.*;
2 public class PrimeNumbers
3 {
4     public static void main (String[] args) {
5         Scanner sc=new Scanner(System.in);
6         int a=sc.nextInt();
7         int b=sc.nextInt();
8         int flag;
9         if(a<=0 || b<=0 || a>=b)
10
11         System.out.println("Provide valid input");
12     else
13     {
14         Inner:
15         while(a<=b)
16         {
17             if(a==2)
18                 System.out.print(a+" ");
19             else if(a==1)
20             {
21                 a++;
22                 continue;
23             }
24             else
25             {
26                 flag=0;
27                 outer:
28                 for(int i=2;i<=a/2;i++)
29                 {
30                     if(a%i==0)
31                     {
32                         flag=1;
33                         break outer;
34                     }
35                 }
36                 if(flag==0)
37                     System.out.print(a+" ");
38             }
39             a++;
40         }
41     }
42 }
43 }

```

Student Details - Constructor  
1 **public class** Student

```

2{
3    private int studentId;
4    private String studentName, studentAddress, collegeName;
5
6    public Student(int studentId, String studentName, String studentAddress)
7    {
8        this.studentAddress=studentAddress;
9        this.studentName=studentName;
10       this.studentId=studentId;
11       this.collegeName="NIT";
12    }
13
14   public Student(int studentId, String studentName, String studentAddress, String collegeName)
15   {
16       this(studentId, studentName, studentAddress);//invoking a constructor
17       this.collegeName=collegeName;
18   }
19
20  public int getStudentId()
21  {
22      return this.studentId;
23  }
24
25  public String getStudentName()
26  {
27      return this.studentName;
28  }
29
30  public String getStudentAddress()
31  {
32      return this.studentAddress;
33  }
34
35  public String getCollegeName()
36  {
37      return this.collegeName;
38  }
39}

```

## Book Detail

### **TestBook.java**

```

import java.util.Scanner;
2 public class TestBook
3{
4    public static void main (String[] args) {
5        Scanner sc=new Scanner(System.in);
6
7        System.out.println("Enter the Book name:");
8        String bookname=sc.nextLine();
9
10       System.out.println("Enter the price:");
11       int price=sc.nextInt();
12       sc.nextLine();
13
14       System.out.println("Enter the Author name:");

```

```

15     String authorname=sc.nextLine();
16
17     Book obj=new Book();
18     obj.setBookName(bookname);
19     obj.setBookPrice(price);
20     obj.setAuthorName(authorname);
21     System.out.println("Book Details");
22     System.out.println("Book Name :" +obj.getBookName());
23     System.out.println("Book Price :" +obj.getBookPrice());
24     System.out.println("Author Name :" +obj.getAuthorName());
25 }
26 }
```

## **Book.java**

```

1 public class Book
2 {
3     private String bookName;
4     private int bookPrice;
5     private String authorName;
6
7     public void setBookName(String bookName)
8     {
9         this.bookName=bookName;
10    }
11
12    public String getBookName()
13    {
14        return this.bookName;
15    }
16
17    public void setBookPrice(int bookPrice)
18    {
19        this.bookPrice=bookPrice;
20    }
21
22    public int getBookPrice()
23    {
24        return this.bookPrice;
25    }
26
27    public void setAuthorName(String authorName)
28    {
29        this.authorName=authorName;
30    }
31
32    public String getAuthorName()
33    {
34        return this.authorName;
35    }
36 }
```

## **Student Details - Constructor**

```

public class Student
2 {
3     private int studentId;
4     private String studentName, studentAddress, collegeName;
5
6     public Student(int studentId, String studentName, String studentAddress)
7     {
```

```

8   this.studentAddress=studentAddress;
9   this.studentName=studentName;
10  this.studentId=studentId;
11  this.collegeName="NIT";
12 }
13
14 public Student(int studentId, String studentName, String studentAddress, String collegeName)
15 {
16     this(studentId, studentName, studentAddress); //invoking a constructor
17     this.collegeName=collegeName;
18 }
19
20 public int getStudentId()
21 {
22     return this.studentId;
23 }
24
25 public String getStudentName()
26 {
27     return this.studentName;
28 }
29
30 public String getStudentAddress()
31 {
32     return this.studentAddress;
33 }
34
35 public String getCollegeName()
36 {
37     return this.collegeName;
38 }
39 }
40

```

## StudentMain.java

```

1 import java.util.Scanner;
2 public class StudentMain
3 { public static void main (String[] args)
4  {
5
6     Scanner sc=new Scanner(System.in);
7     System.out.println("Enter Student's Id:");
8     int studentId=sc.nextInt();
9     sc.nextLine();
10
11    System.out.println("Enter Student's Name:");
12    String studentName=sc.nextLine();
13
14    System.out.println("Enter Student's address:");
15    String studentAddress=sc.nextLine();
16
17    while(true)
18    {
19        System.out.println("Whether the student is from NIT(Yes/No):");
20        String answer=sc.nextLine();
21
22        if(answer.equalsIgnoreCase("yes"))
23        {
24            Student obj = new Student(studentId, studentName, studentAddress);
25            System.out.println("Student id:"+obj.getStudentId());

```

```

26     System.out.println("Student name:"+obj.getStudentName());
27     System.out.println("Address:"+obj.getStudentAddress());
28     System.out.println("College name:"+obj.getCollegeName());
29     break;
30 }
31
32 else if(answer.equalsIgnoreCase("no"))
33 {
34     System.out.println("Enter the college name:");
35     String collegename=sc.nextLine();
36
37     Student obj = new Student(studentId, studentName, studentAddress, collegename);
38
39     System.out.println("Student id:"+obj.getStudentId());
40     System.out.println("Student name:"+obj.getStudentName());
41     System.out.println("Address:"+obj.getStudentAddress());
42     System.out.println("College name:"+obj.getCollegeName());
43     break;
44 }
45
46 else
47 {
48     System.out.println("Wrong Input");
49 }
50 }
51 }
52 }
53

```

## Volume calculator- Over Loading

```

import java.util.Scanner;
2
3 public class TestMain
4 {
5
6
7     public static void main(String[] args)
8     {
9         System.out.println("Enter the choice : ");
10        System.out.println("1.Find the volume for cylinder");
11        System.out.println("2.Find the volume for cuboid");
12        Scanner sc = new Scanner(System.in);
13        VolumeCalculator vc=new VolumeCalculator();
14        int choice = sc.nextInt();
15
16        switch(choice)
17        {
18            case 1:
19                System.out.println("Enter the radius : ");
20                double radius = sc.nextDouble();
21                System.out.println("Enter the height : ");
22                double height = sc.nextDouble();
23                System.out.println("Volume of the cylinder is :" + Math.round((vc.calculateVolume(radius, height)) *
100.0) / 100.0);
24                break;
25            case 2:
26                System.out.println("Enter the length : ");
27                int length = sc.nextInt();
28                System.out.println("Enter the breadth : ");
29                int breadth = sc.nextInt();

```

```

30
31     System.out.println("Enter the height : ");
32     int height1 = sc.nextInt();
33     System.out.println("Volume of the cuboid is :" +Math.round((vc.calculateVolume(length, breadth,
height1)) * 100.0) / 100.0);
34         break;
35     }
36
37 }
38
39 }
40

```

## Ticket Price Calculation – Static

### Main.java

```

import java.util.Scanner;
1 public class Main
2 {
3     public static void main (String[] args) {
4         Scanner sc=new Scanner(System.in);
5         Ticket obj=new Ticket();
6
7         System.out.println("Enter no of bookings:");
8         int no_bookings=sc.nextInt();
9
10        System.out.println("Enter the available tickets:");
11        obj.setAvailableTickets(sc.nextInt());
12
13        while(no_bookings>0)
14        {
15            System.out.println("Enter the ticketid:");
16            obj.setTicketid(sc.nextInt());
17
18            System.out.println("Enter the price:");
19            obj.setPrice(sc.nextInt());
20
21            System.out.println("Enter the no of tickets:");
22            int no_tickets=sc.nextInt();
23            if(obj.calculateTicketCost(no_tickets)==-1)
24            {
25                continue;
26            }
27
28            System.out.println("Available tickets: "+obj.getAvailableTickets());
29
30            System.out.println("Total amount:"+obj.calculateTicketCost(no_tickets));
31
32            System.out.println("Available ticket after booking:"+obj.getAvailableTickets());
33
34            no_bookings--;
35        }
36    }
37 }
38

```

### Ticket.java

```
public class Ticket
```

```

2{
3    private int ticketid;
4    private int price;
5    private static int availableTickets;
6
7    public void setTicketid(int ticketid)
8    {
9        this.ticketid=ticketid;
10   }
11
12    public int getTicketid()
13    {
14        return this.ticketid;
15   }
16
17    public void setPrice(int price)
18    {
19        this.price=price;
20   }
21
22    public int getPrice()
23    {
24        return this.price;
25   }
26
27    public static void setAvailableTickets(int availableTickets1)
28    { if(availableTickets1>=0)
29    {
30        availableTickets=availableTickets1;
31    }
32   }
33
34    public static int getAvailableTickets()
35    {
36        return availableTickets;
37   }
38
39    public int calculateTicketCost(int nooftickets)
40    {
41        if(availableTickets>=nooftickets)
42        {
43            availableTickets-=nooftickets;
44            return nooftickets*this.price;
45        }
46        else
47        {
48            return -1;
49        }
50    }
51}

```

## Employee Salary Calculation

```

import java.util.Scanner;
2 public class Main extends Employee {
3     public static Employee getEmployeeDetails() {
4         Scanner sc=new Scanner(System.in);
5         Employee emp=new Employee();
6         System.out.println("Enter Id:");
7         emp.setEmployeeId(sc.nextInt());
8         sc.nextLine();

```

```

9         System.out.println("Enter Name:");
10        emp.setEmployeeName(sc.nextLine());
11        System.out.println("Enter Salary:");
12        emp.setSalary(sc.nextDouble());
13
14
15        return emp;
16    }
17    public static int getPFPercentage() {
18        Scanner sc=new Scanner(System.in);
19        System.out.println("Enter PF percentage");
20        return sc.nextInt();
21    }
22    public static void main(String[] args) {
23        Employee emp=new Employee();
24        emp=getEmployeeDetails();
25        emp.calculateNetSalary(getPFPercentage());
26        System.out.println("Id: "+emp.getEmployeeId());
27        System.out.println("Name: "+emp.getEmployeeName());
28        System.out.println("Salary: "+emp.getSalary());
29        System.out.println("Net Salary: "+emp.getNetSalary());
30
31    }
32
33}
34
```

## Travel Details

### TestMain.java

```

1 import java.util.*;
2 public class TestMain
3 {
4     public static BusTicket getTicketDetails()
5     {
6         BusTicket bt= new BusTicket();
7         Scanner sc= new Scanner(System.in);
8         System.out.println("Enter the ticket no:");
9         bt.setTicketNo(sc.nextInt());
10        System.out.println("Enter the ticket price:");
11        bt.setTicketPrice(sc.nextFloat());
12        return bt;
13    }
14
15    public static void main (String[] args) {
16        Scanner sc= new Scanner(System.in);
17        Person p= new Person();
18        System.out.println("Enter the passenger name:");
19        p.setName(sc.nextLine());
20        System.out.println("Enter the gender(M or F / m or f):");
21        p.setGender(sc.next().charAt(0));
22        System.out.println("Enter the age:");
23        p.setAge(sc.nextInt());
24        BusTicket bt= getTicketDetails();
25        bt.setPerson(p);
26        p= bt.getPerson();
27        System.out.println("Ticket no:"+bt.getTicketNo());
28        System.out.println("Passenger Name:"+p.getName());
29        System.out.println("Price of a ticket : "+bt.getTicketPrice());
```

```
30     bt.calculateTotal();
31     System.out.println("Total Amount : "+bt.getTotalAmount());
32   }
33 }
34 }
```

## BusTicket.java

```
1
2  public class BusTicket
3  {
4      private int ticketNo;
5      private float ticketPrice;
6      private float totalAmount;
7      private Person person;
8      public void setPerson(Person person)
9      {
10         this.person=person;
11     }
12     public Person getPerson()
13     {
14         return person;
15     }
16     public void setTicketNo(int ticketNo)
17     {
18         this.ticketNo=ticketNo;
19     }
20     public void setTicketPrice(float ticketPrice)
21     {
22         this.ticketPrice=ticketPrice;
23     }
24     public void setTotalAmount(float totalAmount)
25     {
26         this.totalAmount= totalAmount;
27     }
28     public int getTicketNo()
29     {
30         return ticketNo;
31     }
32     public float getTicketPrice()
33     {
34         return ticketPrice;
35     }
36     public float getTotalAmount()
37     {
38         return totalAmount;
39     }
40     public void calculateTotal()
41     {
42         if(person.getAge()<=15)
43         {
44             setTotalAmount(ticketPrice/2);
45         }
46         else if(person.getAge()>=60)
47         {
48             setTotalAmount(ticketPrice*75/100);
49         }
50         else if(person.getGender()=='F'||person.getGender()=='F')
51         {
52             setTotalAmount(ticketPrice*90/100);
53         }
54     }
55 }
```

```

54     else
55     {
56         setTotalAmount(ticketPrice);
57     }
58 }
59 }
```

## BankAccountDetails

```

import java.util.*;
2 public class AccountDetails
3 {
4     public static Account getAccountDetails()
5     {
6         Account acc=new Account();
7         Scanner sc=new Scanner(System.in);
8         System.out.println("Enter account id: ");
9         acc.setAccountId(sc.nextInt());
10        sc.nextLine();
11        System.out.println("Enter account type: ");
12        acc.setAccountType(sc.nextLine());
13        int b;
14        do
15        {
16            System.out.println("Enter Balance");
17            acc.setBalance(sc.nextInt());
18            b=acc.getBalance();
19            if(b<=0)
20                System.out.println("Balance should be positive");
21            }
22            while(b<=0);
23            return acc;
24        }
25        public static int getWithdrawAmount()
26        {
27            Scanner sc=new Scanner(System.in);
28            int w;
29            do
30            {
31                System.out.println("Enter amount to be withdrawn:");
32                w=sc.nextInt();
33                if(w<=0)
34                    System.out.println("Amount should be positive");
35            }
36            while(w<=0);
37            return w;
38        }
39        public static void main(String[] args)
40        {
41            Account accObj=new Account();
42            accObj=getAccountDetails();
43            int c=getWithdrawAmount();
44            accObj.withdraw(c);
45        }
46 }
```

## Account.java

```

1 public class Account
2 {
3     private int accountId;
```

```

4  private String accountType;
5  private int balance;
6  public int getAccountId()
7  {
8      return accountId;
9
10 }
11 public String getAccountType()
12 {
13     return accountType;
14 }
15 public int getBalance()
16 {
17     return balance;
18 }
19 }
20 public void setAccountId(int id)
21 {
22     accountId=id;
23 }
24 public void setAccountType(String s)
25 {
26     accountType=s;
27 }
28 public void setBalance(int b)
29 {
30     balance=b;
31 }
32 public boolean withdraw(int w)
33 {
34     if(getBalance()<w)
35     {
36         System.out.println("Sorry!!! No enough balance");
37         return false;
38     }
39     else
40     {
41         System.out.println("Balance amount after withdraw: "+(getBalance()-w));
42         return true;
43     }
44 }
45 }

```

## Movie Ticket Calculation

### Main.java

```

1 import java.util.*;
2 public class Main{
3     public static Movie getMovieDetails(){
4         Movie mov=new Movie();
5         Scanner s=new Scanner(System.in);
6         System.out.println("Enter the movie name:");
7         mov.setMovieName(s.nextLine());
8         System.out.println("Enter the movie category:");
9         mov.setMovieCategory(s.nextLine());
10        return mov;
11    }
12    public static String getCircle(){
13        Scanner s=new Scanner(System.in);
14        System.out.println("Enter the circle:");

```

```

15     return(s.nextLine());
16 }
17 public static void main (String[] args) {
18     Scanner s=new Scanner(System.in);
19     Movie m=getMovieDetails();
20     String circle=getCircle();
21     int type=m.calculateTicketCost(circle);
22     System.out.println("Movie name = "+m.getMovieName());
23     System.out.println("Movie category = "+m.getMovieCategory());
24     if(type==0)
25         System.out.println("The ticket cost is = "+m.getTicketCost());
26     if(type==-1)
27         System.out.println("Sorry there is no "+m.getMovieCategory()+" type of category in theater.");
28     if(type==2)
29         System.out.println("Sorry!!! Circle is Invalid.");
30     if(type==3)
31         System.out.println("Sorry!!! Both circle and category are Invalid.");
32 }
33 }

```

## Movie.java

```

1 public class Movie{
2     private String movieName,movieCategory;
3     private int ticketCost;
4
5     public void setMovieName(String s){
6         movieName=s;
7     }
8     public void setMovieCategory(String s){
9         movieCategory=s;
10    }
11    public void setTicketCost(int i){
12        ticketCost=i;
13    }
14    public String getMovieName(){
15        return movieName;
16    }
17    public String getMovieCategory(){
18        return movieCategory;
19    }
20    public int getTicketCost(){
21        return ticketCost;
22    }
23    public int calculateTicketCost(String circle){
24        String cat=getMovieCategory();
25        if(circle.equalsIgnoreCase("GOLD")&& cat.equalsIgnoreCase("2d")){
26            setTicketCost(300);
27            //System.out.println("circle and cat=="+circle+cat);
28            return 0;
29        }
30        else if(circle.equalsIgnoreCase("GOLD")&& cat.equalsIgnoreCase("3d"))
31        {
32            setTicketCost(500);
33            return 0;
34        }
35        else if(circle.equalsIgnoreCase("SILVER")&&cat.equalsIgnoreCase("2d"))
36        {
37            setTicketCost(250);
38            return 0;
39        }

```

```

40     else if(circle.equalsIgnoreCase("SILVER")&&cat.equalsIgnoreCase("3d"))
41     {
42         setTicketCost(450);
43         return 0;
44     }
45     else if(circle.equalsIgnoreCase("GOLD")||circle.equalsIgnoreCase("SILVER"))
46     {
47         return -1;
48     }
49     else if(cat.equalsIgnoreCase("2d")||cat.equalsIgnoreCase("3d"))
50     {
51         return -2;
52     }
53     return -3;
54 }
55 }
```

## Average and Grade Calculation

### *Student.java*

```

1 public class Student
2 {
3     private int id;
4     private String name;
5     private int marks[];
6     private float average;
7     private char grade;
8     public Student(int a,String b,int[] c)
9     {
10         id=a;
11         name=b;
12         marks=c;
13     }
14     public void setId(int n)
15     {
16         id=n;
17     }
18     public int getId()
19     {
20         return id;
21     }
22     public void setMarks(int[] marks)
23     {
24         this.marks=marks;
25     }
26     public int[] getMarks()
27     {
28         return marks;
29     }
30     public void setName(String n)
31     {
32         name=n;
33     }
34 }
35     public String getName()
36     {
37         return name;
38     }
39     public void setAverage(float n)
40     {
```

```

41     average=n;
42 }
43 public float getAverage()
44 {
45     return average;
46 }
47 public void setGrade(char n)
48 {
49     grade=n;
50 }
51 public char getGrade()
52 {
53     return grade;
54 }
55 public void calculateAvg()
56 {
57     float a1=0.0F;
58     for(int a=0;a<this.getMarks().length;a++)
59     {
60         a1=a1+this.marks[a];
61     }
62     this.setAverage(a1/getMarks().length);
63 }
64 public void findGrade()
65 {
66     int min=this.marks[0];
67     for(int b=0;b<this.getMarks().length;b++)
68     {
69         if(this.marks[b]<min)
70         {
71             min=this.marks[b];
72         }
73     }
74     if(min<50)
75     {
76         this.setGrade('F');
77     }
78     else if(this.getAverage()>=80 && this.getAverage()<=100)
79     {
80         this.setGrade('O');
81     }
82     else
83     {
84         this.setGrade('A');
85     }
86 }
87 }
```

### **StudentMain.java**

```

1 import java.util.Scanner;
2 public class StudentMain
3 {
4     public static void main(String[] args)
5     {
6         Student s=getStudentDetails();
7         s.calculateAvg();
8         s.findGrade();
9         System.out.println("Id:" +s.getId());
10        System.out.println("Name:" +s.getName());
11        System.out.println("Average:" +String.format("%.2f",s.getAverage()));
```

```

12     System.out.println("Grade:" +s.getGrade());
13 }
14 public static Student getStudentDetails()
15 {
16     Scanner sc=new Scanner(System.in);
17     System.out.println("Enter the id:");
18     int id=Integer.parseInt(sc.nextLine());
19     String name=sc.nextLine();
20     System.out.println("Enter the no of subjects:");
21     int n=sc.nextInt();
22     if(n<=0)
23     {
24         while(n<=0)
25         {
26             System.out.println("Invalid number of subject");
27             System.out.println("Enter the no of subjects");
28             n=sc.nextInt();
29         }
30     }
31     int arr[]=new int[n];
32     for(int a=0;a<n;a++)
33     {
34         System.out.println("Enter mark for subject "+(a+1)+":");
35         int b=sc.nextInt();
36         if(b<0||b>100)
37         {
38             System.out.println("Invalid Mark");
39             System.out.println("Enter mark for subject "+(a+1)+":");
40             b=sc.nextInt();
41         }
42         arr[a]=b;
43     }
44     Student obj=new Student(id,name,arr);
45     obj.setId(id);
46     obj.setName(name);
47     return obj;
48 }
49 }
```

## Student and Department Detail

### TestMain.java

```

1 import java.util.*;
2 public class TestMain {
3     public static Department d1;
4     public static Student s1;
5     public static Student createStudent(){
6         Scanner sc = new Scanner(System.in);
7         s1 = new Student();
8         d1 = new Department();
9         System.out.println("Enter the Department id:");
10        int did = sc.nextInt();
11        sc.nextLine();
12        System.out.println("Enter the Department name:");
13        String dname=sc.nextLine();
14        System.out.println("Enter the Student id:");
15        int sid = sc.nextInt();
16        sc.nextLine();
17        System.out.println("Enter the Student name:");
18        String sname=sc.nextLine();
```

```

19 d1.setDid(did);
20 d1.setDname(dname);
21 s1.setSid(sid);
22 s1.setSname(sname);
23 s1.setDepartment(d1);
24 return s1;
25 }
26 public static void main(String args[]){
27 s1=TestMain.createStudent();
28 System.out.println("Department id:"+d1.getDid());
29 System.out.println("Department name:"+d1.getDname());
30 System.out.println("Student id:"+s1.getSid());
31 System.out.println("Student name:"+s1.getSname());
32 }
33 }
```

### **Student.java**

```

1 public class Student {
2     private int sid;
3     private String sname;
4     private Department department;
5     public void setSid(int sid) {
6         this.sid=sid;
7     }
8     public int getSid(){
9         return sid;
10    }
11    public void setSname(String sname) {
12        this.sname=sname;
13    }
14    public String getSname() {
15        return sname;
16    }
17    public void setDepartment(Department department){
18        this.department=department;
19    }
20    public Department getDepartment(){
21        return department;
22    }
23 }
```

### **Department.java**

```

1 public class Department {
2     private int did;
3     private String dname;
4     public void setDid(int did){
5         this.did=did;
6     }
7     public int getDid(){
8         return did;
9     }
10    public void setDname(String dname){
11        this.dname=dname;
12    }
13    public String getDname() {
14        return dname;
15    }
16 }
```

## Display array values

```
import java.util.Scanner;
2
3  public class Main
4  {
5      public static void main(String args[])
6      {
7          Scanner in = new Scanner(System.in);
8          System.out.println("Enter the array size");
9          int n = in.nextInt();
10
11         int arr[] = new int[n];
12         System.out.println("Enter the values");
13         for (int i = 0; i < n; i++) {
14             arr[i] = in.nextInt();
15         }
16
17         for (int i = 0; i < n; i++) {
18             System.out.println( arr[i]);
19         }
20     }
21
22 }
23 }
```

## Display array values in reverse order

```
import java.util.Scanner;
2  class Main
3  {
4      public static void main(String[] args)
5      {
6          int n, res,i,j=0;
7          Scanner s = new Scanner(System.in);
8          System.out.print("Enter the array size:");
9          n = s.nextInt();
10         int array[] = new int[n];
11         int rev[] = new int[n];
12         System.out.println("Enter the values ");
13         for( i=0; i < n; i++)
14         {
15             array[i] = s.nextInt();
16         }
17
18         for( i=n;i>0 ; i--,j++)
19         {
20             rev[j] = array[i-1];
21             System.out.println(rev[j]);
22         }
23     }
24 }
```

## Find & Display the position of a number

```
import java.util.Scanner;
2  public class Main
3  {
4      public static void main(String[] args)
```

```

5      {
6
7      int n, x, flag = 0, i = 0;
8      Scanner s = new Scanner(System.in);
9      System.out.println("Enter the array size,");
10     n = s.nextInt();
11     int a[] = new int[n];
12     System.out.println(" Enter the values");
13     for(i =0; i < n; i++)
14     {
15         a[i] = s.nextInt();
16     }
17     System.out.println(" Enter the number to find,");
18     x = s.nextInt();
19     for(i = 0; i < n; i++)
20     {
21         if(a[i] == x)
22         {
23             flag = 1;
24             break;
25         }
26         else
27         {
28             flag = 0;
29         }
30     }
31     if(flag == 1)
32     {
33         System.out.println((i + 1));
34     }
35     else
36     {
37         System.out.println( "0");
38     }
39 }
40 }
```

## Sort the values

### Main.java

```

1 import java.util.Scanner;
2
3 public class Main
4
5 {
6
7     public static void main(String[] args)
8
9     {
10
11         int n, temp;
12
13         Scanner s = new Scanner(System.in);
14
15         System.out.print("Enter the array size");
16
17         n = s.nextInt();
18
19         int a[] = new int[n];
```

```

20     System.out.println("Enter the values");
21
22     for (int i = 0; i < n; i++)
23     {
24
25         a[i] = s.nextInt();
26
27     }
28
29     for (int i = 0; i < n; i++)
30     {
31
32         for (int j = i + 1; j < n; j++)
33         {
34
35             if (a[i] > a[j])
36
37             {
38
39                 temp = a[i];
40
41                 a[i] = a[j];
42
43                 a[j] = temp;
44
45             }
46
47         }
48
49     }
50
51 }
52
53 }
54
55
56
57 for (int i = 0; i < n - 1; i++)
58 {
59
60     System.out.print(a[i] + ",");
61
62 }
63
64 System.out.print(a[n - 1]);
65
66 }
67
68 }
69 }
```

## Sort the values in descending order

### Main.java

```

1 import java.util.Scanner;
2 public class Main
3 {
4     public static void main(String[] args)
5     {
6         int n, temp;
```

```

7 Scanner s = new Scanner(System.in);
8 System.out.print("Enter the array size:");
9 n = s.nextInt();
10 int a[] = new int[n];
11 System.out.println("Enter the values:");
12 for (int i = 0; i < n; i++)
13 {
14     a[i] = s.nextInt();
15 }
16 for (int i = 0; i < n; i++)
17 {
18     for (int j = i + 1; j < n; j++)
19     {
20         if (a[i] < a[j])
21         {
22             temp = a[i];
23             a[i] = a[j];
24             a[j] = temp;
25         }
26     }
27 }
28 System.out.print("Descending Order:");
29 for (int i = 0; i < n - 1; i++)
30 {
31     System.out.print(a[i] + " ");
32 }
33 System.out.print(a[n - 1]);
34 }
35 }
```

## Find the Player Details

### *Player.java*

```

1
2 public class Player {
3
4     private int playerId;
5     private String playerName;
6     private int age;
7     private long phoneNumber;
8     public int getPlayerId() {
9         return playerId;
10    }
11    public void setPlayerId(int playerId) {
12        this.playerId = playerId;
13    }
14    public String getPlayerName() {
15        return playerName;
16    }
17    public void setPlayerName(String playerName) {
18        this.playerName = playerName;
19    }
20    public int getAge() {
21        return age;
22    }
23    public void setAge(int age) {
24        this.age = age;
25    }
26    public long getPhoneNumber() {
```

```

28             return phoneNumber;
29     }
30     public void setPhoneNumber(long phoneNumber) {
31         this.phoneNumber = phoneNumber;
32     }
33     public Player(int playerId, String playerName, int age, long phoneNumber) {
34         super();
35         this.playerId = playerId;
36         this.playerName = playerName;
37         this.age = age;
38         this.phoneNumber = phoneNumber;
39     }
40
41
42
43
44 }
45

```

## **PlayerUtility.java**

```

1 import java.io.*;
2 import java.util.*;
3 public class PlayerUtility{
4     public Player findPlayerDetailsById(Player []arr,int playerIdToSearch){
5         int arrlength=arr.length;
6         for(int i=0;i<arrlength;i++){
7             if(arr[i].getPlayerId()==playerIdToSearch){
8                 return arr[i];
9             }
10        }
11        return null;
12    }
13 }
14

```

## **Main.java**

```

1 import java.io.*;
2 import java.util.*;
3 public class Main{
4     public static void main(String args[]){
5         Player[] pObj=new Player[4];
6         pObj[0]=new Player(146,"Athul",16,9876543210);
7         pObj[1]=new Player(135,"Rakshana",16,9956231045);
8         pObj[2]=new Player(168,"Christy",16,9856471230);
9         pObj[3]=new Player(194,"Athulya",16,9768541230);
10
11         Scanner sc=new Scanner(System.in);
12         System.out.println("Enter the id to be searched");
13         int x=sc.nextInt();
14         PlayerUtility obj=new PlayerUtility();
15         Player finded=obj.findPlayerDetailsById(pObj,x);
16         if(finded==null)
17 {
18             System.out.println("No player found");
19 }
20 else
21 {
22     System.out.println("Name:"+finded.getPlayerName());
23     System.out.print("Phone number:"+finded.getPhoneNumber());
24 }
25

```

```
24
25     }
26 }
```

## Pass and Fail Count

### Count.java

```
1 import java.util.*;
2 public class Count {
3     public static void main (String[] args) {
4         Scanner sc=new Scanner(System.in);
5
6         System.out.println("Enter the no of subjects:");
7         int no_sub=sc.nextInt();
8
9         if(no_sub>0 && no_sub<=20)
10        {
11            int marks[]={new int[no_sub];
12
13            for(int i=0;i<no_sub;i++)
14            {
15                marks[i]=sc.nextInt();
16            }
17
18            int pass=0,fail=0;
19            for(int i=0;i<no_sub;i++)
20            {
21                if(marks[i]<50)
22                {
23                    fail++;
24                }
25                else
26                {
27                    pass++;
28                }
29            }
30
31            if(fail==0)
32            {
33                System.out.println("Ram passed in all subjects");
34            }
35            else if(pass==0)
36            {
37                System.out.println("Ram failed in all subjects");
38            }
39            else
40            {
41                System.out.println("Ram passed in "+pass+" subjects and failed in "+fail+" subjects");
42            }
43        }
44        else
45        {
46            System.out.println("Invalid input range");
47        }
48    }
49 }
50 }
```

# Search a Course

## Course.java

```
1 import java.util.*;
2 public class Course
3 {
4     public static void main (String[] args) {
5         Scanner sc=new Scanner(System.in);
6         System.out.println("Enter no of course:");
7         int no_crs=sc.nextInt();
8         if(no_crs>0)
9             {
10                 System.out.println("Enter course names:");
11
12                 String crs[]={new String[no_crs];
13
14                 for(int i=0;i<no_crs;i++)
15                 {
16                     crs[i]=sc.next();
17                 }
18
19                 System.out.println("Enter the course to be searched:");
20                 String srch=sc.next();
21
22                 int flag=0;
23                 for(int i=0;i<no_crs;i++)
24                 {
25                     if(srch.equals(crs[i]))
26                     {
27                         flag++;
28                     }
29                 }
30
31                 if(flag!=0)
32                 {
33                     System.out.println(srch+" course is available");
34                 }
35                 else
36                 {
37                     System.out.println(srch+" course is not available");
38                 }
39             }
40             else
41             {
42                 System.out.println("Invalid Range");
43             }
44
45     }
46 }
```

# Sum of the maximum and the minimum element

## Sum.java

```
1 import java.util.Scanner;
2
3
4 public class Sum {
```

```

6
7  public static void main(String[] args) {
8      Scanner sc = new Scanner(System.in);
9      System.out.println("Enter the size of an array:");
10     int arraySize = sc.nextInt();
11     //check if array size is less than or equal to zero
12     if(arraySize<=0)
13     {
14         //print the error message
15         System.out.println("Invalid Array Size");
16     }
17     else
18     {
19         //declare the array of size entered by user
20         int array[] = new int[arraySize];
21
22         //ask user to enter the array element
23         System.out.println("Enter the elements:");
24         for(int i=0;i<arraySize;i++)
25         {
26             array[i] = sc.nextInt();
27         }
28
29         int min =array[0];
30         int max = array[0];
31         //run for loop to calculate the min and max
32         for(int i=0;i<arraySize;i++)
33         {
34             if(array[i]<min)
35             {
36                 min = array[i];
37             }
38             if(array[i]>max)
39             {
40                 max = array[i];
41             }
42         }
43         //calculate the sum of min and max
44         int minMaxSum = min+max;
45         //print the sum
46         System.out.println(minMaxSum);
47     }
48     sc.close();
49
50
51 }
52
53
54 }
55

```

## Cumulative sum in an array

```

import java.util.Scanner;
2 import java.util.StringJoiner;
3
4 class CumulativeSum {
5     public static void main(String[] args) {
6         Scanner scanner = new Scanner(System.in);
7         int n;

```

```
8
9     System.out.println("Enter the number of elements");
10    n = scanner.nextInt();
11
12    if (n <= 0) {
13        System.out.println("Invalid Range");
14    } else {
15        int sum = 0;
16        StringJoiner stringJoiner = new StringJoiner(" ");
17
18        System.out.println("Enter the elements");
19
20        for (int i = 0; i < n; ++i) {
21            int num = scanner.nextInt();
22            sum += num;
23            stringJoiner.add(String.valueOf(sum));
24        }
25
26        System.out.println(stringJoiner.toString());
27    }
28}
29}
30}
31}
```

## Array Compatibility

## CompatibleArrays.java

```
1 import java.util.*;
2 public class CompatibleArrays
3 {
4     public static void main (String[] args) {
5         Scanner sc=new Scanner(System.in);
6         System.out.println("Enter the size for First array:");
7         int size1=sc.nextInt();
8         if(size1<=0)
9             System.out.println("Invalid array size");
10        else
11        {
12            int arr1[]={new int[size1]};
13            System.out.println("Enter the elements for First array:");
14            for(int i=0;i<size1;i++)
15                arr1[i]=sc.nextInt();
16            System.out.println("Enter the size for Second array:");
17            int size2=sc.nextInt();
18            if(size2<=0)
19                System.out.println("Invalid array size");
20            else
21            {
22                int count=0;
23                int arr2[]={new int[size2]};
24                System.out.println("Enter the elements for Second array:");
25                for(int i=0;i<size2;i++)
26                {
27                    arr2[i]=sc.nextInt();
28                }
29                if(size1==size2)
30                {
31                    for(int i=0;i<size1;i++)
```

```

32         {
33             if(arr1[i]>=arr2[i])
34                 count++;
35         }
36         if(count==(size1))
37             System.out.println("Arrays are Compatible");
38         else
39         {
40             System.out.println("Arrays are Not Compatible");
41         }
42     }
43     else
44     System.out.println("Arrays are not Compatible");
45 }
46 }
47 }
48 }
```

## Sort the first and second half of an array

### ArraySort.java

```

1 import java.util.*;
2 public class ArraySort
3 {
4     public static void main (String[] args) {
5         Scanner sc=new Scanner(System.in);
6
7         System.out.println("Enter the size of an array:");
8         int n=sc.nextInt();
9
10        if(n>0)
11        {
12            int[] arr=new int[n];
13            System.out.println("Enter the elements:");
14            for(int i=0;i<n;i++)
15            {
16                arr[i]=sc.nextInt();
17            }
18
19            int[] arr1=Arrays.copyOfRange(arr, 0, (n+1)/2);
20            int[] arr2=Arrays.copyOfRange(arr, (n+1)/2, n );
21
22            Arrays.sort(arr1);
23            //Arrays.sort(arr2, Collections.reverseOrder());
24
25            for(int i=0;i<n/2;i++)
26            {
27                for(int j=0;j<(n/2)-1;j++)
28                {
29                    if(arr2[j]<arr2[j+1])
30                    {
31                        int temp=arr2[j];
32                        arr2[j]=arr2[j+1];
33                        arr2[j+1]=temp;
34                    }
35                }
36            }
37
38            int len2=arr1.length+arr2.length;
39            int[] res=new int[len2];
```

```

40
41     System.arraycopy(arr1, 0, res, 0, (n+1)/2 );
42     System.arraycopy(arr2, 0, res, (n+1)/2, n/2 );
43
44     for(int i : res)
45     {
46         System.out.println(i);
47     }
48 }
49 else
50 {
51     System.out.println("Array size should be greater than 0");
52 }
53 }
54 }
```

## Highest Mark in Each Semester

```

import java.util.*;
2 import java.util.stream.Collectors;
3
4 class Semester {
5     private final int subjects;
6     List<Integer> marks;
7
8     public Semester(int subjects) {
9         this.subjects = subjects;
10        marks = new ArrayList<>(subjects);
11    }
12
13     public void setMarks(int[] marks) {
14         this.marks.addAll(Arrays.stream(marks).boxed().collect(Collectors.toList()));
15     }
16
17     public int getSubjects() {
18         return subjects;
19     }
20
21     public int getMaximumMarks() {
22         return Collections.max(marks);
23     }
24 }
25
26 class HighestMarkPerSem {
27     public static void main(String[] args) {
28         Scanner scanner = new Scanner(System.in);
29         int sems = 0;
30
31         System.out.println("Enter no of semester:");
32         sems = scanner.nextInt();
33
34         if (sems <= 0) {
35             System.out.println("Invalid Input");
36         } else {
37             List<Semester> semesters = new ArrayList<>();
38
39             for (int i = 0; i < sems; ++i) {
40                 int subjects;
41
42                 System.out.println("Enter no of subjects in " + (i + 1) + " semester:");
43                 subjects = scanner.nextInt();
44
45                 Semester s = new Semester(subjects);
46                 s.setMarks(new int[subjects]);
47
48                 for (int j = 0; j < subjects; ++j) {
49                     System.out.print("Enter mark for subject " + (j + 1) + ": ");
50                     int mark = scanner.nextInt();
51                     s.marks.add(mark);
52                 }
53
54                 semesters.add(s);
55             }
56
57             System.out.println("Highest mark per semester is: " + semesters.stream()
58                             .mapToInt(Semester::getMaximumMarks)
59                             .max()
60                             .orElse(0));
61         }
62     }
63 }
```

```

44     if (subjects < 0) {
45         System.out.println("Invalid Input");
46         System.exit(0);
47     } else {
48         Semester semester = new Semester(subjects);
49         semesters.add(semester);
50     }
51 }
52 }
53
54 for (int i = 0; i < sems; ++i) {
55     Semester semester = semesters.get(i);
56     int subjects = semester.getSubjects();
57     int[] marks = new int[subjects];
58
59     System.out.println("Marks obtained in semester " + (i + 1) + ":");
60
61     for (int j = 0; j < subjects; ++j) {
62         int mark = scanner.nextInt();
63
64         if (mark < 0 || mark > 100) {
65             System.out.println("You have entered invalid mark.");
66             System.exit(0);
67         } else {
68             marks[j] = mark;
69         }
70     }
71
72     semester.setMarks(marks);
73 }
74
75 for (int i = 0; i < semesters.size(); i++) {
76     System.out.println("Maximum mark in " + (i + 1) + " semester:" +
semesters.get(i).getMaximumMarks());
77 }
78 }
79 }
80 }

```

## Sum of factorial of positive and single digit numbers in an array

### Factorial.java

```

1 import java.util.*;
2
3
4 public class Factorial
5
6 {
7
8     public static void main(String[] args)
9
10    {
11
12
13
14
15
16    Scanner sc = new Scanner(System.in); //System.in is a standard input stream

```

```

17
18 System.out.println("Enter the size of an array:");
19
20 int size = sc.nextInt();
21
22 int[] arr = new int[size];
23 System.out.println("Enter the elements:");
24 for (int i = 0; i < size; i++)
25 {
26
27
28
29
30     int number = sc.nextInt();
31
32     arr[i] = number;
33
34 }
35
36
37
38 int sum = 0;
39
40
41
42 for (int i = 0; i < size; i++)
43 {
44
45     if (arr[i] < 10 && arr[i] > 0)
46     {
47
48         int val = calc(arr[i]);
49
50         sum += val;
51
52     }
53
54 }
55
56
57 if (sum == 0) {
58     System.out.println("No positive and single digit numbers found in an array");
59 } else {
60     System.out.println(sum);
61 }
62
63
64
65 }
66
67
68
69 static int calc(int val)
70 {
71
72     int i, fact = 1;
73
74     for (i = 1; i <= val; i++) {
75

```

```

76         fact = fact * i;
77     }
78
79     }
80
81
82
83     return fact;
84
85 }
86
87
88
89 }
90

```

## String length

### Main.java

```

1 import java.util.Scanner;
2
3 public class Main {
4
5     public static void main(String[] args){
6         Scanner sc=new Scanner(System.in);
7         String str=sc.nextLine();
8         int len=str.length();
9         if(len%2 ==0){
10             System.out.println(len+" even");
11         }
12         else{
13             System.out.println(len+" odd");
14         }
15     }
16 }
17

```

## Count of alphabets

```

import java.util.Scanner;
2
3 public class Main {
4
5     public static void main(String[] args){
6         Scanner sc=new Scanner(System.in);
7         String str=sc.nextLine();
8         int count=0;
9         for(int i=0;i<str.length();i++){
10             if(Character.isLetter(str.charAt(i)))
11             {
12                 count++;
13             }
14             System.out.println(count);
15         }
16     }
17
18 }
19
20
21

```

## SubString

```
2 import java.util.Scanner;
3
4 public class Main {
5
6     public static void main(String[] args){
7         Scanner sc=new Scanner(System.in);
8         String str=sc.nextLine();
9         int start=sc.nextInt();
10        int end=sc.nextInt();
11        System.out.println(str.substring(start,end));
12    }
13}
14}
15}
```

## String Contains

### Main.java

```
1 import java.util.Scanner;
2
3 public class Main {
4
5     public static void main(String[] args){
6         Scanner sc=new Scanner(System.in);
7         String str=sc.nextLine();
8         String sub_str=sc.nextLine();
9         if(str.indexOf(sub_str)!=-1){
10             System.out.println(sub_str+" is contained in a sentence");
11         }
12         else{
13             System.out.println(sub_str+" is not contained in a sentence");
14         }
15     }
16 }
17 }
18 }
```

## Compare two strings

### Main.java

```
1 import java.util.Scanner;
2
3 public class Main {
4
5     public static void main(String[] args){
6         Scanner sc=new Scanner(System.in);
7         String str1= sc.nextLine();
8         String str2=sc.nextLine();
9         if(str1.equals(str2))
10             System.out.println("Case sensitive");
11         else if(str1.equalsIgnoreCase(str2))
12             System.out.println("Case insensitive");
13         else
14             System.out.println("Not equal");
15     }
16 }
17 }
18 }
```

# String Spilt-Up

## Main.java

```
1 import java.util.*;
2
3 public class Main {
4
5     public static void main(String[] args){
6         Scanner sc=new Scanner(System.in);
7         String str=sc.nextLine();
8         String splt=sc.nextLine();
9
10        String[] splitstr=str.split(splt);
11        int len=splitstr.length;
12        for(int i=0;i<len;i++){
13            System.out.println(splitstr[i]);
14        }
15    }
16}
17}
18}
19}
20}
21}
```

# String - Find and replace the character (First occurrence)

## FirstOccurrence.java

```
1 import java.util.*;
2 public class FirstOccurrence
3 {
4     public static void main (String[] args) {
5         Scanner sc=new Scanner(System.in);
6         System.out.println("Enter the string:");
7         char[] str=sc.nextLine().toCharArray();
8
9         System.out.println("Enter the character to be searched:");
10        char srch=sc.next().charAt(0);
11
12        System.out.println("Enter the character to replace:");
13        char replace=sc.next().charAt(0);
14
15        int flag=0;
16        int len=str.length;
17        for(int i=0;i<len;i++)
18        {char r=str[i];
19         if(r==srch)
20         {
21             str[i]=replace;
22             flag++;
23             break;
24         }
25     }
26     if(flag==0)
27     {
28         System.out.println("character not found");
29     }
30 }
```

```

29     }
30     else
31     {
32         for(char i : str)
33         {
34             System.out.print(i);
35         }
36     }
37 }
38 }
```

## String Concatenation

### Authority.java

```

1 import java.util.*;
2 class Authority
3 {
4     public static void main (String[] args) {
5         Scanner sc=new Scanner(System.in);
6         System.out.print("Inmate's name:");
7         String name=sc.nextLine();
8         System.out.print("Inmate's father's name:");
9         String fname=sc.nextLine();
10        name=name.concat(" ").concat(fname);
11        char []chars=name.toCharArray();
12        for(char c:chars)
13        {
14            if(Character.isDigit(c)||c=='+'||c=='!'||c=='@'||c=='#'||c=='%'||c=='$'||c=='^'||c=='&'||c=='*'||c=='('
15            ||c==')'||c=='-'||c=='='||c=='/'||c=='<'||c=='>'||c=='?'||c=='{'||c==''}'||c=='['||c==']')
16            {
17                System.out.print("Invalid name");
18                System.exit(0);
19            }
20        }
21        System.out.println(name.toUpperCase());
22    }w
23 }
```

## Length of String

```

import java.util.*;
2
3 class FindLength {
4
5     public static void main(String[] args) {
6
7         String testString;
8
9         Scanner in = new Scanner(System.in);
10
11        System.out.println("Enter String : ");
12
13        testString = in.nextLine();
14
15        for (int i = 0; i < testString.length(); i++) {
16
17            if ((testString.charAt(i) >= 'A' && testString.charAt(i) <= 'Z') || (testString.charAt(i) >= 'a' &&
testString.charAt(i) <= 'z') || testString.charAt(i) == ' ')
18            {
19
```

```

20         continue;
21     } else
22     {
23         System.out.println("Invalid String.");
24         System.exit(0);
25     }
26 }
27 System.out.println("No. of characters is : " + testString.length());
28 }
29 }
30 }
31 }
32 }
33 }
34 }
35 }
36 }
37 }
38 }
39 }
40 }
41 }
```

## PAN Card Validation

### *PanCard.java*

```

1 import java.util.*;
2 public class PanCard{
3     public static void main (String[] args) {
4         Scanner in=new Scanner(System.in);
5         System.out.println("Enter the PAN no:");
6         String s1=in.nextLine();
7         if(s1.matches("[A-Z]{5}[0-9]{4}[A-Z]{1}"))
8         {
9             System.out.println("Valid PAN no");
10        }
11        else{
12            System.out.println("Invalid PAN no");
13        }
14    }
15 }
```

## Print unique characters

```

import java.util.*;
2
3 class UniqueChar {
4     private static boolean printUnique(String sentence) {
5         char[] chars = sentence.toCharArray();
6         Map<Character, Integer> map = new LinkedHashMap<>();
7
8         for (char ch : chars) {
9             if (Character.isDigit(ch)) {
10                 return false;
11             } else if (!Character.isWhitespace(ch)) {
12                 map.put(ch, map.getOrDefault(ch, 0) + 1);
13             }
14         }
15     List<Character> uniqueCharacters = new ArrayList<>();
```

```

17
18     for (char key : map.keySet()) {
19         if (map.get(key) == 1) {
20             uniqueCharacters.add(key);
21         }
22     }
23
24     if (uniqueCharacters.isEmpty()) {
25         System.out.println("No unique characters");
26     } else {
27         System.out.println("Unique characters:");
28
29         for (char ch : uniqueCharacters) {
30             System.out.println(ch);
31         }
32     }
33
34     return true;
35 }
36
37 public static void main(String[] args) {
38     Scanner scanner = new Scanner(System.in);
39     String sentence;
40
41     System.out.println("Enter the sentence:");
42     sentence = scanner.nextLine();
43
44     if (!printUnique(sentence)) {
45         System.out.println("Invalid Sentence");
46     }
47 }
48 }
49

```

## InitCap

### *InitCap.java*

```

1 import java.util.*;
2 public class InitCap
3 {
4     static void convert(String str)
5     {
6         int count=0;
7         char []ch=str.toCharArray();
8         for(int i=0;i<str.length();i++)
9         {
10            if(i==0&&ch[i]!=' '|&&ch[i]!=' '&&ch[i-1]==' ')
11            {
12                if(ch[i]>='a'&& ch[i]<='z')
13                {
14                    ch[i]=(char)(ch[i]-'a'+'A');
15                    count++;
16                }
17            }
18            else if(ch[i]>='A'&&ch[i]<='Z')
19                ch[i]=(char)(ch[i]+ 'a'-'A');
20        }
21        if(count==0)
22            System.out.println("First character of each word is already in uppercase");

```

```

23     else
24     {
25         String st=new String(ch);
26         System.out.println(st);
27     }
28 }
29 public static void main (String[] args) {
30     Scanner sc=new Scanner(System.in);
31     System.out.println("Enter the String:");
32     String str=sc.nextLine();
33     convert(str);
34 }
35 }
```

## Palindrome

### *Palindrome.java*

```

1 import java.util.*;
2 import java.util.regex.Matcher;
3 import java.util.regex.Pattern;
4 public class Palindrome
5 {
6     public static void main (String[] args) {
7         Scanner sc= new Scanner(System.in);
8
9         System.out.println("Enter the word :");
10        String str=sc.nextLine();
11        int len=str.length();
12
13        Pattern p=Pattern.compile("[^a-zA-Z]", Pattern.CASE_INSENSITIVE);
14        Matcher m=p.matcher(str);
15        boolean b=m.find();
16
17        if(b)
18        {
19            System.out.println("Invalid Input");
20        }
21        else
22        {
23            String rvs=new String();
24
25            for(int i=(len-1); i>=0; i--)
26            {
27                rvs=rvs+str.charAt(i);
28            }
29
30            if(str.equalsIgnoreCase(rvs))
31            {
32                System.out.println(str+" is a Palindrome");
33            }
34            else
35            {
36                System.out.println(str+" is not a Palindrome");
37            }
38        }
39    }
40 }
```

# Find and Replace

## ReplaceWord.java

```
1 import java.util.*;
2 public class ReplaceWord{
3     public static void main (String[] args) {
4         Scanner in=new Scanner(System.in);
5         System.out.println("Enter the String:");
6         String a=in.nextLine();
7         Character dot=a.charAt(a.length()-1);
8         System.out.println("Enter the word to be searched:");
9         String b=in.nextLine();
10        System.out.println("Enter the word to be replaced:");
11        String n=in.nextLine();
12        String res="";
13        int count=0;
14        String[] words=a.split("\\W");
15
16        for(String word:words){
17            if(word.equals(b)){
18                res=res+n+" ";
19                count=count+1;
20            }
21            else{
22                res=res+word+" ";
23            }
24        }
25        if(count>=1){
26            System.out.print(res.trim());
27            if(dot=='.')
28                System.out.print(".");
29        }
30        else{
31            System.out.print("The word "+b+" not found");
32        }
33    }
34 }
```

----- SET 3 -----

1) Find the sum of the numbers in the given input string array

Input{“2AA”,”12”,”ABC”,”c1a”}

Output:6 (2+1+2+1)

Note in the above array 12 must not be considered as such

i.e, it must be considered as 1,2

```
package Set3;
public class ClassSeT01 {
public static void main(String[] args) {
    String[] s1={"2AA","12","A2C","C5a"};
    getSum(s1);
}
public static void getSum(String[] s1) {
    int sum=0;
    for(int i=0;i<s1.length;i++)
        for(int j=0;j<s1[i].length();j++){
            char c=s1[i].charAt(j);
            if(Character.isDigit(c)){
                String t=String.valueOf(c);
                int n=Integer.parseInt(t);
                sum=sum+n; } }
    System.out.println(sum); }
```

2) Create a program to get the hashmap from the given input string array where the key for the hashmap is first three letters of array element in uppercase and the value of hashmap is the element itself

Input:{“goa”,”kerala”,”gujarat”} [string array]

Output:{ {GOA,goa},{KER,kerala},{GUJ,Gujarat} } [hashmap]

```
package Set3;
import java.util.*;
public class ClassSeT02 {
public static void main(String[] args) {
    String[] s1={"goa","kerala","gujarat"};
    putvalues(s1);
}
public static void putvalues(String[] s1) {
    ArrayList<String> l1=new ArrayList<String>();
    HashMap<String,String> m1=new HashMap<String,String>();
    ArrayList<String> l2=new ArrayList<String>();
    for(String s:s1)
        l1.add(s.toUpperCase().substring(0, 3));
    for(String s:s1)
        l2.add(s);
    for(int i=0;i<l1.size();i++)
        m1.put(l1.get(i),l2.get(i));
    System.out.println(m1);
}
```

3) String[] input1=[“Vikas”,”Lokesh”,Ashok]  
expected output String: “Vikas,Lokesh,Ashok”

```

package Set3;
public class ClassSeT03 {
    public static void main(String[] args) {
        String[] ip={"Vikas","Lokesh","Ashok"};
        System.out.println(getTheNamesinGivenFormat(ip));
    }
    public static String getTheNamesinGivenFormat(String[] ip) {
        StringBuffer sb=new StringBuffer();
        for(int i=0;i<ip.length;i++)
            sb.append(ip[i].append(','));
        sb.deleteCharAt(sb.length()-1);
        return sb.toString();
    }
}
-----||
```

#### 4) Email Validation

```

String input1="test@gmail.com"
    1)@ & . should be present;
    2)@ & . should not be repeated;
    3)there should be five characters between @ and .;
    4)there should be atleast 3 characters before @ ;
    5)the end of mail id should be .com;
```

```

package Set3;
import java.util.*;
public class ClassSeT04 {
    public static void main(String[] args) {
        String ip="test@gmail.com";
        boolean b=emailIdValidation(ip);
        if(b==true)
            System.out.println("valid mail Id");
        else
            System.out.println("not a valid Id");
    }
    public static boolean emailIdValidation(String ip) {
        int i=0;
        boolean b=false;
        StringTokenizer t=new StringTokenizer(ip,"@");
        String s1=t.nextToken();
        String s2=t.nextToken();
        StringTokenizer t1=new StringTokenizer(s2,".");
        String s3=t1.nextToken();
        String s4=t1.nextToken();
        if(ip.contains("@") && ip.contains(".")) {
            i++;
            if(i==1)
                if(s3.length()==5)
                    if(s1.length()>=3)
                        if(s4.equals("com"))
                            b=true;
        }
        return b;
    }
}
```

}

-----||  
5) Square root calculation of

$((x_1+x_2)*(x_1+x_2)) + ((y_1+y_2)*(y_1+y_2))$

o/p should be rounded off to int;

package Set3;

public class ClassSeT05 {

public static void main(String[] args) {

    int x1=4,x2=8;

    int y1=3,y2=5;

    sqrt(x1,x2,y1,y2);

}

public static void sqrt(int x1, int x2, int y1, int y2) {

    int op;

    op=(int) (Math.sqrt((x1+x2)\*(x1+x2)) + ((y1+y2)\*(y1+y2)));

    System.out.println(op);

}

}

-----||  
6) I/P hashmap<String String>{ "ram:hari","cisco:barfi","honeywell:cs","cts:hari" };

i/p 2="hari";

o/p string[]={ "ram","cts" };

package Set3;

import java.util.\*;

import java.util.Map.Entry;

public class ClassSeT06 {

public static void main(String[] args) {

    HashMap<String, String> m1=new HashMap<String, String>();

    m1.put("ram","hari");

    m1.put("cisco","barfi");

    m1.put("honeywell","cs");

    m1.put("cts","hari");

    String s2="hari";

    getvalues(m1,s2);

}

public static void getvalues(HashMap<String, String> m1, String s2) {

    ArrayList<String>l1=new ArrayList<String>();

    for(Entry<String, String> m:m1.entrySet()){

        m.getKey();

        m.getValue();

        if(m.getValue().equals(s2))

            l1.add(m.getKey());

    String[] op= new String[l1.size()];

    for(int i=0;i<l1.size();i++){

        op[i]=l1.get(i) ;

        System.out.println(op[i]);

}

}

-----||  
7) Input1={"ABX","ac","acd"};

Input2=3;

Output1=X\$d

```

package Set3;
import java.util.*;
public class ClassSeT07 {
    public static void main(String[] args) {
        String[] s1={"abc","da","ram","cat"};
        int ip=3;
        getStr(s1,ip);
    }
    public static void getStr(String[] s1, int ip) {
        String op=" ";
        String s2=" ";
        ArrayList<String> l1=new ArrayList<String>();
        for(String s:s1)
            if(s.length()==ip)
                l1.add(s);
        StringBuffer buff=new StringBuffer();
        for(String l:l1){
            s2=l.substring(l.length()-1);
            buff.append(s2).append("$");
        }
        op=buff.deleteCharAt(buff.length()-1).toString();
        System.out.println(op);
    }
}
-----||
```

8) INPUT1= helloworld  
 INPUT2= 2. delete the char,if rpted twice.  
 if occurs more than twice,leave the first occurence and delete the duplicate  
 O/P= helwrd;

```

package Set3;
public class ClassSeT08 {
    public static void main(String[] args) {
        String input1="HelloWorld";
        int input2=2;
        System.out.println(deletingtheCharOccuringTwice(input1,input2));
    }
    public static String deletingtheCharOccuringTwice(String input1, int input2) {
        StringBuffer sb=new StringBuffer(input1);
        int c=1;
        for(int i=0;i<sb.length();i++){
            c=1;
            for(int j=i+1;j<sb.length();j++)
                if(sb.charAt(i)==sb.charAt(j))
                    c++;
            if(c>=input2){
                for(int j=i+1;j<sb.length();j++)
                    if(sb.charAt(i)==sb.charAt(j))
                        sb.deleteCharAt(j);
                sb.deleteCharAt(i);
                i--;
            }
        }
        return sb.toString();
    }
}
```

9) String[] input={"100","111","10100","10","1111"} input2="10"  
output=2;count strings having prefix"10" but "10" not included in count  
operation-- for how many strings input2 matches the prefix of each string in input1

String[] input={"01","01010","1000","10","011"}  
output=3; count the strings having prefix"10","01" but "10","01" not included

```
package Set3;  
import java.util.*;  
public class ClassSeT09 {  
public static void main(String[] args) {  
    String[] ip={"100","111","10100","10","1111"};  
    gteCount(ip);  
}  
public static void gteCount(String[] ip) {  
    int op=0;  
    ArrayList<String> l1=new ArrayList<String>();  
    for(String s:ip)  
        if(s.startsWith("10") || s.startsWith("01") &&(s.length()>2))  
            l1.add(s);  
    op=l1.size();  
    System.out.println(op);  
}
```

10) input1=1 ,input2=2 ,input3=3 --- output=6;  
input1=1 ,input2=13,input3=3 --- output=1;  
input1=13,input2=2 ,input3=8 --- output=8;  
if value equal to 13,escape the value '13', as well as the next value to 13.  
sum the remaining values

```
package Set3;  
import java.util.*;  
public class ClassSeT10 {  
public static void main(String[] args) {  
    int ip1=13,ip2=2,ip3=8;  
    System.out.println(thirteenLapse(ip1,ip2,ip3));  
}  
public static int thirteenLapse(int ip1, int ip2, int ip3) {  
    List<Integer> l=new ArrayList<Integer>();  
    l.add(ip1);  
    l.add(ip2);  
    l.add(ip3);  
    int s=0;  
    for(int i=0;i<l.size();i++){  
        if(l.get(i)!=13)  
            s+=l.get(i);  
        if(l.get(i)==13)  
            i=i+1;}  
    return s;  
}
```

11) input="hello"  
output="hlo"; Alternative positions...

```
package Set3;
public class ClassSeT11 {
public static void main(String[] args) {
    String s="Hello";
    System.out.println(alternatingChar(s));
}
public static String alternatingChar(String s){
    StringBuffer sb=new StringBuffer();
    for(int i=0;i<s.length();i++)
        if(i%2==0)
            sb.append(s.charAt(i));
    return sb.toString();
}
```

12) Input1="Hello World"; output-----à “dello WorlH”.

```
package Set3;
public class ClassSeT12 {
public static void main(String[] args) {
    String s="Hello World";
    System.out.println(reArrangingWord(s));
}
public static String reArrangingWord(String s) {
    StringBuffer sb=new StringBuffer();
    sb.append(s.substring(s.length()-1));
    sb.append(s.substring(1, s.length()-1));
    sb.append(s.substring(0, 1));
    return sb.toString();
}
```

13) Collect no's frm list1 which is not present in list2  
& Collect no's frm list2 which is not present in list1  
and store it in output1[].  
ex: input1={1,2,3,4}; input2={1,2,3,5}; output1={4,5};

```
package Set3;
import java.util.*;
public class ClassSeT13 {
public static void main(String[] args) {
    List<Integer> l1=new ArrayList<Integer>();
    l1.add(1);
    l1.add(2);
    l1.add(3);
    l1.add(4);
    List<Integer> l2=new ArrayList<Integer>();
    l2.add(1);
    l2.add(2);
    l2.add(3);
    l2.add(5);
```

```

int o[]={};
for(int i:o)
System.out.println(i);
}
public static int[] commonSet(List<Integer> l1, List<Integer> l2) {
List<Integer> l3=new ArrayList<Integer>();
List<Integer> l4=new ArrayList<Integer>();

l3.addAll(l1);l4.addAll(l2);
l1.removeAll(l2);l4.removeAll(l3);
l1.addAll(l4);

int o[]={};
for(int j=0;j<o.length;j++)
o[j]=l1.get(j);
return o;
}
}

```

---

14) String array will be given.if a string is Prefix of an any other string in that array means count.

```

package Set3;
public class ClassSeT14 {
public static void main(String[] args) {
String[] a={"pinky","preethi","puppy","preeth","puppypreethi"};
System.out.println(namesWithPreFixes(a));
}
public static int namesWithPreFixes(String[] a) {
int n=0;
for(int i=0;i<a.length;i++)
for(int j=i+1;j<a.length;j++){
String s1=a[i];
String s2=a[j];
if(s2.startsWith(s1)||s1.startsWith(s2))
n++;
}
return n;
}
}

```

---

15) count the number of words in the string

Input string="i work in cognizant.";  
output=4;

```

package Set3;
import java.util.StringTokenizer;
public class ClassSeT15 {
public static void main(String[] args) {
String s="I work for cognizant";
System.out.println(noOfWordsInString(s));
}
public static int noOfWordsInString(String s) {
StringTokenizer t=new StringTokenizer(s," ");
return t.countTokens();
}

```

```
}
```

---

```
16) int[] input={2,1,4,1,2,3,6};
```

check whether the input has the sequence of "1,2,3". if so-  
output=true;  
int[] input={1,2,1,3,4,5,8};  
output=false

```
package Set3;
```

```
public class ClassSeT16 {  
    public static void main(String[] args) {
```

```
        //int[] a={2,1,4,1,2,3,6};  
        int[] a={1,2,1,3,4,5,8};  
        System.out.println(sequenceInArray(a));
```

```
    }  
    public static boolean sequenceInArray(int[] a) {
```

```
        boolean b=false;  
        int n=0;  
        for(int i=0;i<a.length-1;i++)  
            if((a[i+1]-a[i])==1)  
                n++;  
        if(n==2)  
            b=true;  
        return b;
```

```
}
```

```
}
```

---

```
17) input-- String input1="AAA/abb/CCC"
```

char input2='/'  
output-- String[] output1;  
output1[]={"aaa","bba","ccc"};

operation-- get the strings from input1 using StringTokenizer

reverse each string  
then to lower case  
finally store it in output1[] string array

```
package Set3;
```

```
import java.util.*;
```

```
public class ClassSeT17 {
```

```
    public static void main(String[] args) {
```

```
        String ip1="AAA/abb/CCC";  
        char ip2='/';  
        String op[]=loweringCasenReverseofaString(ip1,ip2);  
        for(String s:op)  
            System.out.println(s);
```

```
}
```

```
    public static String[] loweringCasenReverseofaString(String ip1, char ip2){
```

```
        List<String> l=new ArrayList<String>();  
        StringTokenizer t=new StringTokenizer(ip1,"/");  
        while(t.hasMoreTokens()) {  
            StringBuffer sb=new StringBuffer(t.nextToken().toLowerCase());  
            l.add(sb.reverse().toString()); }  
        String op[]=new String[l.size()];
```

```
        for(int i=0;i<op.length;i++)
            op[i]=l.get(i);
        return op;
    }
}
```

||

18) Input1="cowboy"; Output1="cowcow";  
Input1="so";output1="sososo";  
HINT: if they give 3 letter word u have to display 2 time;

package Set3;

```
public class ClassSeT18 {
public static void main(String[] args) {
    String ip1="cowboy";
    String ip2="cow";
    System.out.println(printingStringDependingOncharCount(ip1,ip2));
}
public static String printingStringDependingOncharCount(String ip1,String ip2) {
    StringBuffer sb=new StringBuffer();
    int n1=ip2.length();
    if(n1==3)
        for(int i=0;i<n1-1;i++)
            sb.append(ip1.substring(0, n1));
    else if(n1==2)
        for(int i=0;i<n1+1;i++)
            sb.append(ip1.substring(0, n1));
    return sb.toString();
}
}
```

||

19) input---input1=1;
input2=4;
input3=1;
output1=4;
operation--- print the element which is not repeated
if all the inputs r different sum all inputs
input---input1=1;
input2=2;
input3=3;
output1=6;

```
package Set3;
public class ClassSeT19 {
public static void main(String[] args) {
    int ip1=1,ip2=4,ip3=1;
    //int ip1=1,ip2=2,ip3=3;
    //int ip1=1,ip2=1,ip3=1;
    System.out.println(sumOfNonRepeatedChars(ip1,ip2,ip3));
}
public static int sumOfNonRepeatedChars(int ip1, int ip2, int ip3){
    int n=0;
    if(ip1!=ip2 && ip2!=ip3 && ip3!=ip1)
        n=ip1+ip2+ip3;
}
```

||

```

        else if(ip1==ip2 && ip2==ip3)
            n=0;
        else{
            if(ip1==ip2)
                n=ip3;
            else if(ip1==ip3)
                n=ip2;
            else if(ip2==ip3)
                n=ip1; }
        return n;
}
}
-----||
```

20) input1-List1-{apple,orange,grapes}  
 input2-List2-{melon,apple,mango}  
 output={mango,orange}  
 operation-- In 1st list remove strings starting with 'a' or 'g'  
 In 2nd list remove strings ending with 'n' or 'e'  
 Ignore case, return in string array

```

package Set3;
import java.util.*;
public class ClassSeT20 {
public static void main(String[] args) {
    List<String> l1=new ArrayList<String>();
    l1.add("apple");
    l1.add("orange");
    l1.add("grapes");
    List<String> l2=new ArrayList<String>();
    l2.add("melon");
    l2.add("apple");
    l2.add("mango");
    String[] s2=fruitsList(l1,l2);
    for(String s3:s2)
        System.out.println(s3);
}
public static String[] fruitsList(List<String> l1, List<String> l2){
    List<String> l3=new ArrayList<String>();
    for(int i=0;i<l1.size();i++){
        String s1=l1.get(i);
        if(s1.charAt(0)!= 'a' && s1.charAt(0)!= 'A' && s1.charAt(0)!= 'g' && s1.charAt(0)!= 'G')
            l3.add(s1); }
    for(int i=0;i<l2.size();i++){
        String s1=l2.get(i);
        if(s1.charAt(s1.length()-1)!= 'n' && s1.charAt(s1.length()-1)!= 'N' && s1.charAt(s1.length()-1)!= 'e' &&
s1.charAt(s1.length()-1)!= 'E')
            l3.add(s1); }
    Collections.sort(l3);
    String[] s2=new String[l3.size()];
    for(int i=0;i<s2.length;i++)
        s2[i]=l3.get(i);
    return s2;
}
}
```

21) input1-- Hello\*world  
output-- boolean(true or false)  
operation-- if the character before and after \* are same return true else false  
if there is no star in the string return false(Ignore case)

```
package Set3;  
import java.util.*;  
public class ClassSeT21 {  
    public static void main(String[] args) {  
        String input="Hello*world";  
        System.out.println(characterCheck(input));  
    }  
    public static boolean characterCheck(String input) {  
        boolean b=false;  
        StringTokenizer t=new StringTokenizer(input,"*");  
        String s1=t.nextToken();  
        String s2=t.nextToken();  
        String s3=s1.substring(s1.length()-1);  
        String s4=s2.substring(0,1);  
        if(s3.equalsIgnoreCase(s4))  
            b=true;  
        return b;  
    }  
}
```

22) input --String input1 ="xaXafxsd"  
output--String output1="aXafsdxx"  
operation-- remove the character "x"(only lower case) from string and place at the end

```
package Set3;  
public class ClassSeT22 {  
    public static void main(String[] args) {  
        String input="xaXafxsd";  
        System.out.println(removalOfx(input));  
    }  
    public static String removalOfx(String input) {  
        StringBuffer sb=new StringBuffer(input);  
        int j=0;  
        for(int i=0;i<sb.length();i++)  
            if(sb.charAt(i)=='x') {  
                sb.deleteCharAt(i);  
                j++;  
            }  
        for(int i=0;i<j;i++)  
            sb.append('x');  
        return sb.toString();  
    }  
}
```

23) HashMap<String, Integer> h1={“abc”:50,”efg”:70};  
if the mark is less than 60 then put the output in the  
HashMap<String, String> h2={“abc”：“fail”, ”efg”：“pass”}

```
package Set3;
```

```

import java.util.*;
public class ClassSeT23 {
public static void main(String[] args) {
    Map<String, Integer> m1=new HashMap<String, Integer>();
    m1.put("abc", 90);
    m1.put("efg", 50);
    m1.put("mno", 60);
    m1.put("rst", 75);
    m1.put("xyz", 35);
    System.out.println(examResult(m1));
}
public static Map<String, String> examResult(Map<String, Integer> m1) {
    Map<String, String> m2=new HashMap<String, String>();
    String s1=new String();
    String s2=new String();
    int n=0;
    Iterator<String> i=m1.keySet().iterator();
    while(i.hasNext()){
        s1=(String) i.next();
        n=m1.get(s1);
        if(n>=60)
            s2="PASS";
        else
            s2="FAIL";
        m2.put(s1, s2); }
    return m2;
}
}

```

-----||  
24) String i/p1=2012;  
sTRING i/p2=5  
IF EXPERIENCE IS GREATER THAN INPUT 2 THEN TRUE;

```

package Set3;
import java.text.*;
import java.util.*;
public class ClassSeT24 {
public static void main(String[] args) throws ParseException {
    String ip1="2012";
    String ip2="5";
    System.out.println(experienceCalc(ip1,ip2));
}
public static boolean experienceCalc(String ip1, String ip2) throws ParseException {
    boolean b=false;
    SimpleDateFormat sdf=new SimpleDateFormat("yyyy");
    Date d1=sdf.parse(ip1);
    Date d2=new Date();
    int n1=d1.getYear();
    int n2=d2.getYear();
    int n3=Integer.parseInt(ip2);
    if((n2-n1)>n3)
        b=true;
    return b;
}

```

}

-----||  
25) input string="hello", n=2

output: lolo

```
package Set3;
public class ClassSeT25 {
public static void main(String[] args) {
    String s1="hello";
    int n1=2;
    System.out.println(formattingOfString(s1,n1));
}
public static String formattingOfString(String s1, int n1) {
    String s2=s1.substring(s1.length()-n1, s1.length());
    StringBuffer sb=new StringBuffer();
    for(int i=0;i<n1;i++)
        sb.append(s2);
    return sb.toString();
}
}
```

-----||  
26) prove whether a number is ISBN number or not

input="0201103311"

ISBN number: sum=0\*10 +2\*9+ 0\*8 +1\*7+ 1\*6 +0\*5+ 3\*4 +3\*3+ 1\*2 +1\*1

sum%11==0 then it is ISBN number

```
package Set3;
public class ClassSeT26 {
public static void main(String[] args) {
    String ip="0201103311";
    boolean b=ISBNnumber(ip);
    if(b==true)
        System.out.println("valid ISBN number");
    else
        System.out.println("check ur data");
}
public static boolean ISBNnumber(String ip) {
    boolean b=false;
    int sum=0;
    for(int i=0,j=ip.length();i<ip.length();i++,j--){
        String s=String.valueOf(ip.charAt(i));
        int n=Integer.parseInt(s);
        sum+=(n*j);
    }
    //System.out.println(sum);
    if(sum%11==0)
        b=true;
    return b;
}
}
```

-----||  
27) Validate Password

validation based on following criteria:

-> minimum length is 8

-> should contain any of these @/\_/#

- > should not start with number/special chars(@/#/\_)
- > should not end with special chars
- > can contain numbers,letters,special chars

```

package Set3;
import java.util.*;
public class ClassSeT40 {
public static void main(String[] args) {
    Scanner s=new Scanner(System.in);
    String s1=s.next();
    boolean b=passwordValidation(s1);
    if(b==true)
        System.out.println("valid password");
    else
        System.out.println("not a valid password");
}
public static boolean passwordValidation(String s1) {
    boolean b=false,b1=false,b2=false;
    if(s1.length()>=8)
        if(!Character.isDigit(s1.charAt(0)))
            if(s1.charAt(0)!='@' && s1.charAt(0)!='_' && s1.charAt(0)!='#')
                if(s1.charAt(s1.length()-1)!='@' && s1.charAt(s1.length()-1)!='_' && s1.charAt(s1.length()-1)!='#')
                    b1=true;
        if(b1==true)
            for(int i=0;i<s1.length();i++)
                if(Character.isAlphabetic(s1.charAt(i)) || Character.isDigit(s1.charAt(i)) || s1.charAt(i)=='#' || s1.charAt(i)=='@' || s1.charAt(i)=='_')
                    b2=true;
        if(b2==true)
            if(s1.contains("#") || s1.contains("@") || s1.contains("_"))
                b=true;
    return b;
}
}
-----||
```

## 28) pan card number validation:

all letters shud be in caps,shud be of 8 chars.  
first three letters must be alphabets.  
next 4 letters shud be digits and last letter shud be an alphabet

```

package Set3;
import java.util.*;
public class ClassSeT28 {
public static void main(String[] args) {
    Scanner s=new Scanner(System.in);
    String pan=s.next();
    boolean b=panNumberValidation(pan);
    if(b==true)
        System.out.println("valid Pancard Number");
    else
        System.out.println("not a valid credential");
}
public static boolean panNumberValidation(String pan) {
```

```

boolean b=false,b1=false,b2=false;
String s1=pan.substring(0, 3);
String s2=pan.substring(3, 7);
if(pan.length()==8)
    if(Character.isAlphabetic(pan.charAt(pan.length()-1)) && Character.isUpperCase(pan.charAt(pan.length()-1)))
        b1=true;
if(b1==true)
    for(int i=0;i<s1.length();i++)
        if(Character.isAlphabetic(s1.charAt(i)) && Character.isUpperCase(s1.charAt(i)))
            b2=true;
        else
            {b2=false;break;}
if(b2==true)
    for(int i=0;i<s2.length();i++)
        if(Character.isDigit(s2.charAt(i)))
            b=true;
        else
            {b=false;break;}
return b;
}
}
-----||
```

29) In a hashmap if key is odd then find average of value as integer

ex: h1={ 1:4,2:6,4:7,5:9}  
output=(4+9)/2

```

package Set3;
import java.util.*;
public class ClassSeT29 {
public static void main(String[] args) {
    Map<Integer, Integer> m1=new HashMap<Integer, Integer>();
    m1.put(1, 4);
    m1.put(2, 6);
    m1.put(4, 7);
    m1.put(5, 9);
    System.out.println(avgValuesOfOddKeys(m1));
}
public static int avgValuesOfOddKeys(Map<Integer, Integer> m1) {
    int l=0,m=0;
    Iterator<Integer> i=m1.keySet().iterator();
    while(i.hasNext()){
        int n=(Integer) i.next();
        if(n%2!=0){
            m+=m1.get(n);
            l++;} }
    return m/l;
}
}
-----||
```

30) Return 1 if the last & first characters of a string are equal else

return -1. Consider case.

Eg: Input = "this was great"

Output= 1

```
package Set3;
public class ClassSeT30 {
    public static void main(String[] args) {
        String input="this was great";
        System.out.println(checkForFirstAndLastChar(input));
    }
    public static int checkForFirstAndLastChar(String input) {
        int n=0;
        if(input.charAt(0)==input.charAt(input.length()-1))
            n=1;
        else n=-1;
        return n;
    }
}
```

-----||  
31) concat two string if length of two string is equal.

if length of one string is greater, then remove the character from largest string and then add. The number of characters removed from largest string is equal to smallest string's length  
for example: input 1="hello";  
              input 2="helloworld";  
              output="worldhello";

```
package Set3;
public class ClassSeT31 {
    public static void main(String[] args) {
        String ip1="hello";
        String ip2="helloworld";
        System.out.println(removalOfCharFromLargestString(ip1,ip2));
    }
    public static String removalOfCharFromLargestString(String ip1,String ip2){
        StringBuffer sb=new StringBuffer();
        int n1=ip1.length();
        int n2=ip2.length();
        if(n1<n2)
            sb.append(ip2.substring(n1, n2)).append(ip1);
        return sb.toString();
    }
}
```

-----||  
32) i/p: Honesty is my best policy

o/p: Honesty  
Return the maximum word length from the given string.  
If there are two words of same length then,  
return the word which comes first based on alphabetical order.

```
package Set3;
import java.util.*;
public class ClassSeT32 {
    public static void main(String[] args) {
        String s1="is a poppppy preethi";
        System.out.println(lengthiestString(s1));
```

```

}
public static String lengthiestString(String s1) {
    int max=0;
    String s2=new String();
    StringTokenizer t=new StringTokenizer(s1," ");
    loop:
    while(t.hasMoreTokens()){
        String s3=t.nextToken();
        int n=s3.length();
        if(n>max){
            max=n;
            s2=s3;
        }
        if(n==max)
            for(int i=0;i<s3.length();i++){
                char c1=s2.charAt(i);
                char c2=s3.charAt(i);
                if(c1!=c2){
                    if(c2<c1)
                        s2=s3;
                    continue loop; }
            }
    }
    return s2;
}
}
-----||
```

33) In a string check whether all the vowels are present

if yes return 1 else 2.

ex: String 1="education"

output=1.

```

package Set3;
public class ClassSeT33 {
public static void main(String[] args) {
    String s1="education";
    System.out.println(vowelsCheck(s1));
}
public static boolean vowelsCheck(String s1) {
    boolean b=false;
    int n1=0,n2=0,n3=0,n4=0,n5=0;
    for(int i=0;i<s1.length();i++){
        char c=s1.charAt(i);
        if(c=='a' || c=='A')
            n1++;
        if(c=='e' || c=='E')
            n2++;
        if(c=='i' || c=='I')
            n3++;
        if(c=='o' || c=='O')
            n4++;
        if(c=='u' || c=='U')
            n5++;}
    if(n1==1 && n2==1 && n3==1 && n4==1 && n5==1)
        b=true;
    return b;
}
```

```
}
```

-----||  
34) swap the every 2 characters in the given string

If size is odd number then keep the last letter as it is.

Ex:- input: forget

output: ofgrte

Ex:- input : NewYork

output : eNYwrok

```
package Set3;
public class ClassSet34 {
public static void main(String[] args) {
    String s1="newyork";
    System.out.println(formattingGivenString(s1));
}
public static String formattingGivenString(String s1) {
    StringBuffer sb=new StringBuffer();
    int j=0;
    if(s1.length()%2==0)
        j=s1.length();
    else
        j=s1.length()-1;

    for(int i=0;i<j;i+=2){
        String s2=(s1.substring(i, i+2));
        StringBuffer sb1=new StringBuffer(s2);
        sb.append(sb1.reverse());}

    String s3=new String();
    if(s1.length()%2==0)
        s3=sb.toString();
    else
        s3=sb.append(s1.charAt(s1.length()-1)).toString();
    return s3;
}
}
```

-----||  
35) i/p: bengal

o/p: ceogbl

if z is there replace with a

```
package Set3;
public class ClassSeT35 {
public static void main(String[] args) {
    String s1="bengal";
    System.out.println(stringFormatting(s1));
}
public static String stringFormatting(String s1) {
    StringBuffer sb=new StringBuffer();
    for(int i=0;i<s1.length();i++){
        char c=s1.charAt(i);
        if(i%2==0){
            if(c==122)
```

```

        c=(char) (c-25);
    else{
        c=(char) (c+1);}
    sb.append(c);}
else
    sb.append(c);}
return sb.toString();
}
}
-----||
```

36) find the maximum chunk of a given string

i/p: this isssss soooo good  
o/p=5

```

package Set3;
import java.util.*;
public class ClassSeT36 {
public static void main(String[] args) {
    String s1="this is sooo good";
    System.out.println(maxChunk(s1));
}
public static int maxChunk(String s1) {
    int max=0;
    StringTokenizer t=new StringTokenizer(s1, " ");
    while(t.hasMoreTokens()){
        String s2=t.nextToken();
        int n=0;
        for(int i=0;i<s2.length()-1;i++)
            if(s2.charAt(i)==s2.charAt(i+1))
                n++;
        if(n>max)
            max=n;
    }
    return (max+1);
}
}
```

37) i/p1: new york

i/p2: new jersey  
o/p: new y+r+

solution:

```

package Set3;
public class ClassSeT37 {
public static void main(String[] args) {
    String s1="New york";
    String s2="New jersey";
    System.out.println(commonCharsinAString(s1,s2));
}
private static String commonCharsinAString(String s1, String s2) {
    int f;
    StringBuffer sb=new StringBuffer();
    for(int i=0;i<s1.length();i++){
        f=0;

```

```

        char c1=s1.charAt(i);
        for(int j=0;j<s2.length();j++)
            if(c1==s2.charAt(j))
                f=1;
        if(f!=0)
            sb.append(c1);
        else
            sb.append('+');
    return sb.toString();
}
}
-----||
```

38) input1={2,4,3,5,6};  
if odd find square  
if even find cube  
finally add it  
output1=208(4+16+27+125+36)

```

package Set3;
public class ClassSeT38 {
public static void main(String[] args) {
    int a[]={2,4,3,5,6};
    System.out.println(summationPattern(a));
}
public static int summationPattern(int[] a) {
    int n1=0,n2=0;
    for(int i=0;i<a.length;i++)
        if(a[i]%2==0)
            n1+=(a[i]*a[i]);
        else
            n2+=(a[i]*a[i]*a[i]);
    return n1+n2;
}
}
-----||
```

39) input1="the sun rises in the east";  
output1=raises;  
count no vowels in each word and print the word which has max  
no of vowels if two word has max no of vowel print the first one

```

package Set3;
import java.util.*;
public class ClassSeT39 {
public static void main(String[] args) {
    String s1="the sun rises in the east";
    System.out.println(wordWithMaxVowelCount(s1));
}
public static String wordWithMaxVowelCount(String s1) {
    int max=0;
    String s2="aeiouAEIOU";
    String s3=new String();
    StringTokenizer t=new StringTokenizer(s1, " ");
    while(t.hasMoreTokens()){
        String s4=t.nextToken();
        int count=0;
        for(int i=0;i<s2.length();i++)
            if(s4.indexOf(s2.substring(i, i+1))>-1)
                count++;
        if(count>max)
            max=count;
    }
    for(int i=0;i<t.countTokens();i++)
        if(max==t.nextToken().length())
            s3+=t.nextToken();
    return s3;
}
}
-----||
```

```

int c=0;
for(int i=0;i<s4.length();i++)
    for(int j=0;j<s2.length();j++)
        if(s4.charAt(i)==s2.charAt(j))
            c++;
if(c>max){
    max=c;
    s3=s4; } }
return s3;
}
}

```

---

40) String format : CTS-LLL-XXXX

ip1: CTS-hyd-1234  
ip2: hyderabad  
-> LLL must be first 3 letters of ip2.  
-> XXXX must be a 4-digit number

```

package Set3;
import java.util.*;
public class ClassSeT41 {
public static void main(String[] args) {
    String s1="CTS-hyd-1234";
    String s2="hyderabad";
    boolean b=formattingString(s1,s2);
    if(b==true)
        System.out.println("String format:CTS-LLL-XXXX");
    else
        System.out.println("not in required format");
}
public static boolean formattingString(String s1, String s2) {
    String s3=s2.substring(0, 3);
    boolean b=false;
    StringTokenizer t=new StringTokenizer(s1,"-");
    String s4=t.nextToken();
    String s5=t.nextToken();
    String s6=t.nextToken();
    if(s4.equals("CTS") && s5.equals(s3) && s6.length()==4)
        for(int i=0;i<s6.length();i++){
            if(Character.isDigit(s6.charAt(i)))
                b=true;
            else{
                b=false; } }
    return b;
}
}

```

---

41) ip: "this is sample test case"

op: "this amplec"  
remove the duplicates in the given string

```

package Set3;
import java.util.*;
public class ClassSeT42 {

```

```

public static void main(String[] args) {
    String s1="this is sample test case";
    System.out.println(removeDuplicates(s1));
}
public static String removeDuplicates(String s1) {
    StringBuffer sb=new StringBuffer();
    Set<Character> c1=new LinkedHashSet<Character>();
    for(int i=0;i<s1.length();i++)
        c1.add(s1.charAt(i));
    for(char c2:c1)
        sb.append(c2);
    return sb.toString();
}
}

```

42) input1 is a map<Integer,Float>  
{1:2.3,2:5.6,3:7.7,4:8.4}  
get the even number from keys and find the avg of values of these keys.  
answer should be rounded to two numbers after decimal  
eg:- the output number 15.2499999 should be 15.25

```

package Set3;
import java.util.*;
public class ClassSeT43 {
public static void main(String[] args) {
    Map<Integer,Float> m1=new HashMap<Integer, Float>();
    m1.put(1, (float) 12.93);
    m1.put(2, (float) 15.67);
    m1.put(3, (float) 17.27);
    m1.put(4, (float) 14.88);
    System.out.println(avgOfEvenKeyValues(m1));
}
public static float avgOfEvenKeyValues(Map<Integer, Float> m1) {
    int n1=0,n3=0;
    float n2=0;
    Iterator<Integer> i=m1.keySet().iterator();
    while(i.hasNext()){
        n1=(Integer) i.next();
        if(n1%2==0){
            n2+=m1.get(n1);
            n3++;} }
    float n=Math.round((n2/n3)*100)/100f;
    return n;
}
}

```

43) Color Code Validation:  
String should starts with the Character '#'.  
Length of String is 7.  
It should contain 6 Characters after '#' Symbol.  
It should contain Characters Between 'A-F' and Digits '0-9'.  
if String is acceptable then Output1=1  
else Output1=-1;

```

package Set3;
import java.util.*;
public class ClassSeT44 {
public static void main(String[] args) {
    Scanner s=new Scanner(System.in);
    String s1=s.next();
    boolean b=colorCodeValidation(s1);
    if(b==true)
        System.out.println("valid color code");
    else
        System.out.println("invalid color code");
}
public static boolean colorCodeValidation(String s1) {
    boolean b=false,b1=false;
    String s2=s1.substring(1,s1.length());
    if(s1.length()==7)
        if(s1.charAt(0)=='#')
            b1=true;

    if(b1==true)
        for(int i=0;i<s2.length();i++){
            char c=s2.charAt(i);
            if(c!='#'){
                if((Character.isAlphabetic(c)&& Character.isUpperCase(c)) || Character.isDigit(c))
                    b=true;
                else{
                    b=false;
                    break;}}}
    return b;
}
}
-----||
```

44) Find the Maximum span of the given array.

span is the number of elements between the duplicate element including those 2 repeated numbers.

if the array has only one element, then the span is 1.

```

input[]={1,2,1,1,3}
output1=4
input[]={1,2,3,4,1,1,5}
output1=6
```

```

package Set3;
public class ClassSeT45 {
public static void main(String[] args) {
    int[]a={1,2,1,1,3};
    System.out.println(maxSpan(a));
}
public static int maxSpan(int[] a) {
    String s2 = null;
    int n=0;
    StringBuffer sb=new StringBuffer();
    for(int i=0;i<a.length;i++)
        sb.append(String.valueOf(a[i]));
    String s1=sb.toString();
```

```

        for(int i=0;i<s1.length();i++)
            for(int j=i+1;j<s1.length();j++)
                if(s1.charAt(i)==s1.charAt(j))
                    s2=String.valueOf(s1.charAt(j));
        int n1=s1.indexOf(s2);
        int n2=s1.lastIndexOf(s2);
        for(int i=n1+1;i<n2;i++)
            n++;
        return (n+2);
    }
}

```

-----||  
45) Getting the first and last n letters from a word where wordlength > 2n.

Ex: Input: california,3.

output: calnia.

```

package Set3;
public class ClassSeT46 {
public static void main(String[] args) {
    String s1="california";
    int n1=3;
    System.out.println(subStringOfgivenString(s1,n1));
}
public static String subStringOfgivenString(String s1, int n1) {
    StringBuffer sb=new StringBuffer();
    sb.append(s1.substring(0, n1)).append(s1.substring(s1.length()-n1,s1.length()));
    return sb.toString();
}

```

-----||  
46) input1="aBrd";  
input2="aqrbA";  
input3=2;  
output1=true;  
2nd char of ip1 and last 2nd char of ip2 show be equal

```

package Set3;
public class ClassSeT46 {
public static void main(String[] args) {
    String ip1="aBrd";
    String ip2="aqrbA";
    int ip3=2;
    System.out.println(charCheck(ip1,ip2,ip3));
}
public static boolean charCheck(String ip1, String ip2, int ip3){
    boolean b=false;
    String s1=String.valueOf(ip1.charAt(ip3-1));
    String s2=String.valueOf(ip2.charAt(ip2.length()-ip3));
    if(s1.equalsIgnoreCase(s2))
        b=true;
    return b;
}

```

47) Add elements of digits:9999

output:9+9+9+9=3+6=9;

```
package Set3;
public class ClassSeT47 {
public static void main(String[] args) {
    int n=9999;
    System.out.println(conversiontoaSingleDigit(n));
}
public static int conversiontoaSingleDigit(int n){
    loop:
        while(n>10){
            int l=0,m=0;
            while(n!=0){
                m=n%10;
                l=l+m;
                n=n/10; }
            n=l;
            continue loop; }
    return n;
}
}
```

-----||  
48) leap year or not using API?

```
package Set3;
import java.util.*;
public class ClassSeT48 {
public static void main(String[] args) {
    String s="2013";
    System.out.println(leapYear(s));
}
public static boolean leapYear(String s) {
    int n=Integer.parseInt(s);
    GregorianCalendar c=new GregorianCalendar();
    boolean b=c.isLeapYear(n);
    return b;
}
}
```

-----||  
49) perfect no or not?

```
package Set3;
public class ClassSeT49 {
public static void main(String[] args) {
    int n=28;
    System.out.println(perfectNumber(n));
}
public static boolean perfectNumber(int n) {
    int n1=0;
    boolean b=false;
    for(int i=1;i<n;i++)
        if(n%i==0)
            n1+=i;
}
```

```

//System.out.println(n1);
if(n1==n)
    b=true;
return b;
}
}

-----||

50) HashMap<String, String> input1={"mouse":"100.2","speaker":"500.6","Monitor":"2000.23"};
String[] input2={"speaker","mouse"};
Float output=600.80(500.6+100.2);

package Set3;
import java.util.*;
public class ClassSeT50 {
public static void main(String[] args) {
    HashMap<String, String> m1=new HashMap<String, String>();
    m1.put("mouse", "100.2");
    m1.put("speaker", "500.6");
    m1.put("monitor", "2000.23");
    String[] s={"speaker","mouse"};
    System.out.println(getTheTotalCostOfPheripherals(m1,s));
}
public static float getTheTotalCostOfPheripherals(HashMap<String, String> m1,
    String[] s) {
    Float f=(float) 0;
    Iterator<String> i=m1.keySet().iterator();
    while(i.hasNext()){
        String s1=(String) i.next();
        Float f1=Float.parseFloat(m1.get(s1));
        for(int j=0;j<s.length;j++)
            if(s[j].equals(s1))
                f+=f1;
    }
    return f;
}
}

```

-----||

51) Input1=845.69, output=3:2;  
Input1=20.789; output=2:3;  
Input1=20.0; output=2:1;  
output is in Hashmap format.  
Hint: count the no of digits.

```

package Set3;
import java.util.*;
public class ClassSeT51 {
public static void main(String[] args) {
    double d=845.69;
    System.out.println(noOfDigits(d));
}
public static String noOfDigits(double d) {
    int n1=0,n2=0;
    String s=String.valueOf(d);
    StringTokenizer t=new StringTokenizer(s,".");
    String s1=t.nextToken();

```

```

String s2=t.nextToken();
n1=s1.length();
n2=s2.length();
if(s1.charAt(0)=='0')
    n1=s1.length()-1;
if(n2!=1)
    if(s2.charAt(s2.length()-1)=='0')
        n2=s2.length()-1;
String s3=String.valueOf(n1)+":"+String.valueOf(n2);
return s3;
}
}

```

-----||  
-----||  
-----||  
-----||  
-----||

### SET 2

- 1) compare two strings, if the characters in string 1 are present in string 2, then it should be put as such in output, else '+' should be put in output...ignore case difference.

input 1:"New York"  
input 2:"NWYR"

output:N+w+Y+r+

```

package Set2;
public class ClassSET1 {
public static void main(String[] args) {
    String s1="New York";
    String s2="NWYR";
    System.out.println(StringFormatting(s1,s2));
}
public static String StringFormatting(String s1, String s2) {
    StringBuffer s4=new StringBuffer();
    String s3=s1.toUpperCase();
    for(int i=0;i<s2.length();i++)
        for(int j=0;j<s3.length();j++)
            if(s2.charAt(i)==s3.charAt(j))
                s4.append(s1.charAt(j)).append('+');
    return s4.toString();
}
}

```

2) input:

||

```

Searchstring s1="GeniusRajkumarDev";
String s2="Raj";
String s3="Dev";
output:
Return 1 if s2 comes before s3 in searchstring else return 2

```

```

package Set2;
public class ClassSET2 {
public static void main(String[] args) {
    String srch="MaanVeerSinghKhurana";
    String s1="Veer";
    String s2="Singh";
    int n=searchString(srch,s1,s2);
    if(n==1)
        System.out.println(s1+" comes before "+s2);
    else
        System.out.println(s2+" comes before "+s1);
}
public static int searchString(String srch, String s1, String s2) {
    int n=0;
    int n1=srch.indexOf(s1);
    int n2=srch.indexOf(s2);
    if(n1<n2)
        n=1;
    else
        n=2;
    return n;
}
}
-----||
```

3) months between two dates

```

package Set2;
import java.text.*;
import java.util.*;
public class ClassSET3 {
public static void main(String[] args) throws ParseException {
    String s1="30/05/2013";
    String s2="01/06/2013";
    System.out.println(monthsBetweenDates(s1,s2));
}
public static int monthsBetweenDates(String s1, String s2) throws ParseException {
    SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
    Date d1=sdf.parse(s1);
    Date d2=sdf.parse(s2);
    Calendar cal=Calendar.getInstance();
    cal.setTime(d1);
    int months1=cal.get(Calendar.MONTH);
    int year1=cal.get(Calendar.YEAR);

```

```
    cal.setTime(d2);
    int months2=cal.get(Calendar.MONTH);
    int year2=cal.get(Calendar.YEAR);
    int n=((year2-year1)*12)+(months2-months1);
    return n;
}
}
```

-----||  
4) input="xyzwabcd"

```
input2=2;
input3=4;
output=bawz
```

```
package Set2;
public class ClassSET4 {
public static void main(String[] args) {
    String s1="xyzwabcd";
    int n1=2,n2=4;
    System.out.println(retrievalOfString(s1,n1,n2));
}
```

```
public static String retrievalOfString(String s1, int n1, int n2) {
    StringBuffer sb=new StringBuffer(s1);
    sb.reverse();
    String s2=sb.substring(n1, n1+n2);
    return s2;
}
}
```

-----||  
5) Given integer array

```
input :int[] arr={2,3,5,4,1,6,7,7,9};
```

Remove the duplicate elements and print sum of even numbers in the array..

print -1 if arr contains only odd numbers

```
package Set2;
import java.util.*;
public class ClassSET5 {
public static void main(String[] args) {
    int a[]={2,3,5,4,1,6,7,7,9};
    System.out.println(sumOfEvenNos(a));
}
public static int sumOfEvenNos(int[] a) {
    List<Integer> l1=new ArrayList<Integer>();
    for(int i=0;i<a.length;i++)
```

```

        l1.add(a[i]);
    List<Integer> l2=new ArrayList<Integer>();
    for(int i=0;i<a.length;i++)
        for(int j=i+1;j<a.length;j++)
            if(a[i]==a[j])
                l2.add(a[j]);
    l1.removeAll(l2);
    l1.addAll(l2);
    int n=0,n1;
    for(int i=0;i<l1.size();i++)
        if(l1.get(i)%2==0)
            n+=l1.get(i);
    if(n==0)
        n1=-1;
    else
        n1=n;
    return n1;
}
}
-----||
```

- 6) input1="abc2012345"  
 input2="abc2112660"  
 input 3=4  
 here "abc\*\*" refers to customer id.  
 12345 refers to last month eb reading and 12660 refers to this month eb reading  
 find the difference between two readings and multiply it by input3

ie., output=(12660-12345)\*4

```

package Set2;
public class ClassSET6 {
    public static void main(String[] args) {
        String input1="abc2012345";
        String input2="abc2112660";
        int input3=4;
        System.out.println(meterReading(input1,input2,input3));
    }
    public static int meterReading(String input1, String input2, int input3) {
        int n1=Integer.parseInt(input1.substring(5, input1.length()));
        int n2=Integer.parseInt(input2.substring(5, input2.length()));
        int n=Math.abs((n2-n1)*input3);
        return n;
    }
}
-----||
```

- 7) Given array of integers, print the sum of elements squaring/cubing as per the power of their indices.  
 //answer= sum=sum+a[i]^i;  
 eg:input:{2,3,5}  
 Output:29

```

package Set2;
public class ClassSET7 {
public static void main(String[] args) {
    int a[]={2,3,5};
    System.out.println(sumOfElementsWrtIndices(a));
}
public static int sumOfElementsWrtIndices(int[] a) {
    int s=0;
    for(int i=0;i<a.length;i++)
        s+=(Math.pow(a[i], i));
    return s;
}
}
-----||
```

8) Given array of string check whether all the elements contains only digits not any alphabets.

if condn satisfied print 1 else -1.

Input:{ "123", "23.14", "522" }

Output:1

Input1:{ "asd", "123", "42.20" }

Output:-1

```

package Set2;
public class ClassSET8 {
public static void main(String[] args) {
    String[] input1={"123", "23.14", "522"};
    //String[] input1={"asd", "123", "42.20"};
    System.out.println(stringOfDigits(input1));
}
public static int stringOfDigits(String[] input1) {
    int n=0;
    for(int i=0;i<input1.length;i++){
        String s1=input1[i];
        for(int j=0;j<s1.length();j++){
            char c1=s1.charAt(j);
            if(Character.isDigit(c1))
                n=1;
            else {n=-1;break;}}
    }
    return n;
}
}
-----||
```

9) int[] a={12,14,2,26,35}

find the difference b/w max and min values in array

Output:35-2=33.

```

package Set2;
import java.util.Arrays;
public class ClassSET9 {
public static void main(String[] args) {
    int a[]={12,14,2,26,35};
    System.out.println(diffBwMaxAndMin(a));
}
public static int diffBwMaxAndMin(int[] a) {
    Arrays.sort(a);
    int n=a[a.length-1]-a[0];
    return n;
}
}
-----||
```

10) Given an array find the largest 'span'(span is the number of elements between two same digits)  
find their sum.

```
a[]={1,4,2,1,4,1,5}
Largest span=5
```

```

package Set2;
public class ClassSET10 {
public static void main(String[] args) {
    int a[]={1,4,2,1,4,1,5};
    System.out.println("sum of largest span elements:"+largestSpan(a));
}
public static int largestSpan(int[] a) {
    int max=0;
    int p1=0;
    int p2=0;
    int n=0;
    int sum=0;
    for(int i=0;i<a.length-1;i++){
        for(int j=i+1;j<a.length;j++)
            if(a[i]==a[j])
                n=j;
        if(n-i>max){
            max=n-i;
            p1=i;
            p2=n; }
    }
    System.out.println("largest span:"+ (p2-p1));
    for(int i=p1;i<=p2;i++)
        sum=sum+a[i];
    return (sum);
}
```

-----||  
11) input={"1","2","3","4"}  
    output=10

ie,if string elements are nos,add it.

    input={"a","b"}  
    output=-1;

```
package Set2;  
public class ClassSET11 {  
public static void main(String[] args) {  
    String s[]={ "1","2","3","4"};  
    //String s[]={ "a","b","3","4"};  
    System.out.println(checkForStringElements(s));  
}  
public static int checkForStringElements(String[] s) {  
    int n=0;  
    boolean b=false;  
    for(int i=0;i<s.length;i++){  
        String s1=s[i];  
        for(int j=0;j<s1.length();j++){  
            char c=s1.charAt(j);  
            if(Character.isDigit(c))  
                b=true;  
            else{  
                b=false;  
                break; }  
        if(b==true)  
            n+=Integer.parseInt(s1);  
        else{  
            n=-1;  
            break; }  
    return n;  
}  
}
```

-----||  
12) arraylist1={ 1,2,3,4,5}

    arraylist2={ 6,7,8,9,10}

size of both list should be same

output={6,2,8,4,10}

```
package Set2;  
public class ClassSET12 {  
public static void main(String[] args) {  
    int a[]={1,2,3,4,5};  
    int b[]={6,7,8,9,10};  
    int c[] = alternativeIndicesElements(a,b);
```

```

        for(int d:c)
            System.out.println(d);
    }
    public static int[] alternativeIndicesElements(int[] a, int[] b){
        int c[]=new int[a.length];
        if(a.length==b.length)
            for(int i=0;i<a.length;i++)
                if(i%2==0)
                    c[i]=b[i];
                else
                    c[i]=a[i];
        return c;
    }
}

```

13) count the number of times the second word in second string occurs in first string-case sensitive ||

input1=hai hello hai where hai Hai;  
 input2=what hai  
 output=3;

```

package Set2;
import java.util.StringTokenizer;
public class ClassSET13 {
    public static void main(String[] args) {
        String input1="hai hello how are you?? hai hai";
        String input2="what hai";
        System.out.println(stringOccurance(input1,input2));
    }
}

```

```

public static int stringOccurance(String input1, String input2){
    int count=0;
    StringTokenizer t1=new StringTokenizer(input2, " ");
    String s1=t1.nextToken();
    String s2=t1.nextToken();
    StringTokenizer t2=new StringTokenizer(input1, " ");
    while(t2.hasMoreTokens()){
        if(t2.nextToken().equals(s2))
            count++;
    }
    return count;
}
}

```

14) find the no.of characters,that has repeated 3 consecutive times ||

input1:"aaabbccc"  
 ouput1=2;

```

package Set2;
public class ClassSET14 {
    public static void main(String[] args) {
        String input1="aaabbccc";
        System.out.println(consecutiveRepeationOfChar(input1));
    }

    public static int consecutiveRepeationOfChar(String input1) {
        int c=0;
        int n=0;
        for(int i=0;i<input1.length()-1;i++){
            if(input1.charAt(i)==input1.charAt(i+1))
                n++;
            else
                n=0;

            if(n==2)
                c++;
        }
        return c;
    }
}
-----||
```

15)What will be the DAY of current date in next year

```

package Set2;
import java.text.SimpleDateFormat;
import java.util.*;
public class ClassSet15 {
    public static void main(String[] args) {
        Date d=new Date();
        System.out.println(sameDayOnUpcomingYear(d));
    }

    public static String sameDayOnUpcomingYear(Date d) {
        Date d1=new Date();
        d1.setYear(d.getYear()+1);
        SimpleDateFormat sdf=new SimpleDateFormat("EEEE");
        String s=sdf.format(d1);
        return s;
    }
}
-----||
```

16)Get all the numbers alone from the string and return the sum.

Input"123gif"

Output:6

```

package Set2;
public class ClassSET16 {
public static void main(String[] args) {
    String s="1234gif";
    System.out.println(summationOfNosInaString(s));
}
public static int summationOfNosInaString(String s) {
    int n=0;
    for(int i=0;i<s.length();i++){
        char c=s.charAt(i);
        if(Character.isDigit(c)){
            String s1=String.valueOf(c);
            n+=Integer.parseInt(s1); } }
    return n;
}
}
-----||
```

17)input array={red,green,blue,ivory}  
sort the given array  
reverse the given array  
if user input is 1 it should give oth element of an reversed array.

```

package Set2;
import java.util.*;
public class ClassSET17 {
public static void main(String[] args) {
    String[] s={"red","green","blue","ivory","yellow"};
    int n=1;
    System.out.println(retrievingRequiredColor(s,n));
}
public static String retrievingRequiredColor(String[] s, int n) {
    String s1=new String();
    List<String> l=new ArrayList<String>();
    for(int i=0;i<s.length;i++)
        l.add(s[i]);
    Collections.sort(l,Collections.reverseOrder());
    for(int i=0;i<l.size();i++)
        if(i==(n-1))
            s1=l.get(i);
    return s1;
}
}
-----||
```

18)String a="a very fine day"  
output:A Very Fine Day

```

package Set2;
import java.util.StringTokenizer;
public class ClassSET18 {
public static void main(String[] args){
    String s1="its a very fine day";
    System.out.println(capsStart(s1));
}
public static String capsStart(String s1){
    StringBuffer s5=new StringBuffer();
    StringTokenizer t=new StringTokenizer(s1, " ");
    while(t.hasMoreTokens()){
        String s2=t.nextToken();
        String s3=s2.substring(0,1);
        String s4=s2.substring(1, s2.length());
        s5.append(s3.toUpperCase()).append(s4).append(" ");
    }
    return s5.toString();
}
}
-----||
```

19)Take the word with a max length in the given sentence  
in that check for vowels if so count the no.of occurrences !

Input 1="Bhavane is a good girl"

Output =3

Input 1="Bhavanee is a good girl"

Output =4

```

package Set2;
import java.util.StringTokenizer;
public class ClassSET19 {
public static void main(String[] args) {
    String s1="Bhavane is a good girl";
    System.out.println(countVowelsInMaxLengthString(s1));
}
public static int countVowelsInMaxLengthString(String s1) {
    int n1=0, max=0;
    String s4="AEIOUaeiou";
    String s3=new String();
    StringTokenizer t=new StringTokenizer(s1, " ");
    while(t.hasMoreTokens()){
        String s2=t.nextToken();
        int n2=s2.length();
        if(n2>max){
            max=n2;
            s3=s2; } }
    for(int i=0;i<s3.length();i++)
        for(int j=0;j<s4.length();j++)
            if(s3.charAt(i)==s4.charAt(j))
                n1++;
    return n1;
}
```

}

-----||  
20) for the given string display the middle 2 value, case satisfies only if string is of even length

ip - java

op - av

```
package Set2;
public class ClassSET20 {
public static void main(String[] args) {
    String s1="sumithra";
    System.out.println(printingSubstringOfMiddleChars(s1));
}
public static String printingSubstringOfMiddleChars(String s1) {
    String s2=s1.substring((s1.length()/2)-1, (s1.length()/2)+1);
    return s2;
}
```

-----||  
21) Given an array int a[]. Add the sum at even indexes.do the same with odd indexes.

if both the sum is equal return 1 or return -1.

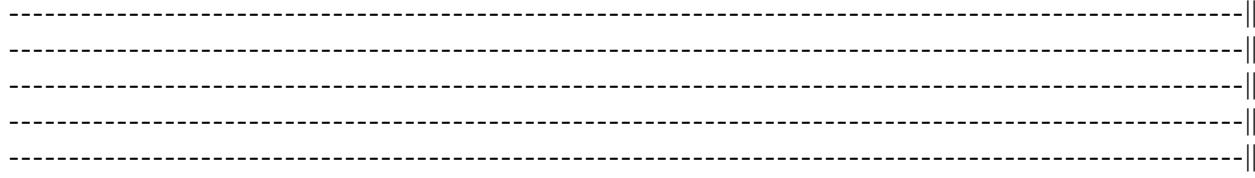
```
package Set2;
public class ClassSET21 {
public static void main(String[] args) {
    int a[]={9,8,5,3,2,6,4,7,5,1};
    System.out.println(oddEvenIndicesCount(a));
}
public static int oddEvenIndicesCount(int[] a) {
    int n1=0,n2=0,n3=0;
    for(int i=0;i<a.length;i++)
        if(i%2==0)
            n1+=a[i];
        else
            n2+=a[i];
    if(n1==n2)
        n3=1;
    else
        n3=-1;
    return n3;
}
```

-----||  
22)no of days in a month in specific year

```

package Set2;
import java.util.*;
public class ClassSET22 {
public static void main(String[] args){
    Calendar ca=new GregorianCalendar(2013,Calendar.FEBRUARY,03);
    System.out.println(noOfDaysInaMonth(ca));
}
public static int noOfDaysInaMonth(Calendar ca){
    int n=ca.getActualMaximum(Calendar.DAY_OF_MONTH);
    return n;
}
}

```



-----SET 1-----

1. A number is given as input. Find the odd digits in the number, add them and find if the sum is odd or not.if even return -1, if odd return 1

input:52315  
logic:5+3+1+5=14(even)  
output:-1  
input:1112  
logic:1+1+1=3(odd)  
output:1

package Set1;

```

public class ClassSet1 {
    public static int SumOfOddsAndEvens(int n){
        int n1,n2=0,n3;
        while(n!=0)
        {
            n1=n%10;
            if((n1%2)!=0)
                n2+=n1;
            n/=10;
        }
        if(n2%2==0)
            n3=-1;
        else
            n3=1;
        return n3;
    }
}

```

```
public static void main(String[] args) {  
    int n=12346;  
    System.out.println(SumOfOddsAndEvens(n));  
}  
}
```

---

---

2.Find the day(Friday) of a date input given as MM-dd-yyyy (format)

input:12-27-2012

output:Thursday

```
package Set1;  
import java.util.*;  
import java.text.*;  
public class ClassSet2 {  
    public static String getDay(Date d1){  
        String s1;  
        SimpleDateFormat sdf=new SimpleDateFormat("EEEE");  
        s1=sdf.format(d1);  
        return s1;  
    }  
    public static void main(String[] args) {  
        Date d1=new Date(2012/12/27);  
        System.out.println("day is:"+getDay(d1));  
    }  
}
```

---

---

3.A integer array is given as input. find the difference between each element. Return the index of the largest element which has the largest difference gap.

input: {2,3,4,2,3}

logic: 2-3=1,3-4=1,4-2=2,2-3=1

2 is the max diff between 4 and 2,return the index of 4(2)

output:2

```
package Set1;
```

```
public class ClassSet3 {  
    public static int getDiffArray(int[] n1){  
        int n2,n3=0,n4=0,i;  
        for(i=0;i<n1.length-1;i++){  
            n2=Math.abs(n1[i]-n1[i+1]);  
            if(n2>n3){  
                n3=n2;  
                n4=i+1; }}  
        return n4;
```

```
}

public static void main(String[] args) {
    int[] n1={2,9,4,7,6};
    System.out.println(getDiffArray(n1));
}

-----
```

4.Given two integer arrays. merge the common elements into a new array. find the sum of the elements

input1:{1,2,3,4}

input2:{3,4,5,6}

logic:{3,4}

output:7

```
package Set1;
import java.util.*;
public class ClassSet4 {
    public static int mergeArray(int a[],int b[]){
        List<Integer> l1=new ArrayList<Integer>();
        List<Integer> l2=new ArrayList<Integer>();
        int i,d=0;
        for(i=0;i<a.length;i++)
            l1.add(a[i]);
        for(i=0;i<b.length;i++)
            l2.add(b[i]);
        l1.retainAll(l2);
        for(i=0;i<l1.size();i++)
            d+=(Integer) l1.get(i);
        return d;
    }
    public static void main(String[] args) {
        int[] a={1,2,3,4};
        int[] b={3,4,5,6};
        System.out.println(mergeArray(a,b));
    }
}
```

5.Given two arrayslist. retrieve the odd position elements form first input and even position elemets from second list.  
put it into the new arraylist at the same positions from where they are retrieved from.

(consider 0th position as even position, and two lists are of same size)

input1:{12,1,32,3}

input2:{0,12,2,23}

output:{0,1,2,3}

```

package Set1;

public class ClassSet5 {
    public static int[] retrievePosition(int[] a,int[] b){
        int[] c=new int[a.length];
        int i;
        for(i=0;i<a.length;i++){
            if(i%2==0)
                c[i]=b[i];
            if(i%2!=0)
                c[i]=a[i]; }
        return c;
    }
    public static void main(String[] args) {
        int[] a={12,1,32,3};
        int[] b={0,12,2,23};
        int[] c=retrievePosition(a,b);
        for(int d:c)
            System.out.println(d);
    }
}

```

---



---



---

6. Sum of fibonaaci series upto given input.

input:3  
 logic:1+1+2  
 output:4

```

package Set1;
import java.util.*;
public class ClassSet6 {
    public static int sumOfFibonacci(int n){
        int a=-1,b=1,c=0,d=0;
        for(int i=0;i<=n;i++){
            c=a+b;
            a=b; b=c;
            d+=c;
        }
        return d;
    }
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        System.out.println(sumOfFibonacci(n));
    }
}

```

---



---

7. Retrieve the odd numbers till given input number. add and subtract it consecutively and return the result.

Input:9

Output:1+3-5+7-9=-3

```
package Set1;
import java.util.*;
public class ClassSet7 {
    public static int consecutiveSumSubofOddNos(int n){
        List<Integer> l1=new ArrayList<Integer>();
        for(int i=1;i<=n;i++)
            if(i%2!=0)
                l1.add(i);
        int n1=l1.get(0);
        for(int i=1;i<l1.size();i++)
            if(i%2!=0)
                n1=n1+l1.get(i);
            else
                n1=n1-l1.get(i);
        return n1;
    }
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        System.out.println(consecutiveSumSubofOddNos(n));
    }
}
```

8. Given date in dd-MM-yyyy format.return the month in full name format(January)

input:"23-01-2012"

output:January

```
package Set1;
import java.text.SimpleDateFormat;
import java.util.*;
public class ClassSet8 {
    public static String monthDiff(Date d1){
        SimpleDateFormat sdf=new SimpleDateFormat("MMMM");
        String s=(sdf.format(d1));
        return s;
    }
    public static void main(String[] args) {
        Date d1=new Date(23/01/2012);
        System.out.println(monthDiff(d1));
    }
}
```

9. Two dates are given as input in "yyyy-MM-dd" format. Find the number of months between the two dates  
input1:"2012-12-01"  
input2:"2012-01-03"  
output:11

```
package Set1;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
public class ClassSet9 {
    public static int monthDiff(String s1, String s2) throws ParseException{
        SimpleDateFormat sdf=new SimpleDateFormat("yyyy-MM-dd");
        Calendar c=Calendar.getInstance();
        Date d1=sdf.parse(s1);
        Date d2=sdf.parse(s2);
        c.setTime(d1);
        int m1=c.get(Calendar.MONTH);
        int y1=c.get(Calendar.YEAR);
        c.setTime(d2);
        int m2=c.get(Calendar.MONTH);
        int y2=c.get(Calendar.YEAR);
        int n=(y1-y2)*12+(m1-m2);
        return n;
    }
    public static void main(String[] args) throws ParseException {
        String s1=new String("2013-12-01");
        String s2=new String("2012-01-03");
        System.out.println(monthDiff(s1,s2));
    }
}
```

10. arraylist of string type which has name#mark1#mark2#mark3 format. retrieve the name of the student who has scored max marks(total of three)  
input:{ "arun#12#12#12", "deepak#13#12#12" }  
output:deepak

```
package Set1;
import java.util.*;
public class ClassSet10 {
    public static String retrieveMaxScoredStudent(String[] s1){
        Map<String, Integer> m1=new HashMap<String, Integer>();
```

```

for(int i=0;i<s1.length;i++){
    String s2=s1[i];
    StringTokenizer t=new StringTokenizer(s2,"#");
    String s3=t.nextToken();
    int n1=Integer.parseInt(t.nextToken());
    int n2=Integer.parseInt(t.nextToken());
    int n3=Integer.parseInt(t.nextToken());
    int n=n1+n2+n3;
    m1.put(s3, n);
}
//System.out.println(m1);
int max=0;
String m=new String();
Iterator<String> i=m1.keySet().iterator();
while(i.hasNext()){
    String s4=i.next();
    int j=m1.get(s4);
    if(j>max){
        max=j;
        m=s4; }
}
return m;
}
public static void main(String[] args) {
    String[] s1={"arun#12#12#12","deepak#13#12#12","puppy#12#11#12"};
    System.out.println(retrieveMaxScoredStudent(s1));
}
}

```

---



---

11.Two inputs of a string array and a string is received. The array shld be sorted in descending order. The index of second input in a array shld be retrieved.

input1:{"ga","yb","awe"}  
input2:awe  
output:2

```

package Set1;
import java.util.*;
public class ClassSet11 {
    public static void main(String[] args) {
        String[] s1={"good","yMe","awe"};
        String s2="awe";
        System.out.println(stringRetrieval(s1,s2));
    }
    public static int stringRetrieval(String[] s1, String s2){
        ArrayList<String> l1=new ArrayList<String>();
        for(int i=0;i<s1.length;i++)
            l1.add(s1[i]);
        Collections.sort(l1,Collections.reverseOrder());
        System.out.println(l1);
        int n=l1.indexOf(s2);
    }
}

```

```
        return n;
    }
}
```

---

12.A time input is received as stirng. Find if the hour format is in 12 hour format. the suffix am/pm is case insensitive.  
input:"09:36 am"  
output:true

```
package Set1;
import java.text.*;
import java.util.*;
public class ClassSet12 {
    public static boolean hourFormat(String s) throws ParseException{
        boolean b=false;
        StringTokenizer st=new StringTokenizer(s, " ");
        String s1=st.nextToken();
        String s2=st.nextToken();
        StringTokenizer st1=new StringTokenizer(s1,":");
        int n1=Integer.parseInt(st1.nextToken());
        int n2=Integer.parseInt(st1.nextToken());
        if((s2.equalsIgnoreCase("am"))|| (s2.equalsIgnoreCase("pm")))
            if((n1<=12)&&(n2<=59))
                b=true;
        return b;
    }
    public static void main(String[] args) throws ParseException {
        String s="19:36 am";
        boolean b=hourFormat(s);
        if(b==true)
            System.out.println("the time is in 12 hr format");
        else
            System.out.println("the time is in 24 hr format");
    }
}
```

---

13.Retrieve the palindorme-true number set from given number limit and return the sum  
input1:90 input2:120  
logic:99+101+111  
output:311

```
package Set1;
import java.util.*;
public class ClassSet13 {
    public static int sumOfPalindromeNos(int n1,int n2){
        List<Integer> l1=new ArrayList<Integer>();
```

```

for(int i=n1;i<=n2;i++){
    int r=0,n3=i;
    while(n3!=0){
        r=(r*10)+(n3%10);
        n3=n3/10;    }
    if(r==i)
        l1.add(i);      }
System.out.println(l1);
int s=0;
for(int i=0;i<l1.size();i++)
    s+=l1.get(i);
return s;
}
public static void main(String[] args) {
    Scanner s=new Scanner(System.in);
    System.out.println("enter the range:");
    int n1=s.nextInt();
    int n2=s.nextInt();
    System.out.println("sum of palindrome nos.within given range is:"+sumOfPalindromeNos(n1,n2));
}
}

```

---



---



---

14. find the max length-word in a given string and return the max-length word. if two words are of same length return the first occurring word of max-length

input: "hello how are you aaaaa"

output: hello(length-5)

```

package Set1;
import java.util.*;
public class ClassSet14 {
    public static String lengthiestString(String s1){
        int max=0;
        String s2=null;
        StringTokenizer t=new StringTokenizer(s1, " ");
        while(t.hasMoreTokens()){
            s1=t.nextToken();
            int n=s1.length();
            if(n>max){
                max=n;
                s2=s1; }
        }
        return s2;
    }
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        System.out.println("enter the String:");
        String s1=s.nextLine();
        System.out.println("the lengthiest string is:"+lengthiestString(s1));
    }
}

```

```
}
```

---

---

---

15. Get a input string. reverse it and parse it with '-'.

input:computer

output:r-e-t-u-p-m-o-c

```
package Set1;
import java.util.Scanner;
public class ClassSet15 {
    public static String reversedAndParsedString(String s1){
        StringBuffer sb=new StringBuffer(s1);
        sb.reverse();
        StringBuffer sb1=new StringBuffer();
        for(int i=0;i<(2*s1.length()-1;i++)
            if(i%2!=0)
                sb1=sb.insert(i, '-');
        return sb1.toString();
    }
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        System.out.println("enter the String:");
        String s1=s.next();
        System.out.println("the formatted string is:"+reversedAndParsedString(s1));
    }
}
```

---

---

16. HashMap with regnum as key and mark as value. find the avg of marks whose key is odd.

input:{ 12:90,35:90,33:90,56:88 }

output:avg of(90+90) =90

```
package Set1;
import java.util.*;
public class ClassSet16 {
    public static int averageOfMarks(Map<Integer,Integer> m1){
        int n=0,c=0;
        Iterator<Integer> i=m1.keySet().iterator();
        while(i.hasNext()){
            Integer b=i.next();
            if(b%2!=0){
                c++;
                n+=m1.get(b); } }
        return (n/c);
    }
    public static void main(String[] args) {
        Map<Integer,Integer> m1=new HashMap<Integer,Integer>();
```

```

        m1.put(367, 89);
        m1.put(368, 98);
        m1.put(369, 92);
        m1.put(366, 74);
        m1.put(365, 67);
        System.out.println(averageOfMarks(m1));
    }
}

```

---



---



---

17. A integer input is accepted. find the square of individual digits and find their sum.

input:125

output:1\*1+2\*2+5\*5

```

package Set1;
import java.util.*;
public class ClassSet16 {
    public static int averageOfMarks(Map<Integer,Integer> m1){
        int n=0,c=0;
        Iterator<Integer> i=m1.keySet().iterator();
        while(i.hasNext()){
            Integer b=i.next();
            if(b%2!=0){
                c++;
                n+=m1.get(b); } }
        return (n/c);
    }
    public static void main(String[] args) {
        Map<Integer,Integer> m1=new HashMap<Integer,Integer>();
        m1.put(367, 89);
        m1.put(368, 98);
        m1.put(369, 92);
        m1.put(366, 74);
        m1.put(365, 67);
        System.out.println(averageOfMarks(m1));
    }
}

```

---



---

18.Two input strings are accepted. If the two strings are of same length concat them and return, if they are not of same length, reduce the longer string to size of smaller one, and concat them

input1:"hello"

input2:"hi"

output:"lohi"

input1:"aaa"

input2:"bbb"

output:"aaabbb"

```

package Set1;
import java.util.*;
public class ClassSet18 {
    public static String concatStrings(String s1, String s2){
        StringBuffer sb=new StringBuffer();
        if(s1.length()==s2.length())
            sb.append(s1).append(s2);
        else if(s1.length()>s2.length()){
            s1=s1.substring(s1.length()/2, s1.length());
            sb.append(s1).append(s2);
        }
        else if(s1.length()<s2.length()){
            s2=s2.substring(s2.length()/2, s2.length());
            sb.append(s1).append(s2);
        }
        return sb.toString();
    }
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.next();
        String s2=s.next();
        System.out.println(concatStrings(s1,s2));
    }
}

```

---



---



---

19.accept a string and find if it is of date format "dd/mm/yyyy".

input:01/13/2012

output:false

```

package Set1;
import java.util.StringTokenizer;
public class ClassSet19 {
    public static boolean formatDate(String s){
        boolean b1=false;
        StringTokenizer t=new StringTokenizer(s,"/");
        int n1=Integer.parseInt(t.nextToken());
        int n2=Integer.parseInt(t.nextToken());
        String s2=t.nextToken();
        int n3=Integer.parseInt(s2);
        int n4=s2.length();
        if(n4==4)
        {
            if(n3%4==0)
            {
                if((n2==2)&&(n1<=29))
                    b1=true;
            }
        }
    }
}

```

```

        else if((n2==4)||(n2==6)||(n2==9)||(n2==11)&&(n1<=30))
            b1=true;
        else if((n2==1)||(n2==3)||(n2==5)||(n2==7)||(n2==8)||(n2==10)||(n2==12)&&(n1<=31))
            b1=true;
    }
    else
    {
        if((n2==2)&&(n1<=28))
            b1=true;
        else if((n2==4)||(n2==6)||(n2==9)||(n2==11)&&(n1<=30))
            b1=true;
        else if((n2==1)||(n2==3)||(n2==5)||(n2==7)||(n2==8)||(n2==10)||(n2==12)&&(n1<=31))
            b1=true;
    }
}
return b1;
}
public static void main(String[] args) {
String s="31/5/2012";
boolean b=formattingDate(s);
if(b==true)
    System.out.println("date is in dd/MM/yyyy format");
else
    System.out.println("date is not in dd/MM/yyyy format");
}

```

---



---



---

20. Get a integer array as input. Find the average of the elements which are in the position of prime index  
input:{1,2,3,4,5,6,7,8,9,10}  
logic:3+4+6+8(the indeces are prime numbers)  
output:21

```

package Set1;
import java.util.*;
public class ClassSet20 {
    public static int sumOfPrimeIndices(int[] a,int n){
        int n1=0;
        for(int i=2;i<n;i++){
            int k=0;
            for(int j=2;j<i;j++)
                if(i%j==0)
                    k++;
            if(k==0)
                n1+=a[i];
        }
        return n1;
    }
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        System.out.println("enter the array limit:");

```

```
int n=s.nextInt();
System.out.println("enter the array elements:");
int[] a=new int[n];
for(int i=0;i<n;i++)
    a[i]=s.nextInt();
System.out.println(sumOfPrimeIndices(a,n));
}
}
```

---

21.Find the extension of a given string file.

input: "hello.jpeg"  
output: "jpeg"

```
package Set1;
import java.util.*;
public class ClassSet21 {
    public static String extensionString(String s1){
        StringTokenizer t=new StringTokenizer(s1,".");
        t.nextToken();
        String s2=t.nextToken();
        return s2;
    }
    public static void main(String[] args) {
        String s1="hello.jpeg";
        System.out.println(extensionString(s1));
    }
}
```

---

22.Get two integer arrays as input. Find if there are common elements in the arrays. find the number of common elements.

input1: {1,2,3,4}  
input2: {3,4,5,6}  
output: 4(3,4,3,4)

```
package Set1;
public class ClassSet22 {
    public static int commonElements(int[] a,int[] b){
        int c=0;
        for(int i=0;i<a.length;i++)
            for(int j=0;j<b.length;j++)
                if(a[i]==b[j])
                    c++;
        return (2*c);
    }
}
```

```
public static void main(String[] args){  
    int a[]={1,2,3,4,5};  
    int b[]={3,4,5,6,7};  
    System.out.println(commonElements(a,b));  
}  
}
```

---

23. Find if a given pattern appears in both the input strings at same positions.

input1: "hh--ww--"

input2: "rt--er--"

output: true(false otherwise)

```
package Set1;  
public class ClassSet23 {  
    public static boolean stringPattern(String s1,String s2){  
        String st1=s1.length()<s2.length()?s1:s2;  
        String st2=s1.length()>s2.length()?s1:s2;  
        boolean b=true;  
        String s=st2.substring(st1.length());  
        if(s.contains("-"))  
            b=false;  
        else{  
            loop:  
            for(int i=0;i<st1.length();i++)  
                if((st1.charAt(i)=='-') || (st2.charAt(i)=='-'))  
                    if(st1.charAt(i)!=st2.charAt(i)){  
                        b=false;  
                        break loop; }  
            }  
            return b;  
    }  
    public static void main(String[] args) {  
        String s1="he--ll--";  
        String s2="we--ll--";  
        boolean b=stringPattern(s1,s2);  
        if(b==true)  
            System.out.println("same pattern");  
        else  
            System.out.println("different pattern");  
    }  
}
```

---

24. Input a int array. Square the elements in even position and cube the elements in odd position and add them as result.

input: {1,2,3,4}

output:  $1^3+2^2+3^3+4^2$

```

package Set1;
public class ClassSet24 {
public static int squaringAndCubingOfElements(int[] a){
    int n1=0,n2=0;
    for(int i=0;i<a.length;i++)
        if(i%2==0)
            n1+=(a[i]*a[i]*a[i]);
        else
            n2+=(a[i]*a[i]);
    return (n1+n2);
}
public static void main(String[] args) {
    int a[]={1,2,3,4,5,6};
    System.out.println(squaringAndCubingOfElements(a));
}

```

---



---



---

25. Check whether a given string is palindrome also check whether it has atleast 2 different vowels

input: "madam"

output: false(no 2 diff vowels)

```

package Set1;
public class ClassSet25 {
    public static boolean palindromeAndVowelCheck(String s){
        boolean b=true;
        int c=0;
        String s1="aeiou";
        StringBuffer sb=new StringBuffer(s);
        String s2=sb.reverse().toString();
        if(!s2.equals(s))
            b=false;
        else{
            loop:
            for(int i=0;i<s1.length();i++)
                for(int j=0;j<s.length();j++)
                    if(s1.charAt(i)==s.charAt(j)){
                        c++;
                        continue loop; }
            if(c<2)
                b=false;
        }
        return b;
    }
    public static void main(String[] args) {
        String s="deed";

```

```
        System.out.println(palindromeAndVowelCheck(s));
    }
}
```

---

26. Find no of characters in a given string which are not repeated.

input: "hello"

output: 3

```
package Set1;
public class ClassSet26 {
public static int noOfnonRepeatedCharacters(String s1){
    StringBuffer sb=new StringBuffer(s1);
    for(int i=0;i<sb.length();i++){
        int n=0;
        for(int j=i+1;j<sb.length();j++){
            if(sb.charAt(i)==sb.charAt(j)){
                sb.deleteCharAt(j);
                n++;
            }
            if(n>0){
                sb.deleteCharAt(i);
                i--;
            }
        }
        return sb.length();
    }
    public static void main(String[] args) {
        String s1="preethi";
        System.out.println(noOfnonRepeatedCharacters(s1));
    }
}
```

---

27. Get a input string. Find if it is a negative number, if true return the absolute value, in other cases return -1.

input: "-123"

output: 123

input: "@123"

output: -1

```
package Set1;
import java.util.*;
public class ClassSet27 {
    public static int negativeString(String s1){
        int n1=0;
        if(s1.startsWith("-")){
            int n=Integer.parseInt(s1);
            if(n<0)
                n1=Math.abs(n);
        }
    }
}
```

```

        else
            n1=-1;
        return n1;
    }
public static void main(String[] args) {
    Scanner s=new Scanner(System.in);
    String s1=s.next();
    System.out.println(negativeString(s1));
}

```

---



---



---

28.An arraylist of Strings is given as input. The count of the String elements that are not of size input2 string is to be returned.

input1: {"aaa","bb","cccc","d"}

input2: "bb"

output: 3("bb"-length:2)

```

package Set1;
import java.util.*;
public class ClassSet28 {
    public static int StringsNotOfGivenLength(List<String> l1,String s1){
        int n1=s1.length();
        int c=0;
        for(int i=0;i<l1.size();i++){
            int n2=l1.get(i).length();
            if(n1!=n2)
                c++;
        }
        return c;
    }
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        System.out.println("enter the no.of elements:");
        int n=s.nextInt();
        List<String> l1=new ArrayList<String>();
        for(int i=0;i<n;i++)
            l1.add(s.next());
        System.out.println("enter the input string:");
        String s1=s.next();
        System.out.println(StringsNotOfGivenLength(l1,s1));
    }
}

```

---



---



---

29.An array of integers is given.The given element is removed new array is returned.

input1:{10,10,20,30,76}

output: {20,20,76}(10 is asked to remove)

```
package Set1;
import java.util.*;
public class ClassSet29 {
    public static int[] removalOfGivenElementFromArray(int[] a,int c){
        List<Integer> l1=new ArrayList<Integer>();
        for(int i=0;i<a.length;i++)
            if(a[i]!=c)
                l1.add(a[i]);
        int b[]={};
        for(int i=0;i<b.length;i++)
            b[i]=l1.get(i);
        return b;
    }
    public static void main(String[] args) {
        int a[]={10,10,20,30,40,50};
        int c=10;
        int[] b=removalOfGivenElementFromArray(a,c);
        for(int d:b)
            System.out.println(d);
    }
}
```

---

---

30. Find the number of days between two input dates.

```
package Set1;
import java.text.*;
import java.util.*;
public class ClassSet30 {
    public static int dateDifference(String s1,String s2) throws ParseException{
        SimpleDateFormat sd=new SimpleDateFormat("dd/MM/yyyy");
        Date d=sd.parse(s1);
        Calendar c=Calendar.getInstance();
        c.setTime(d);
        long d1=c.getTimeInMillis();
        d=sd.parse(s2);
        c.setTime(d);
        long d2=c.getTimeInMillis();
        int n=Math.abs((int) ((d1-d2)/(1000*3600*24)));
        return n;
    }
    public static void main(String[] args) throws ParseException {
        String s1="27/12/2009";
        String s2="15/09/2012";
        System.out.println(dateDifference(s1,s2));
    }
}
```

---

---

31.Enter your name and return a string "print the title first and then comma and then first letter of initial name.

```
package Set1;
import java.util.*;
public class ClassSet31 {
    public static String retrieveName(String s1){
        StringTokenizer t=new StringTokenizer(s1," ");
        String s2=t.nextToken();
        String s3=t.nextToken();
        StringBuffer sb=new StringBuffer(s2);
        sb.append(',').append(s3.charAt(0));
        return sb.toString();
    }
    public static void main(String[] args) {
        String s1="Bhavane Raghupathi";
        System.out.println(retrieveName(s1));
    }
}
```

---

32.Write a Program that accepts a string and removes the duplicate characters.

```
package Set1;
public class ClassSet32 {
    public static String removalOfDuplicateCharacters(String s1){
        StringBuffer sb=new StringBuffer(s1);
        for(int i=0;i<s1.length();i++)
            for(int j=i+1;j<s1.length();j++)
                if(s1.charAt(i)==s1.charAt(j))
                    sb.deleteCharAt(j);
        return sb.toString();
    }
    public static void main(String[] args) {
        String s1="bhavane";
        System.out.println(removalOfDuplicateCharacters(s1));
    }
}
```

---

33.validate a password

- i) there should be atleast one digit
- ii) there should be atleast one of '#','@' or '\$' .
- iii)the password should be of 6 to 20 characters
- iv) there should be more uppercase letter than lower case.
- v) should start with an upper case and end with lower case

```

package Set1;
import java.util.*;
public class ClassSet33 {
public static void main(String[] args) {
    Scanner s=new Scanner(System.in);
    String st=s.next();
    boolean b=validatingPassword(st);
    if(b==true)
        System.out.println("valid password");
    else
        System.out.println("Invalid Password");
}

public static boolean validatingPassword(String st)
{
    boolean b1=false,b2=false;
    if(Character.isUpperCase(st.charAt(0)))
        if(Character.isLowerCase(st.charAt(st.length()-1)))
            if(st.length()>=6 && st.length()<=20)
                for (int i = 0; i < st.length();i++)
                {
                    char c = st.charAt(i);
                    if(Character.isDigit(c)){
                        b1=true; break; } }
    int x=0,y=0;
    for(int i = 0; i < st.length(); i++)
        if(Character.isUpperCase(st.charAt(i)))
            x++;
        else if(Character.isLowerCase(st.charAt(i)))
            y++;
    if(b1==true)
        if(x>y)
            for (int i = 0; i < st.length();i++)
            {
                char c = st.charAt(i);
                if(c=='#' || c=='@' || c=='$'){
                    b2=true; break; } }
    return b2;
}

}

```

---



---



---

34. find the average of the maximum and minimum number in an integer array

```

package Set1;
import java.util.*;
public class ClassSet34 {
public static void main(String[] args) {
    int[] a={10,54,23,14,32,45};
    System.out.println(avgOfMaxandMinNoinArray(a));
}
public static int avgOfMaxandMinNoinArray(int[] a) {

```

```
        Arrays.sort(a);
        int b=a[0];
        int c=a[a.length-1];
        return (b+c)/2;
    }
}
```

---

35.validate the ip address in the form a.b.c.d  
where a,b,c,d must be between 0and 255  
if validated return 1 else return 2

```
package Set1;
import java.util.*;
public class ClassSet35 {
    public static void main(String[] args) {
        String ipAddress="010.230.110.160";
        boolean b=validateIpAddress(ipAddress);
        if(b==true)
            System.out.println("valid ipAddress");
        else
            System.out.println("not a valid ipAddress");
    }
    public static boolean validateIpAddress(String ipAddress) {
        boolean b1=false;
        StringTokenizer t=new StringTokenizer(ipAddress,".");
        int a=Integer.parseInt(t.nextToken());
        int b=Integer.parseInt(t.nextToken());
        int c=Integer.parseInt(t.nextToken());
        int d=Integer.parseInt(t.nextToken());
        if((a>=0 && a<=255)&&(b>=0 && b<=255)&&(c>=0 && c<=255)&&(d>=0 && d<=255))
            b1=true;
        return b1;
    }
}
```

---

36. find the three consecutive characters in a string. if the string consists any three consecutive characters return 1 else return -1  
like AAAxyzaaa will return true.

```
package Set1;
public class ClassSet36 {
```

```

public static void main(String[] args) {
    String s1="aaaaxyzAAA";
    boolean b=consecutiveCharactersCheck(s1);
    if(b==true)
        System.out.println("presence of consecutive characters");
    else
        System.out.println("absence of consecutive characters");
}

public static boolean consecutiveCharactersCheck(String s1) {
    boolean b=false;
    int c=0;
    for(int i=0;i<s1.length()-1;i++)
        if((s1.charAt(i+1)-s1.charAt(i))==1)
            c++;
    if(c>=2)
        b=true;
    return b;
}

```

---



---



---

37. String encryption. replace the odd-index character with next character(if it is 'z' replace with 'a'..if 'a' with 'b' as such), leave the even index as such. return the encrypted string.

```

package Set1;
public class ClassSet37 {
public static void main(String[] args) {
    String s="preethi";
    System.out.println(oddEncryptionOfString(s));
}

public static String oddEncryptionOfString(String s) {
    StringBuffer sb=new StringBuffer();
    for(int i=0;i<s.length();i++){
        char c=s.charAt(i);
        if(c%2!=0){
            c=(char)(c+1);
            sb.append(c); }
        else
            sb.append(c); }
    return sb.toString();
}

```

---



---

38. Retrieve the max from array which is in a odd-index

```

package Set1;
import java.util.*;

```

```

public class ClassSet38 {
    public static void main(String[] args) {
        int[] a={10,23,45,54,32,14,55,65,56};
        System.out.println(retrieveMaxFromOddIndex(a));
    }
    public static int retrieveMaxFromOddIndex(int[] a) {
        List<Integer> l=new ArrayList<Integer>();
        for(int i=0;i<a.length;i++)
            if(i%2!=0)
                l.add(a[i]);
        int b=0;
        for(int i=0;i<l.size();i++){
            int c=(Integer) l.get(i);
            if(c>b)
                b=c; }
        return b;
    }
}

```

---



---



---

39.GIVEN A STRING 555-666-1234  
DISPLAY AS 55-56-661-234?

```

package Set1;
import java.util.StringTokenizer;
public class ClassSet39 {
    public static void main(String[] args) {
        String s="555-666-1234";
        System.out.println(display(s));
    }
    public static String display(String s) {
        StringTokenizer t=new StringTokenizer(s,"-");
        String s1=t.nextToken();
        String s2=t.nextToken();
        String s3=t.nextToken();

        StringBuffer sb=new StringBuffer();
        sb.append(s1.substring(0, s1.length()-1)).append('-');
        sb.append(s1.charAt(s1.length()-1)).append(s2.charAt(0)).append('-');
        sb.append(s2.substring(1, s2.length())).append(s3.charAt(0)).append('-');
        sb.append(s3.substring(1, s3.length()));

        return sb.toString();
    }
}

```

---



---

40.input1="Rajasthan";  
input2=2;

```
input3=5;
output=hts;

package Set1;
public class ClassSet40 {
    public static void main(String[] args) {
        String input1="Rajasthan";
        int input2=2, input3=5;
        System.out.println(retrieveString(input1,input2,input3));
    }
    public static String retrieveString(String input1, int input2, int input3) {
        StringBuffer sb=new StringBuffer(input1);
        sb.reverse();
        String output=sb.substring(input2, input3);
        return output;
    }
}
```

---

---

---

```
41.input1={1,2,3}
input2={3,4,5}
input3;+(union)
output:inp1+inp2
input1:{1,2,3,4,5}
input 2:{2,3,4,5}
input3=*(intersection)
output:inp1*inp2
INPUT1:{1,2,3,4,5}
INPUT2:{3,6,7,8,9}
INPUT3:-(MINUS)
output:inp1-inp2
```

```
package Set1;
import java.util.*;
public class ClassSet41 {
    public static void main(String[] args) {
        int[] a={1,2,3,4,5};
        int[] b={0,2,4,6,8};
        Scanner s=new Scanner(System.in);
        int c=s.nextInt();
        int d[]=arrayFunctions(a,b,c);
        for(int e:d)
            System.out.println(e);
    }
}
```

```

public static int[] arrayFunctions(int[] a, int[] b, int c) {
    List<Integer> l1=new ArrayList<Integer>();
    List<Integer> l2=new ArrayList<Integer>();
    List<Integer> l3=new ArrayList<Integer>();
    for(int i=0;i<a.length;i++)
        l1.add(a[i]);
    for(int i=0;i<b.length;i++)
        l2.add(b[i]);

    switch(c){
        case 1:
            l1.removeAll(l2);
            l1.addAll(l2);
            Collections.sort(l1);
            l3=l1;
            break;
        case 2:
            l1.retainAll(l2);
            Collections.sort(l1);
            l3=l1;
            break;
        case 3:
            if(l1.size()==l2.size())
                for(int i=0;i<l1.size();i++)
                    l3.add(Math.abs(l2.get(i)-l1.get(i)));
            break;
    }
    int[] d=new int[l3.size()];
    for(int i=0;i<d.length;i++)
        d[i]=l3.get(i);
    return d;
}

```

---



---



---



---

42. input1=commitment;  
 output=cmmitmnt;  
 c be the first index position  
 remove even vowels from the string

```

package Set1;
public class ClassSet42 {
    public static void main(String[] args) {
        String s1="cOmmitment";
        System.out.println(removeEvenElements(s1));
    }
    public static String removeEvenElements(String s1) {
        StringBuffer sb1=new StringBuffer();
        for(int i=0;i<s1.length();i++)

```

```

        if((i%2)==0)
            sb1.append(s1.charAt(i));
        else if((i%2)!=0)
            if(s1.charAt(i)!='a' && s1.charAt(i)!='e' && s1.charAt(i)!='i' && s1.charAt(i)!='o' &&
s1.charAt(i)!='u')
                if(s1.charAt(i)!='A' && s1.charAt(i)!='E' && s1.charAt(i)!='T' && s1.charAt(i)!='O' &&
s1.charAt(i)!='U')
                    sb1.append(s1.charAt(i));
    return sb1.toString();
}
}

```

---



---



---

43. Retrieve the odd-position digits from input integer array. Multiply them with their index and return their sum.

```

package Set1;
public class ClassSet43 {
    public static void main(String[] args) {
        int[] a={5,1,6,2,9,4,3,7,8};
        System.out.println("Sum Of Odd Position Elements Multiplied With Their Index Is:");
        System.out.println(retrievalOfOddPositionElements(a));
    }
    public static int retrievalOfOddPositionElements(int[] a) {
        int s=0;
        for(int i=0;i<a.length;i++)
            if(i%2!=0)
                s+=a[i]*i;
        return s;
    }
}

```

---



---

44. Return the number of days in a month, where month and year are given as input.

```

package Set1;
import java.util.*;
public class ClassSet44 {
    public static void main(String[] args){
        String s1="02/2011";
        System.out.println(noOfDaysInAmonth(s1));
    }
    public static int noOfDaysInAmonth(String s1){
        int n=0;
        StringTokenizer r=new StringTokenizer(s1,"/");
        int n1=Integer.parseInt(r.nextToken());
        int n2=Integer.parseInt(r.nextToken());
        if(n1==1 || n1==3 || n1==5 || n1==7 || n1==8 || n1==10 || n1==12)
            n=31;
        else if(n1==4 || n1==6 || n1==9 || n1==11)
            n=30;
        else if(n1==2){
            if(n2%4==0)

```

```
        n=29;
    else
        n=28; }
    return n;
}
-----
```

45. Retrieve the non-prime numbers within the range of a given input. Add-up the non-prime numbers and return the result.

```
package Set1;
import java.util.*;
public class ClassSet45 {
public static void main(String[] args) {
    int i=20,j=40;
    System.out.println("sum of non-prime nos. is:"+additionOfnonPrimeNos(i,j));
}
public static int additionOfnonPrimeNos(int i, int j){
    int k=0;
    List<Integer> l1=new ArrayList<Integer>();
    List<Integer> l2=new ArrayList<Integer>();
    for(int a=i;a<=j;a++){
        int c=0;
        for(int b=2;b<a;b++)
            if(a%b==0)
                c++;
        if(c==0)
            l1.add(a); }
    for(int e=i;e<=j;e++)
        l2.add(e);
    l2.removeAll(l1);
    for(int d=0;d<l2.size();d++)
        k+=l2.get(d);
    return k;
}
}
-----
```

46. Retrieve the elements in a array, which is an input, which are greater than a specific input number. Add them and find the reverse of the sum.

```
package Set1;
public class ClassSet46 {
public static void main(String[] args) {
    int[] a={23,35,11,66,14,32,65,56,55,99};
    int b=37;
    System.out.println(additionOfRetrievedElements(a,b));
}
public static int additionOfRetrievedElements(int[] a, int b) {
    int n=0;
    for(int i=0;i<a.length;i++)
```

```
    if(a[i]>b)
        n+=a[i];
    return n;
}
```

---

---

47.Input a hashmap. Count the keys which are not divisible by 4 and return.

```
package Set1;
import java.util.*;
public class ClassSet47 {
    public static void main(String[] args) {
        Map<Integer, String> m1=new HashMap<Integer, String>();
        m1.put(24, "preethi");
        m1.put(32, "bhanu");
        m1.put(25, "menu");
        m1.put(31, "priya");
        System.out.println(fetchingKeysDivisibleByFour(m1));
    }
    public static int fetchingKeysDivisibleByFour(Map<Integer, String> m1) {
        int k=0;
        Iterator<Integer> i=m1.keySet().iterator();
        loop:
        while(i.hasNext()){
            int j=i.next();
            if(j%4!=0)
                k++;
            continue loop; }
        return k;
    }
}
```

## **JAVA ICT ANS:**

### **1. DATE VALIDATION:**

Main:

```
import java.util.*;  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        Scanner sc = new Scanner(System.in);  
  
        String s= sc.next();  
  
        int b = UserMainCode.getvalues(s);  
  
        if(b==1)  
  
            System.out.println("Valid");  
  
        else  
  
            System.out.println("Invalid");  
  
    }  
  
}
```

.....

UserMainCode:

```
import java.text.ParseException;  
  
import java.text.SimpleDateFormat;  
  
import java.util.Date;  
  
public class UserMainCode {  
  
    public static int getvalues(String s) {  
  
        if(s.matches("[0-9]{2}[.]{1}[0-9]{2}[.]{1}[0-9]{4}"))  
  
    {
```

```
SimpleDateFormat sdf=new SimpleDateFormat("dd.MM.yyyy");
sdf.setLenient(false);

try
{
    Date d1=sdf.parse(s);
    return 1;
} catch (ParseException e) {
    return -1;
}

else if(s.matches("[0-9]{2}[/]{1}[0-9]{2}[/][0-9]{4}"))
{
    SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
    sdf.setLenient(false);

    try
    {
        Date d1=sdf.parse(s);
        return 1;
    } catch (ParseException e) {
        return -1;
    }
}

else if(s.matches("[0-9]{2}[-]{1}[0-9]{2}[-][0-9]{4}"))
{
    SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");
}
```

```
sdf.setLenient(false);

try
{
    Date d1=sdf.parse(s);

    return 1;
} catch (ParseException e) {
    return -1;
}
}

else
return -1;
}
}
```

## **2.DIGITS – II:**

MAIN:

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args)

    {
        Scanner s=new Scanner(System.in);

        int a=s.nextInt();

        int sum=UserMainCode.getDigitSum(a);

        System.out.println(sum);
    }
}
```

```
}
```

```
}
```

.....

USERMAINCODE:

```
public class UserMainCode
```

```
{
```

```
public static int getDigitSum(int n)
```

```
{
```

```
int sum = 0 ;
```

```
int n1=n;
```

```
while(n>10)
```

```
{
```

```
int a = 0 ; sum = 0;
```

```
while(n!=0)
```

```
{
```

```
a = n%10;
```

```
sum+=a;
```

```
n=n/10;
```

```
}
```

```
n=sum;
```

```
}
```

```
return sum;
```

```
}
```

```
}
```

### **3.MAX SUBSTRING:**

Main:

```
import java.util.*;
public class Main {
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        String input1=sc.next();
        String input2=sc.next();
        System.out.println(UserMainCode.extractMax(input1,input2));
    }
}
```

.....

Usermaincode:

```
import java.util.StringTokenizer;
import java.util.*;
public class UserMainCode {
    public static String extractMax(String input1,String input2){
        int max=0;
        String s3=null;
        StringTokenizer st=new StringTokenizer(input1,"-");
        while( st.hasMoreTokens())
        {
            String s2=st.nextToken();
            int n=s2.length();

```

```
if(n>max)
{
    max=n;
    s3=s2;
}
return(s3);
}}
```

## **4.STRING FINDER:**

USERMAINCODE:

```
public class UserMainCode {
    public static int stringFinder(String s1,String s2,String s3)
    {
        String a1=s1.toLowerCase();
        String a2=s2.toLowerCase();
        String a3=s3.toLowerCase();
        if(a1.contains(a2)&&a1.contains(a3))
        {
            if(a1.indexOf(a2)<a1.indexOf(a3))
            {
                return 1;
            }
        }
    }
}
```

```
        return 2;
    }
    return 0;
}

.....
MAIN:
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.next();
        String s2=s.next();
        String s3=s.next();
        int b=UserMainCode.stringFinder(s1, s2, s3);
        if(b==1)
        {
            System.out.println("yes");
        }
        else
            System.out.println("No");
        s.close();
    }
}
```

## **5. Name Shrinking:**

```
Main
import java.text.*;
import java.util.*;
public class Main {
public static void main(String[] args) {
Scanner sc=new Scanner(System.in);
String s1=sc.nextLine();
System.out.println(UserMainCode.getFormatedString(s1));
}
}
UserMainCode
class UserMainCode
{
public static String getFormatedString(String s1) {
StringBuffer sb=new StringBuffer();
StringTokenizer st=new StringTokenizer(s1, " ");
String s2=st.nextToken();
String s3=st.nextToken();
String s4=st.nextToken();
sb.append(s4).append(" ");
sb.append(s3.substring(0,1));
sb.append(".");
sb.append(s2.substring(0,1));
return sb.toString();
}
}
```

## **6.GRADE CALCULATOR:**

### **UserMainCode:**

```
import java.util.Iterator;
import java.util.HashMap;
import java.util.TreeMap;
public class UserMainCode
```

```
{  
public static  
TreeMap<Integer,String>calculateGrade(HashMap<Integer,Integer>hm)  
{  
TreeMap<Integer,String>tm=new TreeMap<Integer,String>();  
Iterator<Integer> it=hm.keySet().Iterator();  
While(it.hasNext())  
{  
int id=it.next();  
int mark=hm.get(id);  
if(mark>=80)  
tm.put(id,"GOLD");  
else if(mark<80 && mark>=60)  
tm.put(id,"SILVER");  
else if(mark<60 && mark>=45)  
tm.put(id,"BRONZE");  
else  
tm.put(id,"FAIL");  
}  
Return tm;  
}}  
}
```

## **MAIN:**

```
Import java.util.HashMap;  
Import java.util.Iterator;  
Import java.util.TreeMap;
```

```
Import java.util.Scanner;

public class main {

public static void main(string[] args){

Scanner sc= new Scanner(System.in);

int s=sc.nextInt();

HashMap<Integer, Integer> hm=new HashMap<Integer, Integer>();

for(int i=0;i<5;i++)

{

hm.put(sc.nextInt(),sc.nextInt());

}

TreeMap<Integer, String> tm=new TreeMap<Integer, String>();

Tm=UserMaiCode.calculateGrade(hm);

Iterator<Integer> it=hm.keySet().iterator();

For(int i=0;i<tm;i++)

{

int n=it.next();

String fac=tm.get(n);

System.out.println(n);

System.out.println(fac);

}

}

}
```

# **JAVA ICT CODES**

## **Rainfall Report Automation**

### **AnnualRainfall**

```
public class AnnualRainfall {  
  
    private int cityPincode;  
    private String cityName;  
    private double averageAnnualRainfall;  
  
    public int getCityPincode() {  
        return cityPincode;  
    }  
  
    public void setCityPincode(int cityPincode) {  
        this.cityPincode = cityPincode;  
    }  
  
    public String getCityName(){  
        return cityName;  
    }  
  
    public void setCityName(String cityName){  
        this.cityName = cityName;  
    }  
}
```

```
public double getAverageAnnualRainfall(){
    return averageAnnualRainfall;
}

public void
setAverageAnnualRainfall(double
averageAnnualRainfall){
    this.averageAnnualRainfall =
averageAnnualRainfall;
}

public void calculateAverageAnnualRainfall
(double monthlyRainfall [ ]){
    double average=0;
    for(int i=0;i<monthlyRainfall.length;i++)
    {
        average+=monthlyRainfall[i];
    }
    average/=12;
    this.averageAnnualRainfall=average;
}

}
```

## DBHandler

```
import java.io.FileInputStream;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;

public class DBHandler {

    private static Connection con = null;
    private static Properties props = new Properties();
    //Write the required business logic as
    //expected in the question description
    public Connection establishConnection()
        throws ClassNotFoundException,
        SQLException {
        try{
            FileInputStream fis
                = null;
            fis = new
            FileInputStream("db.properties");
            props.load(fis);

            // load the Driver
            Class
                Class.forName(props.getProperty("d
b.classname"));

            // create the
            // connection now
            con =
            DriverManager.getConnection(props.getPro
perty("db.url"),props.getProperty("db.usern
ame"),props.getProperty("db.password"));
        }
    }
}
```

```
        }

        catch(IOException e){

            e.printStackTrace();

        }

        return con;

    }

}
```

### **InvalidCityPincodeException**

```
@SuppressWarnings("serial")

public class InvalidCityPincodeException extends Exception {

    public
    InvalidCityPincodeException(String s)

    {

        super(s);

    }

}
```

### **Main**

```
import java.io.IOException;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

public class Main {

    public static void main(String[] args)

        throws IOException,

        InvalidCityPincodeException,

        ClassNotFoundException, SQLException {
```

```

        RainfallReport rf = new
        RainfallReport();

        List<AnnualRainfall> avgli = new
        ArrayList<AnnualRainfall>();

        avgli =
        rf.generateRainfallReport("AllCityMonthlyR
ainfall.txt");

        List<AnnualRainfall> maxli = new
        ArrayList<AnnualRainfall>();

        maxli =
        rf.findMaximumRainfallCities();

        for (int i = 0; i < maxli.size(); i++) {

            AnnualRainfall ob =
            maxli.get(i);

            System.out.println("City
Pincode:" + ob.getCityPincode());

            System.out.println("City
Name:" + ob.getCityName());

            System.out.println("Average
RainFall:" + ob.getAverageAnnualRainfall());

        }

    }

}

```

## **RainfallReport**

```

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.sql.Connection;

```

```
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;

public class RainfallReport {

    public List<AnnualRainfall>
    generateRainfallReport(String filePath)
    throws IOException {

        List<AnnualRainfall> avgList=new
        ArrayList<>();

        FileReader fr = new FileReader(new
        File(filePath));

        BufferedReader br = new
        BufferedReader(fr);

        String l;
        while((l=br.readLine())!=null)
        {
            String[] a=l.split(",");
            String pincode=a[0];
            try
            {
                if(validate(pincode))
                {
                    double[]
                    monthlyRainFall=new double[12];
                    for(int
i=2;i<=13;i++)
                    {

```

```
        monthlyRainFall[i-  
2]=Double.parseDouble(a[i]);  
    }  
  
    AnnualRainfall ar=new  
    AnnualRainfall();  
  
    ar.calculateAverageAnnualRainfall(  
monthlyRainFall);  
  
    ar.setCityName(a[1]);  
  
    ar.setCityPincode(Integer.parseInt(p  
incode));  
  
    avgList.add(ar);  
}  
}  
  
}  
  
catch(InvalidCityPincodeException  
e)  
{  
  
System.out.println(e.getMessage());  
}  
}  
  
}  
  
br.close();  
  
return avgList;  
}  
  
}  
  
public List<AnnualRainfall>  
findMaximumRainfallCities() throws  
SQLException, ClassNotFoundException,  
IOException {  
  
DBHandler d=new DBHandler();
```

```
        List<AnnualRainfall> finalList=new
        ArrayList<>();

        Connection
c=d.establishConnection();

        Statement s=c.createStatement();

        String sql = "SELECT * FROM
ANNUALRAINFALL WHERE
AVERAGE_ANNUAL_RAINFALL IN (SELECT
MAX(AVERAGE_ANNUAL_RAINFALL) FROM
ANNUALRAINFALL)";

        ResultSet rs=s.executeQuery(sql);

        while(rs.next())
        {

            AnnualRainfall ar=new
AnnualRainfall();

            ar.setCityName(rs.getString(2));

            ar.setCityPincode(Integer.parseInt(r
s.getString(1)));

            ar.setAverageAnnualRainfall(Double
.parseDouble(rs.getString(3)));

            finalList.add(ar);
        }

        return finalList;
    }
}
```

```
public boolean validate(String cityPincode)
throws InvalidCityPincodeException {

    if(cityPincode.length()==5)

    {

        return true;
    }

    else
```

```
        {
            throw new
            InvalidCityPincodeException("Invalid
            sCityPincode Exception");
        }
    }
```

## Electricity Bill Automation

### DBHandler

```
import java.sql.Connection;
import java.io.FileInputStream;
import java.io.IOException;
import java.sql.*;
import java.util.Properties;
import java.io.FileNotFoundException;
```

```
public class DBHandler {
```

```
    public Connection establishConnection()
        throws ClassNotFoundException,
        SQLException, FileNotFoundException {
```

```
        Connection con = null;
```

```
Properties props = new Properties();

// this try block reads the db
Properties file and establishConnection.

try{

    FileInputStream fis = new
FileInputStream("src/db.properties");

    props.load(fis);

Class.forName(props.getProperty("db.cl
assname"));

con =
DriverManager.getConnection(props.get
Property("db.url"),props.getProperty("d
b.username"),props.getProperty("db.pa
ssword"));

}

catch(IOException e){

    e.printStackTrace();

}

return con;

}

//fill code here

}
```

## ElectricityBill

```
//This is the POJO/model class

public class ElectricityBill {

    private String consumerNumber;
    private String consumerName;
    private String consumerAddress;
    private int unitsConsumed;
    private double billAmount;

    public String getConsumerNumber() {
        return consumerNumber;
    }

    public void setConsumerNumber(String consumerNumber) {
        this.consumerNumber = consumerNumber;
    }

    public String getConsumerName() {
        return consumerName;
    }

    public void setConsumerName(String consumerName) {
        this.consumerName = consumerName;
    }
}
```

```
public String getConsumerAddress() {  
    return consumerAddress;  
}
```

```
public void setConsumerAddress(String  
consumerAddress) {  
    this.consumerAddress =  
consumerAddress;  
}
```

```
public int getUnitsConsumed() {  
    return unitsConsumed;  
}
```

```
public void setUnitsConsumed(int  
unitsConsumed) {  
    this.unitsConsumed =  
unitsConsumed;  
}
```

```
public double getBillAmount() {  
    return billAmount;  
}
```

```
public void setBillAmount(double  
billAmount) {  
    this.billAmount = billAmount;  
}
```

```
//Write the required business logic as  
expected in the question description  
  
public void calculateBillAmount() {  
  
    // method for calculating the bill  
    amount.  
  
    int units = unitsConsumed;  
  
    double bill = 0;  
  
  
    if(units <= 100){  
  
        bill = 0;  
  
    }  
  
    if(units > 100 && units <= 300){  
  
        bill = (units-100) * 1.5;  
  
    }  
  
    if(units > 300 && units <= 600){  
  
        bill = 200 * 1.5 + (units-300) * 3.5;  
  
    }  
  
    if(units > 600 && units <= 1000){  
  
        bill = 200 * 1.5 + 300 * 3.5 + (units-  
        600) * 5.5;  
  
    }  
  
    if(units > 1000){  
  
        bill = 200 * 1.5 + 300 * 3.5 + 400 *  
        5.5 + (units-1000) * 7.5;  
  
    }  
  
  
    setBillAmount(bill);  
  
  
    //fill the code
```

```
}
```

```
}
```

## ElectricityBoard

```
import java.util.List;
import java.util.*;
import java.io.FileReader;
import java.io.File;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.regex.Pattern;
import java.sql.SQLException;
import java.sql.Connection;
import java.sql.PreparedStatement;

public class ElectricityBoard {

    //write the required business logic methods as expected in the question description

    public void addBill(List<ElectricityBill> billList) {

        DBHandler db = new DBHandler();

        try(Connection con = db.establisConnection()){

            PreparedStatement stmt = con.prepareStatement("insert into ElectricityBill
values(?,?,?,?,?);");

```

```
// for loop to insert the values into the table  
  
for(ElectricityBill obj : billList){  
  
    stmt.setString(1,obj.getConsumerNumber());  
  
    stmt.setString(2,obj.getConsumerName());  
  
    stmt.setString(3,obj.getConsumerAddress());  
  
    stmt.setInt(4,obj.getUnitsConsumed());  
  
    stmt.setDouble(5,obj.getBillAmount());  
  
  
    stmt.execute();  
  
}  
  
}  
  
catch(ClassNotFoundException e){  
  
    e.printStackTrace();  
  
}  
  
  
catch(FileNotFoundException e){  
  
    e.printStackTrace();  
  
}  
  
  
catch(SQLException e){  
  
    e.printStackTrace();  
  
}  
  
  
//fill the code
```

```
}
```

```
public List<ElectricityBill> generateBill(String filePath) {  
  
    List <ElectricityBill> list = new ArrayList<>();  
    File f = new File (filePath);  
  
    // this try block is for opening and reading the file  
    try(BufferedReader br = new BufferedReader(new FileReader(f)))  
    {  
        String line = null;  
  
        while((line = br.readLine())!= null)  
        {  
            String records[] = null;  
            String consumerNumber = "";  
            String consumerName = "";  
            String consumerAddress = "";  
            int unitsConsumed = 0;  
  
            records = line.split(",");  
            consumerNumber = records[0];  
            consumerName = records[1];  
            consumerAddress = records[2];  
            unitsConsumed = Integer.parseInt(records[3]);  
  
            //this try block checks for the validated consumerNumber  
            try{  
                  
            }  
            catch(Exception e){  
                System.out.println("Consumer Number "+ consumerNumber + " is invalid");  
            }  
            ElectricityBill bill = new ElectricityBill(consumerNumber, consumerName, consumerAddress, unitsConsumed);  
            list.add(bill);  
        }  
    }  
}
```

```
if(validate(consumerNumber)){

    ElectricityBill obj = new ElectricityBill();

    obj.setConsumerNumber(consumerNumber);

    obj.setConsumerName(consumerName);

    obj.setConsumerAddress(consumerAddress);

    obj.setUnitsConsumed(unitsConsumed);

    obj.calculateBillAmount();

    list.add(obj);

}

}

catch(InvalidConsumerNumberException e){

    System.out.println(e.getMessage());

}

}

}

catch(FileNotFoundException e){

    e.printStackTrace();

}

catch(IOException e){

    e.printStackTrace();

}

}

return list;

//fill the code

}

public boolean validate(String consumerNumber) throws
    InvalidConsumerNumberException {
```

```

// method for validating the consumerNumber

boolean isValid = Pattern.matches("^[0][0-9]{9}" , consumerNumber);

if(!isValid){
    throw new InvalidConsumerNumberException("Invalid Consumer Number");
}

return true;

//fill the code

}

}

```

## InvalidConsumerNumberException

```

//make the required changes to this class so that InvalidConsumerNumberException is of
//type exception.

public class InvalidConsumerNumberException extends Exception{

    public InvalidConsumerNumberException(String message)
    {
        super(message);
    }
}
```

```
//fill the code
```

```
}
```

## Main

```
import java.util.*;  
import java.util.List;  
import java.util.ArrayList;  
public class Main {  
  
    public static void main(String[] args) {  
        Scanner sc= new Scanner(System.in);  
  
        String filePath = "src/ElectricityBill.txt";  
  
        List<ElectricityBill> list = new ArrayList<>();  
  
        ElectricityBoard eb = new ElectricityBoard();  
        list = eb.generateBill(filePath);  
  
        for(ElectricityBill obj: list){  
            System.out.println(obj.getConsumerNumber() + " " + obj.getConsumerName() + " " +  
                obj.getBillAmount());  
        }  
  
        eb.addBill(list);  
    }  
}
```

```
        System.out.println("Successfully Inserted");

        sc.close();
        //fill your code here

    }

}
```

## **CreditCardAdminSystem**

### **CreditCardDAO**

```
*****
*****
* This class CreditCardDAO is used to persist or retrieve data from database.
*
* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS,
* EXCEPTION CLAUSES, RETURN TYPES
*
* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
*
* DO NOT SUBMIT THE CODE WITH COMPILE ERRORS
*
* DO TEST YOUR CODE USING MAIN METHOD
*
* CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC
*
* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF
* NEED BE,
*
* YOU CAN CATCH THEM AND THROW ONLY
*
* THE APPLICATION SPECIFIC EXCEPTION AS PER EXCEPTION CLAUSE
*
```

```
*****
*****
/
package com.cts.creditcard.dao;

import java.sql.Connection;
import java.util.List;

import com.cts.creditcard.exception.CreditCardAdminSystemException;
import com.cts.creditcard.vo.CreditCard;

public class CreditCardDAO {

    private static Connection conn = null;

    public Boolean
        addCreditCardDetails(List<CreditCard>
    cards) throws
        CreditCardAdminSystemException {
        //TODO add your code here
        return false; //TODO CHANGE
        THIS RETURN TYPE
    }

}
```

## CreditCardAdminSystemException

```
*****
*****
```

- \* This class CreditCardAdminSystemException is as exception class.
- \*
- \* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS,  
EXCEPTION CLAUSES, RETURN TYPES
- \* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
- \* DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS
- \* DO TEST YOUR CODE USING MAIN METHOD
- \* CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC
- \* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF  
NEED BE,
- \* YOU CAN CATCH THEM AND THROW ONLY
- \* THE APPLICATION SPECIFIC EXCEPTION AS PER EXCEPTION CLAUSE

\*

```
*****  
*****  
/
```

package com.cts.creditcard.exception;

```
public class CreditCardAdminSystemException extends Exception {
```

```
    private static final long serialVersionUID  
    = -6349759544203601561L;
```

```
    //TODO add your constructors here
```

```
}
```

## MainApp

```
*****  
*****
```

\* This class MainApp is used to run the service methods and to test the database.

```

*
* YOU CAN INVOKE THE METHODS AS REQUIRED FROM HERE TO TEST THE APP
* DO NOT USE VIA COMMAND LINE ARGUMENTS FROM MAIN METHOD, UNLESS SPECIFIED
*
*****
***** /
package com.cts.creditcard.main;

public class MainApp {
    public static void main(String ag[]) {
        //TODO add your code here
    }
}

```

## CreditCardAdminService

```

/*****
*****
* This class CreditCardAdminService is used to handle business logic for the proposed
system.
*
* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS,
EXCEPTION CLAUSES, RETURN TYPES
*
* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
*
* DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS
*
* DO TEST YOUR CODE USING MAIN METHOD
*
* CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC
*
* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF
NEED BE,

```

\* YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER  
EXCEPTION CLAUSE

\*

\*\*\*\*\*

\*\*\*\*\*

/

```
package com.cts.creditcard.service;
```

```
import java.util.List;
```

```
import com.cts.creditcard.exception.CreditCardAdminSystemException;
```

```
import com.cts.creditcard.vo.CreditCard;
```

```
public class CreditCardAdminService {
```

```
/**
```

```
* @param records
```

```
* @return List<Customer>
```

```
*/
```

```
public static List<CreditCard>
```

```
buildMasterCreditCardList(List<String>
```

```
records) {
```

```
// TODO Add your logic here
```

```
return null; // TODO change this  
return value
```

```
}
```

```
/**
```

```
* @param billAmount
```

```
* @return Double
```

```

        */

    public static Double
    getBillAmountWithLatePaymentCharges
    (Double billAmount) {

        // TODO add your logic here

        return 0.00; // TODO change this
        return value

    }

    /**
     * @param inputFeed
     * @return Boolean
     * @throws
     CreditCardAdminSystemException
     */

    public Boolean
    addCreditCardDetails(String inputFeed)
    throws
    CreditCardAdminSystemException {

        // TODO add your logic here

        return null; //TODO change this
        return value

    }

}

```

## ApplicationUtil

```

*****
*****
```

\* This class ApplicationUtil is used for any utility methods needed for service or dao classes

\*

\* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS,  
EXCEPTION CLAUSES, RETURN TYPES

\* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE

\* DO NOT SUBMIT THE CODE WITH COMPILED ERRORS

\* DO TEST YOUR CODE USING MAIN METHOD

\* CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC

\* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF  
NEED BE,

\* YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER  
EXCEPTION CLAUSE

\*

\*\*\*\*\*

\*\*\*\*\*

/

```
package com.cts.creditcard.util;
```

```
import java.util.Date;
```

```
import java.util.List;
```

```
import com.cts.creditcard.exception.CreditCardAdminSystemException;
```

```
public class ApplicationUtil {
```

```
/**
```

```
 * @param fileName
```

```
 * @return List<String>
```

```
 * @throws
```

```
CreditCardAdminSystemException
```

```
 */
```

```
public static List<String> readFile(String  
fileName) throws  
CreditCardAdminSystemException {  
  
    // TODO Add your logic here  
  
    return null; // TODO change this  
    return value  
}  
  
public static Date  
geDateWithoutTime(Date date) {  
  
    // TODO Add your logic here  
  
    return null; // TODO change this  
    return value  
}  
  
}  
  
/**  
 * @param util  
 *      Date  
 * @return sql Date  
 */  
public static java.sql.Date  
convertUtilToSqlDate(java.util.Date  
uDate) {  
  
    // TODO Add your logic here  
  
    return null; // TODO change this  
    return value  
}  
  
/**  
 * @param inDate  
 * @return Date
```

```
        */  
  
        public static Date  
        convertStringToDate(String inDate) {  
  
            // TODO Add your logic here  
  
            return null; // TODO change this  
            return value  
  
        }  
  
    }  
  
}
```

## DBConnectionManager

```
/*****************************************************************************  
*****  
* This class DBConnectionManager is used acquire database connection  
*  
* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS,  
* EXCEPTION CLAUSES, RETURN TYPES  
* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE  
* DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS  
* DO TEST YOUR CODE USING MAIN METHOD  
* CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC  
* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF  
* NEED BE,  
* YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER  
* EXCEPTION CLAUSE  
*  
*****  
*****  
/
```

```
package com.cts.creditcard.util;
```

```
import com.cts.creditcard.exception.CreditCardAdminSystemException;

public class DBConnectionManager {

    /**
     * @throws
     CreditCardAdminSystemException
     */

    private DBConnectionManager() throws
    CreditCardAdminSystemException {

    }

    /**
     * @return DBConnectionManager
     * @throws
     CreditCardAdminSystemException
     */

    public static DBConnectionManager
    getInstance() throws
    CreditCardAdminSystemException {
        // TODO Add your logic here

        return null; // TODO change this
        return value
    }

}

CreditCard
```

```
*****  
*****
```

\* This class CreditCard is a value object for data transfer between Service and DAO layers

\*

\* DO NOT CHANGE THE NAMES OR DATA TYPES OR VISIBILITY OF THE BELOW MEMBER  
VARIABLES

\* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS,  
EXCEPTION CLAUSES, RETURN TYPES

\* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE

\* DO NOT SUBMIT THE CODE WITH COMPILE ERRORS

\* DO TEST YOUR CODE USING MAIN METHOD

\* CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC

\* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF  
NEED BE,

\* YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER  
EXCEPTION CLAUSE

\*

```
*****  
*****
```

/

```
package com.cts.creditcard.vo;
```

```
import java.util.Date;
```

```
public class CreditCard {
```

```
//DO NOT CHANGE THE NAMES OR  
DATA TYPES OR VISIBILITY OF THE  
BELOW MEMBER VARIABLES
```

```
    public Long creditCardNum;
```

```
    public String customerName;
```

```
        public String customerEmail;  
        public Long customerPhone;  
        public Double billAmount;  
        public Date dueDate;  
        public Date paymentDate;  
  
        //TODO add your code here  
  
    }
```

## **CreditCardAdminSystem**

### **CreditCardDAO**

```
/*****************************************************************************  
 * This class CreditCardDAO is used to persist or retrieve data from database.  
 *  
 * DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS,  
 * EXCEPTION CLAUSES, RETURN TYPES  
 *  
 * YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE  
 *  
 * DO NOT SUBMIT THE CODE WITH COMPILED ERRORS  
 *  
 * DO TEST YOUR CODE USING MAIN METHOD  
 *  
 * CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC  
 *  
 * DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF  
 * NEED BE,  
 *  
 * YOU CAN CATCH THEM AND THROW ONLY  
 *  
 * THE APPLICATION SPECIFIC EXCEPTION AS PER EXCEPTION CLAUSE  
 *
```

```
*****
*****
/
package com.cts.creditcard.dao;

import java.sql.Connection;
import java.util.List;

import com.cts.creditcard.exception.CreditCardAdminSystemException;
import com.cts.creditcard.vo.CreditCard;

public class CreditCardDAO {

    private static Connection conn = null;

    public Boolean
        addCreditCardDetails(List<CreditCard>
    cards) throws
        CreditCardAdminSystemException {
        //TODO add your code here
        return false; //TODO CHANGE
        THIS RETURN TYPE
    }

}
```

## CreditCardAdminSystemException

```
*****
*****
```

- \* This class CreditCardAdminSystemException is as exception class.
- \*
- \* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS,  
EXCEPTION CLAUSES, RETURN TYPES
- \* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
- \* DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS
- \* DO TEST YOUR CODE USING MAIN METHOD
- \* CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC
- \* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF  
NEED BE,

- \* YOU CAN CATCH THEM AND THROW ONLY
- \* THE APPLICATION SPECIFIC EXCEPTION AS PER EXCEPTION CLAUSE
- \*

```
*****  
*****  
/
```

```
package com.cts.creditcard.exception;
```

```
public class CreditCardAdminSystemException extends Exception {
```

```
    private static final long serialVersionUID  
    = -6349759544203601561L;
```

```
    //TODO add your constructors here
```

```
}
```

## MainApp

```
*****  
*****
```

- \* This class MainApp is used to run the service methods and to test the database.

```
*
```

```
* YOU CAN INVOKE THE METHODS AS REQUIRED FROM HERE TO TEST THE APP
```

```
* DO NOT USE VIA COMMAND LINE ARGUMENTS FROM MAIN METHOD, UNLESS SPECIFIED
```

```
*
```

```
*****
```

```
*****
```

```
/
```

```
package com.cts.creditcard.main;
```

```
public class MainApp {
```

```
    public static void main(String ag[]) {
```

```
        //TODO add your code here
```

```
    }
```

```
}
```

### CreditCardAdminService

```
*****
```

```
*****
```

```
*
```

```
* This class CreditCardAdminService is used to handle business logic for the proposed
```

```
    system.
```

```
*
```

```
*
```

```
* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS,
```

```
    EXCEPTION CLAUSES, RETURN TYPES
```

```
*
```

```
* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
```

```
*
```

```
* DO NOT SUBMIT THE CODE WITH COMPILE ERRORS
```

```
*
```

```
* DO TEST YOUR CODE USING MAIN METHOD
```

```
*
```

```
* CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC
```

```
*
```

```
* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF
```

```
    NEED BE,
```

```
*
```

```
* YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER
```

```
    EXCEPTION CLAUSE
```

```
*  
*****  
*****  
/  
  
package com.cts.creditcard.service;  
  
import java.util.List;  
  
import com.cts.creditcard.exception.CreditCardAdminSystemException;  
import com.cts.creditcard.vo.CreditCard;  
  
public class CreditCardAdminService {  
  
    /**  
     * @param records  
     * @return List<Customer>  
     */  
    public static List<CreditCard>  
    buildMasterCreditCardList(List<String>  
    records) {  
        // TODO Add your logic here  
  
        return null; // TODO change this  
        return value  
    }  
  
    /**  
     * @param billAmount  
     * @return Double  
     */
```

```

        public static Double
        getBillAmountWithLatePaymentCharges
        (Double billAmount) {

            // TODO add your logic here

            return 0.00; // TODO change this
            return value

        }

        /**
         * @param inputFeed
         * @return Boolean
         * @throws
         CreditCardAdminSystemException
         */

        public Boolean
        addCreditCardDetails(String inputFeed)
        throws
        CreditCardAdminSystemException {

            // TODO add your logic here

            return null; //TODO change this
            return value

        }

    }

```

## ApplicationUtil

```

*****
*****  

* This class ApplicationUtil is used for any utility methods needed for service or dao classes
*

```

- \* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS,  
EXCEPTION CLAUSES, RETURN TYPES
- \* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
- \* DO NOT SUBMIT THE CODE WITH COMPILED ERRORS
- \* DO TEST YOUR CODE USING MAIN METHOD
- \* CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC
- \* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF  
NEED BE,
- \* YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER  
EXCEPTION CLAUSE

\*

```
*****
***** /
```

```
package com.cts.creditcard.util;

import java.util.Date;
import java.util.List;

import com.cts.creditcard.exception.CreditCardAdminSystemException;

public class ApplicationUtil {

    /**
     * @param fileName
     * @return List<String>
     * @throws CreditCardAdminSystemException
     */
    public static List<String> readFile(String
fileName) throws
CreditCardAdminSystemException {
```

```
// TODO Add your logic here

        return null; // TODO change this
return value
}

public static Date
geDateWithoutTime(Date date) {
    // TODO Add your logic here
    return null; // TODO change this
return value
}

}

/***
 * @param util
 *      Date
 * @return sql Date
 */
public static java.sql.Date
convertUtilToSqlDate(java.util.Date
uDate) {
    // TODO Add your logic here
    return null; // TODO change this
return value
}

/***
 * @param inDate
 * @return Date
 */
public static Date
convertStringToDate(String inDate) {
```

```
// TODO Add your logic here
        return null; // TODO change this
        return value
    }

}

/*****
*****
* This class DBConnectionManager is used acquire database connection
*
* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS,
* EXCEPTION CLAUSES, RETURN TYPES
*
* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
*
* DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS
*
* DO TEST YOUR CODE USING MAIN METHOD
*
* CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC
*
* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF
* NEED BE,
*
* YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER
* EXCEPTION CLAUSE
*
*****
*/
package com.cts.creditcard.util;

import com.cts.creditcard.exception.CreditCardAdminSystemException;

public class DBConnectionManager {
```

```

    /**
     * @throws
     CreditCardAdminSystemException
    */

    private DBConnectionManager() throws
    CreditCardAdminSystemException {

}

    /**
     * @return DBConnectionManager
     * @throws
     CreditCardAdminSystemException
    */

    public static DBConnectionManager
getInstance() throws
CreditCardAdminSystemException {

    // TODO Add your logic here

        return null; // TODO change this
        return value
    }

}

```

## CreditCard

```

*****
*****
* This class CreditCard is a value object for data transfer between Service and DAO layers
*

```

- \* DO NOT CHANGE THE NAMES OR DATA TYPES OR VISIBILITY OF THE BELOW MEMBER VARIABLES
- \* DO NOT CHANGE THE CLASS NAME, PUBLIC METHODS, SIGNATURE OF METHODS, EXCEPTION CLAUSES, RETURN TYPES
- \* YOU CAN ADD ANY NUMBER OF PRIVATE METHODS TO MODULARIZE THE CODE
- \* DO NOT SUBMIT THE CODE WITH COMPILATION ERRORS
- \* DO TEST YOUR CODE USING MAIN METHOD
- \* CHANGE THE RETURN TYPE OF THE METHODS ONCE YOU BUILT THE LOGIC
- \* DO NOT ADD ANY ADDITIONAL EXCEPTIONS IN THE THROWS CLAUSE OF THE METHOD. IF NEED BE,
- \* YOU CAN CATCH THEM AND THROW ONLY THE APPLICATION SPECIFIC EXCEPTION AS PER EXCEPTION CLAUSE

\*

```
*****  
*****  
/
```

```
package com.cts.creditcard.vo;
```

```
import java.util.Date;
```

```
public class CreditCard {
```

```
//DO NOT CHANGE THE NAMES OR  
DATA TYPES OR VISIBILITY OF THE  
BELOW MEMBER VARIABLES
```

```
    public Long creditCardNum;  
    public String customerName;  
    public String customerEmail;  
    public Long customerPhone;  
    public Double billAmount;  
    public Date dueDate;  
    public Date paymentDate;
```

```
//TODO add your code here  
}
```

UNOAdmission\_

StudentAdmissionDAO

```
package com.cts.unoadm.dao;
```

```
import java.util.ArrayList;
```

```
import java.util.List;
```

```
import com.cts.unoadm.exception.StudentAdmissionException;
```

```
import com.cts.unoadm.vo.StudentAdmission;
```

```
public class StudentAdmissionDAO {
```

```
    public boolean  
    addStudentAdmissionDetails(List<Stude  
ntAdmission> stdAdmissions) throws  
StudentAdmissionException {
```

```
        boolean recordsAdded = false;
```

```
        //code here
```

```
        return recordsAdded;
```

```
}
```

```
public List<StudentAdmission>
getAllStudentAdmissionDetails() throws
StudentAdmissionException {

    List<StudentAdmission>
    stdAdmissions = new
    ArrayList<StudentAdmission>();

    //code here

    return stdAdmissions;

}

}
```

## **StudentAdmissionException**

```
package com.cts.unoadm.exception;

public class StudentAdmissionException extends Exception {

    private static final long serialVersionUID
    = -1105431869622052445L;

    /**
     * @param message
     * @param cause
     */
    public
    StudentAdmissionException(String
    message, Throwable cause) {
```

```
        super(message, cause);

    }

}

MainApp

package com.cts.unoadm.main;

import com.cts.unoadm.skeletonvalidator.SkeletonValidator;

public class MainApp {

    public static void main(String[] args) {
        //Don't delete this code
        //Skeletonvalidaton starts
        new SkeletonValidator();
        //Skeletonvalidation ends

        //Write your code here..

    }
}
```

## **StudentAdmissionService**

```
package com.cts.unoadm.service;
```

```
import java.util.ArrayList;
import java.util.List;

import com.cts.unoadm.exception.StudentAdmissionException;
import com.cts.unoadm.vo.StudentAdmission;

public class StudentAdmissionService {

    /**
     * @return List<StudentAdmission>
     */
    public static List<StudentAdmission>
    buildStudentAdmissionsList(List<String>
    studentAdmissionRecords) {
        List<StudentAdmission>
        studentAdmissionList = new
        ArrayList<StudentAdmission>();

        //Code here

        return studentAdmissionList;
    }

    public boolean
    addStudentAdmissionDetails(String
    inputFeed) throws
    StudentAdmissionException {
        //Code here

        return false;
    }
}
```

```

        }

public static double[]
calculateTotalCollegeFee(String
preferCollegeHostel, String
firstGraduate, String departmentName)
{
    double[] studentAdmissionCosts
= new double[4];

//Code here..

return studentAdmissionCosts;
}

public boolean
searchStudentAdmission(String
admissionId) throws
StudentAdmissionException {
    boolean status = false;

//Code here..

return status;
}
}

```

## SkeletonValidator

```
package com.cts.unoadm.skeletonvalidator;
```

```
import java.lang.reflect.Array;
import java.lang.reflect.Method;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * @author t-aarti3
 *
 *      This class is used to verify if the Code Skeleton is intact and not
 * modified by participants thereby ensuring smooth auto evaluation
 */

public class SkeletonValidator {
    public SkeletonValidator() {
        validateClassName("com.cts.uno
adm.util.DBConnectionManager");

        validateClassName("com.cts.uno
adm.util.ApplicationUtil");

        validateClassName("com.cts.uno
adm.service.StudentAdmissionService");

        validateClassName("com.cts.uno
adm.dao.StudentAdmissionDAO");

        validateClassName("com.cts.uno
adm.vo.StudentAdmission");

        validateClassName("com.cts.uno
adm.exception.StudentAdmissionExcept
ion");
    }
}
```

```
 validateMethodSignature(
```

```
    "addStudentAdmissionDetails:boolean,getAllStudentAdmissionDetails>List",
```

```
    "com.cts.unoadm.dao.StudentAdmissionDAO");
```

```
 validateMethodSignature(
```

```
    "buildStudentAdmissionsList>List,addStudentAdmissionDetails:boolean,calculateTotalCollegeFee:double[],searchStudentAdmission:boolean",
```

```
    "com.cts.unoadm.service.StudentAdmissionService");
```

```
 validateMethodSignature(
```

```
    "readFile>List,convertUtilToSqlDate:Date,convertStringToDate:Date,checkIfValidAdmission:boolean",
```

```
    "com.cts.unoadm.util.ApplicationUtil");
```

```
 validateMethodSignature(
```

```
    "getConnection:Connection,getInstance:DBConnectionManager",
```

```
    "com.cts.unoadm.util.DBConnectionManager");
```

```
}
```

```
private static final Logger LOG =
Logger.getLogger("SkeletonValidator");

protected final boolean
validateClassName(String className) {

    boolean iscorrect = false;

    try {

        Class.forName(className);

        iscorrect = true;

        LOG.info("Class Name " +
className + " is correct");

    } catch (ClassNotFoundException e) {

        LOG.log(Level.SEVERE,
"You have changed either the " + "class
name/package. Use the correct package
"

        + "and

class name as provided in the
skeleton");

    }

    } catch (Exception e) {

        LOG.log(Level.SEVERE,
"There is
an error in validating the " + "Class
Name. Please manually verify that the "

        + "Class name is same as
skeleton before uploading");

    }

    return iscorrect;
}
```

```
protected final void
validateMethodSignature(String
methodWithExcptn, String className) {

    Class cls = null;

    try {

        String[] actualmethods =
methodWithExcptn.split(",");
        boolean errorFlag = false;
        String[] methodSignature;
        String methodName =
null;
        String returnType = null;

        for (String singleMethod :
actualmethods) {
            boolean
foundMethod = false;
            methodSignature
= singleMethod.split(":");
            methodName =
methodSignature[0];
            returnType =
methodSignature[1];
            cls =
Class.forName(className);
            Method[]
methods = cls.getMethods();
            for (Method
findMethod : methods) {
```

```
        if  
(methodName.equals(findMethod.getN  
ame())) {
```

```
            foundMethod = true;
```

```
        if  
(!(findMethod.getReturnType().getSimpl  
eName().equals(returnType))) {
```

```
            errorFlag = true;
```

```
            LOG.log(Level.SEVERE, " You  
have changed the " + "return type in " +  
methodName
```

```
            + " method.  
Please stick to the " + "skeleton  
provided");
```

```
    }  
else {
```

```
    LOG.info("Method signature of "  
+ methodName + " is valid");
```

```
}
```

```
}
```

```
if (!foundMethod)  
{
```

```
    errorFlag =  
true;
```

```
    LOG.log(Level.SEVERE, " Unable  
to find the given public method " +  
methodName
```

```
        + ". Do not change the " + "given  
        public method name. " + "Verify it with  
        the skeleton");  
  
    }  
  
}  
  
if (!errorFlag) {  
  
    LOG.info("Method  
signature is valid");  
  
}  
  
}  
  
} catch (Exception e) {  
  
    LOG.log(Level.SEVERE,  
  
            " There is  
            an error in validating the " + "method  
            structure. Please manually verify that  
            the "  
  
            + "Method signature is same as  
            the skeleton before uploading");  
  
}  
  
}  
  
}
```

## ApplicationUtil

```
package com.cts.unoadm.util;
```

```
import java.util.ArrayList;
```

```
import java.util.Date;
import java.util.List;

import com.cts.unoadm.exception.StudentAdmissionException;

public class ApplicationUtil {

    /**
     * @param fileName
     * @return List<String>
     * @throws StudentAdmissionException
     */
    public static List<String> readFile(String
fileName) throws
StudentAdmissionException {
        List<String>
studentAdmissionList = new
ArrayList<String>();
        //Code here..

        return studentAdmissionList;
    }

    /**
     * @param util
     *      Date
     * @return sql Date
     */
    public static java.sql.Date
convertUtilToSqlDate(java.util.Date
uDate) {
```

```
java.sql.Date sDate = null;
```

```
//Code here..
```

```
return sDate;
```

```
}
```

```
/**
```

```
* @param inDate
```

```
* @return Date
```

```
*/
```

```
public static Date
```

```
convertStringToDate(String inDate) {
```

```
//Code here..
```

```
return new Date();//TODO
```

```
change this return value
```

```
}
```

```
public static boolean
```

```
checkIfValidAdmission(Date
```

```
dtOfCounseling, Date dtOfAdmission,
```

```
String manager) {
```

```
boolean admissionValidity =
```

```
false;
```

```
//Code here..
```

```
return admissionValidity;
```

```
    }  
}
```

## DBConnectionManager

```
/**  
 * Don't change this code  
 */  
  
package com.cts.unoadm.util;  
  
import java.io.FileInputStream;  
  
import java.io.FileNotFoundException;  
  
import java.io.IOException;  
  
import java.sql.Connection;  
  
import java.sql.DriverManager;  
  
import java.sql.SQLException;  
  
import java.util.Properties;  
  
  
import com.cts.unoadm.exception.StudentAdmissionException;  
  
  
  
  
  
  
public class DBConnectionManager {  
  
    public static final String PROPERTY_FILE  
        = "database.properties";  
  
    public static final String DRIVER =  
        "drivername";  
  
    public static final String URL = "url";  
  
    public static final String USER_NAME =  
        "username";
```

```
public static final String PASSWORD =
"password";

private static Connection connection =
null;

private static Properties props = null;

/** 
 * @throws StudentAdmissionException
 */
private DBConnectionManager() throws
StudentAdmissionException {
    loadProperties();
    try {

        Class.forName(props.getProperty
(DRIVER));
        this.connection =
DriverManager.getConnection(props.get
Property(URL),
props.getProperty(USER_NAME),

        props.getProperty(PASSWORD));
    } catch (ClassNotFoundException
ex) {

        throw new
StudentAdmissionException("Could not
find Driver class ", ex.getCause());
    } catch (SQLException e) {

        throw new
StudentAdmissionException("Database
Connection Creation Failed",
e.getCause());
    }
}
```

```
}
```

```
/**
```

```
* @return Connection
```

```
*/
```

```
public Connection getConnection() {
```

```
    return connection;
```

```
}
```

```
/**
```

```
* @return DBConnectionManager
```

```
* @throws StudentAdmissionException
```

```
*/
```

```
public static DBConnectionManager
```

```
getInstance() throws
```

```
StudentAdmissionException {
```

```
// Code here
```

```
    return null;
```

```
}
```

```
/**
```

```
* @throws StudentAdmissionException
```

```
*/
```

```
private void loadProperties() throws
```

```
StudentAdmissionException {
```

```
    FileInputStream inputStream =  
    null;
```

```
    try {
```

```

        inputStream = new
InputStream(PROPERTY_FILE);

        props = new Properties();
        props.load(inputStream);

    } catch (FileNotFoundException
e) {

        throw new
StudentAdmissionException("Database
Property File Not Found", e.getCause());

    } catch (IOException e) {

        throw new
StudentAdmissionException("Exception
during property file I/O", e.getCause());

    } finally {

        if (inputStream != null) {

            try {

                inputStream.close();

            } catch
(IOException e) {

                throw new
StudentAdmissionException("Exception
during property file I/O", e.getCause());

            }
        }
    }
}

```

## StudentAdmission

```
/*
 * Don't change this code
 */

package com.cts.unoadm.vo;

import java.util.Date;

public class StudentAdmission {

    String admissionId;
    String studentCode;
    Date dateOfCounseling;
    String departmentName;
    Date dateOfAdmission;
    String preferCollegeHostel;
    String firstGraduate;
    String managerApproval;
    double admissionFee;
    double tuitionFee;
    double hostelFee;
    double totalCollegeFee;
    String finalStatusOfAdmission;

    public StudentAdmission() {
        super();
    }

    public StudentAdmission(String
        admissionId, String studentCode, Date
```

```
        dateOfCounseling, String  
        departmentName,  
  
        Date dateOfAdmission,  
        String preferCollegeHostel, String  
        firstGraduate, String managerApproval,  
  
        double admissionFee,  
        double tuitionFee, double hostelFee,  
        double totalCollegeFee,  
  
        String  
        finalStatusOfAdmission) {  
  
        super();  
  
        this.admissionId = admissionId;  
  
        this.studentCode = studentCode;  
  
        this.dateOfCounseling =  
        dateOfCounseling;  
  
        this.departmentName =  
        departmentName;  
  
        this.dateOfAdmission =  
        dateOfAdmission;  
  
        this.preferCollegeHostel =  
        preferCollegeHostel;  
  
        this.firstGraduate =  
        firstGraduate;  
  
        this.managerApproval =  
        managerApproval;  
  
        this.admissionFee =  
        admissionFee;  
  
        this.tuitionFee = tuitionFee;  
  
        this.hostelFee = hostelFee;  
  
        this.totalCollegeFee =  
        totalCollegeFee;  
  
        this.finalStatusOfAdmission =  
        finalStatusOfAdmission;  
  
    }  

```

```
public String getAdmissionId() {  
    return admissionId;  
}  
  
public void setAdmissionId(String admissionId) {  
    this.admissionId = admissionId;  
}  
  
public String getStudentCode() {  
    return studentCode;  
}  
  
public void setStudentCode(String studentCode) {  
    this.studentCode = studentCode;  
}  
  
public Date getDateOfCounseling() {  
    return dateOfCounseling;  
}  
  
public void setDateOfCounseling(Date dateOfCounseling) {  
    this.dateOfCounseling = dateOfCounseling;  
}  
  
public String getDepartmentName() {  
    return departmentName;
```

```
}
```

```
public void setDepartmentName(String  
departmentName) {
```

```
    this.departmentName =  
    departmentName;
```

```
}
```

```
public Date getDateOfAdmission() {
```

```
    return dateOfAdmission;
```

```
}
```

```
public void setDateOfAdmission(Date  
dateOfAdmission) {
```

```
    this.dateOfAdmission =  
    dateOfAdmission;
```

```
}
```

```
public String getPreferCollegeHostel() {
```

```
    return preferCollegeHostel;
```

```
}
```

```
public void  
setPreferCollegeHostel(String  
preferCollegeHostel) {
```

```
    this.preferCollegeHostel =  
    preferCollegeHostel;
```

```
}
```

```
public String getFirstGraduate() {
```

```
    return firstGraduate;
```

```
}
```

```
public void setFirstGraduate(String  
firstGraduate) {
```

```
    this.firstGraduate =  
    firstGraduate;  
}
```

```
public String getManagerApproval() {  
  
    return managerApproval;  
}
```

```
public void setManagerApproval(String  
managerApproval) {  
  
    this.managerApproval =  
    managerApproval;  
}
```

```
public double getAdmissionFee() {  
  
    return admissionFee;  
}
```

```
public void setAdmissionFee(double  
admissionFee) {  
  
    this.admissionFee =  
    admissionFee;  
}
```

```
public double getTuitionFee() {  
  
    return tuitionFee;  
}
```

```
public void setTuitionFee(double  
tuitionFee) {
```

```
    this.tuitionFee = tuitionFee;
```

```
}
```

```
public double getHostelFee() {
```

```
    return hostelFee;
```

```
}
```

```
public void setHostelFee(double  
hostelFee) {
```

```
    this.hostelFee = hostelFee;
```

```
}
```

```
public double getTotalCollegeFee() {
```

```
    return totalCollegeFee;
```

```
}
```

```
public void setTotalCollegeFee(double  
totalCollegeFee) {
```

```
    this.totalCollegeFee =  
    totalCollegeFee;
```

```
}
```

```
public String
```

```
getFinalStatusOfAdmission() {
```

```
    return finalStatusOfAdmission;
```

```
}
```

```
public void
```

```
setFinalStatusOfAdmission(String  
finalStatusOfAdmission) {
```

```

        this.finalStatusOfAdmission =
finalStatusOfAdmission;

    }

@Override
public String toString() {
    return "Student Admission
Details: [admissionId=" + admissionId +
", studentCode=" + studentCode + ",
dateOfCounseling="

+
dateOfCounseling +",
departmentName=" + departmentName +
", dateOfAdmission=" +
dateOfAdmission +",
preferCollegeHostel="

+
preferCollegeHostel +", firstGraduate=" +
firstGraduate +", managerApproval=" +
managerApproval

+",
admissionFee=" + admissionFee +",
tuitionFee=" + tuitionFee +",
hostelFee=" + hostelFee +",
totalCollegeFee=" + totalCollegeFee

+",
finalStatusOfAdmission=" +
finalStatusOfAdmission +"]";
}

}

```

## **ConstructionCostTimeEstimate**

## **CostAndTimeEstimation**

```
package com.cts.conctes.client;

import com.cts.conctes.service.ConstructionProjectEstimationService;

public class CostAndTimeEstimation {

    public static void main(String[] args)
    {
        ConstructionProjectEstimationService cpeService = new
        ConstructionProjectEstimationService();
        //WRITE YOUR CODE HERE

    }
}
```

## **CostAndTimeEstDAO**

```
package com.cts.conctes.dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Date;
```

```
import com.cts.conctes.exception.ConstructionEstimationException;
import com.cts.conctes.model.ConstructionProject;
import com.cts.conctes.util.ApplicationUtil;

public class CostAndTimeEstDAO {

    public static Connection connection =
        null;

    public boolean
    insertConstructionProject(ArrayList
    <ConstructionProject> constProjects)
    throws
    ConstructionEstimationException {
        boolean recordsAdded = false;

        //WRITE YOUR CODE HERE

        return recordsAdded;
    }

    public ArrayList <ConstructionProject>
    getConstructionProjectsData()
    {

        ArrayList <ConstructionProject>
        consApplicants = new
        ArrayList<ConstructionProject>();

        //WRITE YOUR CODE HERE

        return consApplicants;
    }
}
```

```
    }
```

```
}
```

## **DBConnectionManager**

```
package com.cts.conctes.dao;
```

```
import java.io.FileInputStream;  
import java.io.FileNotFoundException;  
import java.io.IOException;  
import java.sql.Connection;  
import java.sql.DriverManager;  
import java.sql.SQLException;  
import java.util.Properties;
```

```
import com.cts.conctes.exception.ConstructionEstimationException;
```

```
public class DBConnectionManager {
```

```
    private static Connection con = null;
```

```
    private static DBConnectionManager  
    instance;
```

```
    public DBConnectionManager() throws  
    ConstructionEstimationException
```

```
{
```

```
//WRITE YOUR CODE HERE  
//return con;  
}  
  
public static DBConnectionManager  
getInstance() throws  
ConstructionEstimationException  
{  
  
//WRITE YOUR CODE HERE  
return instance;  
}  
  
public Connection getConnection()  
{  
  
//WRITE YOUR CODE HERE  
return con;  
}  
}
```

## ConstructionEstimationException

```
package com.cts.conctes.exception;  
  
public class ConstructionEstimationException extends Exception{  
  
String strMsg1;  
Throwable strMsg2;
```

```
        public  
        ConstructionEstimationException() {  
            super();  
        }  
  
    }  
  
}
```

## ConstructionProject

```
package com.cts.conctes.model;  
  
import java.util.Date;  
  
public class ConstructionProject {  
  
    String projectId;  
    Date plannedDOStart;  
    String typeOfProject;  
    String structure;  
    double areaInSqFt;  
    double estimatedCostInlac;  
    double estimatedTimeInMonths;  
  
    public ConstructionProject() {  
        super();  
    }  
}
```

```
public ConstructionProject(String  
projectId, Date plannedDOStart, String  
typeOfProject, String structure,
```

```
        double areaInSqFt,  
        double estimatedCostInLac, double  
        estimatedTimeInMonths) {
```

```
    super();
```

```
    this.projectId = projectId;
```

```
    this.plannedDOStart =  
    plannedDOStart;
```

```
    this.typeOfProject =  
    typeOfProject;
```

```
    this.structure = structure;
```

```
    this.areaInSqFt = areaInSqFt;
```

```
    this.estimatedCostInLac =  
    estimatedCostInLac;
```

```
    this.estimatedTimeInMonths =  
    estimatedTimeInMonths;
```

```
}
```

```
public String getProjectId() {
```

```
    return projectId;
```

```
}
```

```
public void setProjectId(String projectId)  
{
```

```
    this.projectId = projectId;
```

```
}
```

```
public Date getPlannedDOStart() {
```

```
    return plannedDOStart;
```

```
}
```

```
public void setPlannedDOStart(Date  
plannedDOStart) {
```

```
    this.plannedDOStart =
```

```
    plannedDOStart;
```

```
}
```

```
public String getTypeOfProject() {
```

```
    return typeOfProject;
```

```
}
```

```
public void setTypeOfProject(String  
typeOfProject) {
```

```
    this.typeOfProject =
```

```
    typeOfProject;
```

```
}
```

```
public String getStructure() {
```

```
    return structure;
```

```
}
```

```
public void setStructure(String structure)
```

```
{
```

```
    this.structure = structure;
```

```
}
```

```
public double getAreaInSqFt() {
```

```
    return areaInSqFt;
```

```
}
```

```
public void setAreaInSqFt(double  
areaInSqFt) {  
    this.areaInSqFt = areaInSqFt;  
}  
  
public double getEstimatedCostInLac() {  
    return estimatedCostInLac;  
}  
  
public void  
setEstimatedCostInLac(double  
estimatedCostInLac) {  
    this.estimatedCostInLac =  
estimatedCostInLac;  
}  
  
public double  
getEstimatedTimeInMonths() {  
    return estimatedTimeInMonths;  
}  
  
public void  
setEstimatedTimeInMonths(double  
estimatedTimeInMonths) {  
    this.estimatedTimeInMonths =  
estimatedTimeInMonths;  
}  
  
@Override  
public String toString() {  
    return "ConstructionProject  
[projectId=" + projectId + ",  

```

```

        plannedDOStart=" + plannedDOStart + ",
        typeOfProject="

                + typeOfProject +
        ", structure=" + structure + ",
        areaInSqFt=" + areaInSqFt + ",
        estimatedCostInLac="

                +
        estimatedCostInLac + ",
        estimatedTimeInMonths=" +
        estimatedTimeInMonths + "]";

    }

}

```

## ConstructionProjectEstimationService

```

package com.cts.conctes.service;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;

import com.cts.conctes.dao.CostAndTimeEstDAO;
import com.cts.conctes.exception.ConstructionEstimationException;
import com.cts.conctes.model.ConstructionProject;
import com.cts.conctes.util.ApplicationUtil;

```

```
public class ConstructionProjectEstimationService {  
  
    public static ArrayList  
    <ConstructionProject>  
    buildConstructionProjectList(List  
    <String> consProjectRecords) {  
  
        final String COMMADELIMITER =  
        ",";  
        ArrayList <ConstructionProject>  
        consProjectRecordList = new  
        ArrayList<ConstructionProject>();  
  
        //WRITE YOUR CODE HERE  
  
        return consProjectRecordList;  
    }  
  
    public boolean  
    addConstructionProjectDetails(String  
    inputFeed) throws  
    ConstructionEstimationException {  
  
        //WRITE YOUR CODE HERE  
        return false;  
    }  
  
    public static double[]  
    estimateTimeAndCostForConstruction(S
```

```
tring projectType,String  
structure,double areaInSqFt)  
{  
  
    double  
costEstimateInRs=0.0,timeEstimateInMo  
nths=0.0;  
  
    double costs[] =  
{costEstimateInRs,timeEstimateInMonth  
s};  
  
/*  
 * The Cost Estimate and  
 *  
 */
```

Based on the type of the Project & the Structure , according to the required

area of Construction, the cost & time have to be calculated based on the base

data available in the table provided in the use case document:

For eg. If the Project Type is “Commercial” and the structure

is “Shopping Complex” the cost incurred for the construction of

per sq. ft is Rs.2600 and the time taken for the construction of

the 1000 sq ft of the same project is 0.23 Months,

calculation has to be performed on the similar basis

i.e Pro rata basis depending upon the type and the area of construction.

\*/

```
//WRITE YOUR CODE HERE

    return costs;

}

}
```

## **SkeletonValidator**

```
package com.cts.conctes.skeleton;

import java.lang.reflect.Method;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

import com.cts.conctes.model.ConstructionProject;

/**
 * @author 222805
 *
 * This class is used to verify if the Code Skeleton is intact and not modified by participants
 * thereby ensuring smooth auto
 * evaluation

```

```
*  
*/  
public class SkeletonValidator {  
  
    public SkeletonValidator() {  
  
        validateClassName("com.cts.con  
ctes.model.ConstructionProject");  
  
        validateClassName("com.cts.con  
ctes.dao.CostAndTimeEstDAO");  
  
        validateClassName("com.cts.con  
ctes.dao.DBConnectionManager");  
  
        validateClassName("com.cts.con  
ctes.exception.ConstructionEstimationE  
xception");  
  
        validateClassName("com.cts.con  
ctes.service.ConstructionProjectEstimati  
onService");  
  
        validateClassName("com.cts.con  
ctes.util.ApplicationUtil");  
  
        validateMethodSignature("insert  
ConstructionProject:boolean","com.cts.c  
ontctes.dao.CostAndTimeEstDAO");  
  
        validateMethodSignature("getIns  
tance:DBConnectionManager","com.cts.  
conctes.dao.DBConnectionManager");  
  
        validateMethodSignature("getCo
```

```
nnection:Connection","com.cts.conctes.  
dao.DBConnectionManager");
```

```
validateMethodSignature("build  
ConstructionProjectList:ArrayList,addCo  
nstructionProjectDetails:boolean,estima  
teTimeAndCostForConstruction:double[]]  
","com.cts.conctes.service.Construction  
ProjectEstimationService");
```

```
}
```

```
private static final Logger LOG =  
Logger.getLogger("SkeletonValidator");
```

```
protected final boolean  
validateClassName(String className) {
```

```
boolean iscorrect = false;
```

```
try {
```

```
Class.forName(className);
```

```
incorrect = true;
```

```
LOG.info("Class Name " +  
className + " is correct");
```

```
} catch (ClassNotFoundException  
e) {
```

```
LOG.log(Level.SEVERE,  
"You have changed either the " + "class  
name/package. Use the correct package  
"
```

```
+ "and  
class name as provided in the  
skeleton");  
  
} catch (Exception e) {  
  
    LOG.log(Level.SEVERE,  
            "There is  
an error in validating the " + "Class  
Name. Please manually verify that the "  
  
        + "Class name is same as  
skeleton before uploading");  
}  
  
return iscorrect;  
  
}  
  
protected final void  
validateMethodSignature(String  
methodWithExcptn, String className) {  
  
    Class cls = null;  
    try {  
  
        String[] actualmethods =  
methodWithExcptn.split(",");  
        boolean errorFlag = false;  
        String[] methodSignature;  
        String methodName =  
null;  
        String returnType = null;  
  
        for (String singleMethod :  
actualmethods) {
```

```
        boolean
foundMethod = false;

        methodSignature
= singleMethod.split ":";

        methodName =
methodSignature[0];

        returnType =
methodSignature[1];

        cls =
Class.forName(className);

        Method[]
methods = cls.getMethods();

        for (Method
findMethod : methods) {

            if
(methodName.equals(findMethod.getName())) {

                foundMethod = true;

                if
(!findMethod.getReturnType().getSimpleName().equals(returnType))) {

                    errorFlag = true;

                    LOG.log(Level.SEVERE, " You
have changed the " + "return type in " +
methodName

                    + "' method.
Please stick to the " + "skeleton
provided");

                }

            }

        }

    }

} } }
```

```
        LOG.info("Method signature of "  
+ methodName + " is valid");
```

```
}
```

```
}
```

```
}
```

```
if (!foundMethod)
```

```
{
```

```
    errorFlag =  
    true;
```

```
    LOG.log(Level.SEVERE, " Unable  
to find the given public method " +  
methodName
```

```
    + ". Do not change the " + "given  
public method name. " + "Verify it with  
the skeleton");
```

```
}
```

```
}
```

```
if (!errorFlag) {
```

```
    LOG.info("Method  
signature is valid");
```

```
}
```

```
} catch (Exception e) {
```

```
    LOG.log(Level.SEVERE,  
    " There is  
an error in validating the " + "method  
structure. Please manually verify that  
the "
```

```
        + "Method signature is same as  
the skeleton before uploading");  
    }  
}  
}
```

## ApplicationUtil

```
package com.cts.conctes.util;  
  
import java.io.BufferedReader;  
import java.io.FileInputStream;  
import java.io.FileNotFoundException;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
import java.util.StringTokenizer;  
  
import com.cts.conctes.exception.ConstructionEstimationException;  
  
public class ApplicationUtil {
```

```
public static List<String> readFile(String  
inputfeed) throws  
ConstructionEstimationException {  
  
    List<String> constructionProjects  
= new ArrayList<String>();
```

//WRITE YOUR CODE HERE

```
return constructionProjects;
```

```
}
```

```
public static java.sql.Date  
utilToSqlDateConverter(java.util.Date  
utDate) {
```

```
java.sql.Date sqlDate = null;
```

//WRITE YOUR CODE HERE

```
return sqlDate;
```

```
}
```

```
public static java.util.Date  
stringToDateConverter(String  
stringDate) {
```

```
Date strDate = new Date();
```

//WRITE YOUR CODE HERE

```
return strDate;
```

```
}
```

```
public static boolean  
checkIfCurrentFinYearProject(Date dos)
```

```
{  
    boolean flag = false;  
  
    int givenYear,givenMonth;  
  
    givenYear =  
(dos.getYear()+1900);  
  
    givenMonth = dos.getMonth();  
  
    Date curDate = new Date();  
  
    int curYear,curMonth;  
  
    curYear =  
(curDate.getYear()+1900);  
  
    curMonth = curDate.getMonth();  
  
    if( curYear == givenYear)  
    {  
  
        if(((curMonth  
        >=0)&&(curMonth <= 2)) &&  
        ((givenMonth >=0)&&(givenMonth <=  
        2)))  
        {  
  
            flag = true;  
        }  
  
        else if(((curMonth  
        >=3)&&(curMonth <= 11)) &&  
        ((givenMonth >=3)&&(givenMonth <=  
        11)))  
        {  
  
            flag = true;  
        }  
  
        else  
        {  
  
            flag = false;  
        }  
    }  
}
```

```
else if(curYear > givenYear)
```

```
{
```

```
    int dif = curYear -  
    givenYear;
```

```
    if(dif == 1)
```

```
{
```

```
    if(((curMonth  
    >=0)&&(curMonth <= 2)) &&  
    ((givenMonth >=3)&&(givenMonth <=  
    11)))
```

```
{
```

```
    flag = true;
```

```
}
```

```
    else if(((curMonth  
    >=3)&&(curMonth <= 11)) &&  
    ((givenMonth >=3)&&(givenMonth <=  
    11)))
```

```
{
```

```
    flag = false;
```

```
}
```

```
else
```

```
{
```

```
    flag = false;
```

```
}
```

```
}
```

```
else
```

```
{
```

```
    flag = false;
```

```
}
```

```
}
```

```
else if(curYear < givenYear)
```

```
{
```

```
int dif = givenYear-
curYear;

if(dif == 1)

{

    if(((curMonth
>=3)&&(curMonth <= 11)) &&
((givenMonth >=0)&&(givenMonth <=
2)))

{

    flag = true;

}

else if(((curMonth
>=3)&&(curMonth <= 11)) &&
((givenMonth >=3)&&(givenMonth <=
11)))

{

    flag = false;

}

else

{
```

```
        return flag;  
  
    }  
  
}
```

## Insurance

CollectionAgency

```
import java.io.*;  
import java.sql.*;  
import java.util.*;
```

```
public class CollectionAgency
```

```
{
```

```
//write the required business logic  
methods as expected in the question  
description
```

```
public List<Payment>  
generatePaymentAmount(String  
filePath)
```

```
{
```

```
// fill your code here
```

```
}
```

```
public boolean validate(String policyId)
throws InvalidPolicyIdException

{

    // fill your code here


}

public void updatePolicyDetails(List
<Payment> paymentList)

{

    // fill your code here

}

}
```

## **DBHandler**

```
import java.io.*;
import java.sql.*;
import java.util.*;

public class DBHandler {

    //write the required business logic methods as expected in the question description

    public Connection establishConnection()

    {

        // fill your code here


    }

}
```

```
}
```

```
InvalidPolicyIdException
```

```
//make the required changes to this class so that InvalidPolicyIdException is of type  
exception.
```

```
public class InvalidPolicyIdException
```

```
{
```

```
//fill your code here
```

```
}
```

## Main

```
import java.util.*;
```

```
public class Main
```

```
{
```

```
    public static void main(String[] args)
```

```
{
```

```
    // fill your code here
```

```
}
```

```
}
```

```
Payment
```

```
public class Payment
{
    private String policyId;
    private double monthlyPremium;
    private int noOfMonths;
    private double paymentAmount;
    public String getPolicyId() {
        return policyId;
    }
    public void setPolicyId(String policyId) {
        this.policyId = policyId;
    }
    public double getMonthlyPremium() {
        return monthlyPremium;
    }
    public void setMonthlyPremium(double monthlyPremium) {
        this.monthlyPremium =
monthlyPremium;
    }
    public int getNoOfMonths() {
        return noOfMonths;
    }
    public void setNoOfMonths(int noOfMonths) {
        this.noOfMonths = noOfMonths;
    }
    public double getPaymentAmount() {
```

```
        return paymentAmount;  
    }  
  
    public void setPaymentAmount(double  
        paymentAmount) {  
        this.paymentAmount =  
            paymentAmount;  
    }  
  
    //Write the required business logic as  
    //expected in the question description  
    public void calculatePaymentAmount()  
    {  
        //fill your code here  
    }  
  
}
```

1 QuestionText	Choice1	Choice2	Choice3	Choice4
2  What will be the result of compiling the following program? <pre>public class MyClass {     long var;     public void MyClass(long param) { var =         param; } // (Line no 1)     public static void main(String[] args) {         MyClass a, b;         a = new MyClass(); // (Line no 2) □     } }</pre>	A compilation error will occur at (Line no 2), since the class does not have a constructor that takes one argument of type int.	A compilation error will occur at (2), since the class does not have a default constructor.	A compilation error will occur at (Line no 1), since constructors cannot specify a return value.	A compilation error will occur at (Line no 2), since the class does not have a constructor that takes one argument of type int. The program will compile without errors.  String s = "null"; byte b = 256;
3 Which of the following declarations are correct? (Choose TWO)				
4 What will happen when you attempt to compile and run this code? <pre>abstract class Base{     abstract public void myfunc();     public void another(){         System.out.println("Another method");     } }  public class Abs extends Base{     public static void main(String argv[]){         Abs a = new Abs();         a.amethod();     }     public void myfunc(){         System.out.println("My Func");     }     public void amethod(){         myfunc();     } }</pre>	The compiler will complain at run time that the Base class has non abstract methods.	The code will compile but complain at run time that the Base class has non abstract methods.	The compiler will complain at run time that the myfunc in the base class has no body, nobody at all to print it.	The compiler will complain that the method myfunc in the base class has no body, nobody at all to print it.
5  class A, B and C are in multilevel inheritance hierarchy repectively . In the main method of some other class if class C object is created, in what sequence the three constructors execute?	Constructor of A executes first, followed by the constructor of B and C	Constructor of C executes first followed by the constructor of B and A	Constructor of C executes first followed by the constructor of A and B	Constructor of A executes first followed by the constructor of C and B
6 Consider the following code and choose the correct option: <pre>package aj; private class S{ int roll;     S(){roll=1;} } package aj; class T { public static void main(String ar[]){     System.out.print(new S().roll);}} }</pre>	Compilation error	Compiles and display 1	Compiles but no output	Compiles and display 0

7 Here is the general syntax for method definition:

```
accessModifier returnType methodName(  
parameterList )  
{  
    Java statements  
  
    return returnValue;  
}
```

What is true for the returnType and the returnValue?

8 A) A call to instance method can not be made from static context.

B) A call to static method can be made from non static context.

9 Consider the following code and choose the correct option:

```
class A{ A(){System.out.print("From A");}}  
class B extends A{ B(int z){z=2;}  
public static void main(String args[]){  
    new B(3);}}
```

10 class Sample

```
int a,b;  
Sample()  
{ a=1; b=2;  
System.out.println(a+"\t"+b);  
}  
Sample(int x)  
{ this(10,20);  
a=b=x;  
System.out.println(a+"\t"+b);  
}  
Sample(int a,int b)  
{ this();  
this.a=a;  
this.b=b;  
System.out.println(a+"\t"+b);  
}
```

```
}
```

class This2  
{ public static void main(String args[])  
{  
 Sample s1=new Sample (100);  
}}

What is the Output of the Program?

The returnValue can be any type, but will be automatically converted to returnType when the method returns to the caller

If the returnType is void then the returnValue can be any type

The returnValue must be the same type as the returnType, or be of a type that can be converted to returnType without loss of information

The returnValue must be exactly the same type as the returnType.

Both are FALSE

Both are TRUE

Only A is TRUE

Only B is TRUE

Compilation error

Compiles but throws exception

Compiles and runtime prints From A exception

Compiles and display 3

100	100	1	2	1	2	100	10	20	1	2	100	1	2	10	20	100
-----	-----	---	---	---	---	-----	----	----	---	---	-----	---	---	----	----	-----

10 20

10 20

100

100

11 Consider the following code and choose the correct option: <pre>class A{ private static void display(){     System.out.print("Hi");}     public static void main(String ar[]){         display();}}</pre>	Compiles and displays Hi	Compiles and throw run time exception	Compiles but doesn't display anything	Compilation fails
12 Consider the following code and choose the correct option: <pre>package aj; class A{ protected int j; } package bj; class B extends A { public static void main(String ar[]){     System.out.print(new A().j=23);}}</pre>	code compiles fine and will display 23	code compiles but will not display output	compilation error	j can not be initialized
13 Consider the following code and choose the correct option: <pre>class A{ int z; A(int x){z=x;} } class B extends A{     public static void main(String arg){         new B();}}</pre>	Compilation error	Compiles but throws run time exception	Compiles and displays nothing	None of the listed options
14 class Test{ static void method(){ this.display(); } static display(){ System.out.println("hello"); } public static void main(String[] args){ new Test().method(); } }	consider the code above & select the proper output from the options.	hello	Runtime Error	compiles but no output does not compile
15 What will be the result when you try to compile and run the following code? <pre>private class Base{     Base(){         int i = 100;         System.out.println(i);     } }  public class Pri extends Base{     static int i = 200;     public static void main(String argv[]){         Pri p = new Pri();         System.out.println(i);     } }</pre>		100 followed by 200	Compile time error	100

```

16 public class MyClass {
    static void print(String s, int i) {
        System.out.println("String: " + s + ", int: " +
i);
    }

    static void print(int i, String s) {
        System.out.println("int: " + i + ", String: " +
s);
    }

    public static void main(String[] args) {
        print("String first", 11);
        print(99, "Int first");
    }
}What would be the output?

```

String: String  
 first, int: 11  
 int: 99,  
 String: Int  
 first

int: 27,  
 String: Int  
 first String:  
 String first,  
 int: 27

Compilation Error      Runtime Exception

- 17
- A) No argument constructor is provided to all Java classes by default
  - B) No argument constructor is provided to the class only when no constructor is defined.
  - C) Constructor can have another class object as an argument
  - D) Access specifiers are not applicable to Constructor

Only A is TRUE

All are TRUE      B and C is TRUE

All are FALSE

- 18 Consider the following code and choose the correct option:

```

class Test{ private static void display(){
System.out.println("Display()");
private static void show() { display();
System.out.println("show()");
public static void main(String arg[]){
show();}}

```

Compiles and prints show()

Compiles and prints Display() shows runtime exception

Compilation error

- 19 Which of the following sentences is true?

- A) Access to data member depends on the scope of the class and the scope of data members
- B) Access to data member depends only on the scope of the data members
- C) Access to data member depends on the scope of the method from where it is accessed

Only A and C is TRUE

All are TRUE      FALSE

Only A is TRUE

20 Given:

```
public class Yikes {  
  
    public static void go(Long n)  
    {System.out.print("Long ");}  
    public static void go(Short n)  
    {System.out.print("Short ");}  
    public static void go(int n)  
    {System.out.print("int ");}  
    public static void main(String [] args) {  
        short y = 6;  
        long z = 7;  
        go(y);  
        go(z);  
    }  
}
```

What is the result?

21

Which of the following will print -4.0

22 Suppose class B is sub class of class A:

- A) If class A doesn't have any constructor, then class B also must not have any constructor
- B) If class A has parameterized constructor, then class B can have default as well as parameterized constructor
- C) If class A has parameterized constructor then call to class A constructor should be made explicitly by constructor of class B

23 class Order{

```
Order(){  
    System.out.println("Cat");  
}  
public static void main(String... Args){  
    System.out.println("Ant");  
}  
static{  
    System.out.println("Dog");  
}  
{  
    System.out.println("Man");  
}
```

consider the code above & select the proper output from the options.

24 Consider the following code and choose the correct option:

```
class A{ private void display(){  
    System.out.print("Hi");}  
    public static void main(String ar[]){  
        display();}}
```

int Long	Short Long	Compilation fails.	An exception is thrown at runtime.
System.out.println(Math.ceil(-4.7));	System.out.println(Math.floor(-4.7));	System.out.println(Math.round(-4.7));	System.out.println(Math.min(-4.7));
Only B and C is TRUE	Only A is TRUE	All are FALSE	Only A and C is TRUE
Dog Ant	Dog Man Cat Ant	Man Dog Ant	Dog Man Ant
Compiles but doesn't display anything	Compiles and throws run time exception	Compilation fails	Compiles and displays Hi

25

Consider the following code and choose the correct option:

```
public class MyClass {  
    public static void main(String arguments[]) {  
        amethod(arguments);  
    }  
    public void amethod(String[] arguments) {  
        System.out.println(arguments[0]);  
        System.out.println(arguments[1]);  
    }  
}
```

Command Line arguments - Hi, Hello

prints  
Hi  
Hello

Compiler  
Error

Runs but no  
output

Runtime  
Error

26 package QB;

```
class Sphere {  
    protected int methodRadius(int r) {  
        System.out.println("Radious is: "+r);  
        return 0;  
    }  
}
```

```
package QB;  
public class MyClass {  
    public static void main(String[] args) {  
        double x = 0.89;  
        Sphere sp = new Sphere();  
        // Some code missing  
    }  
}
```

to get the radius value what is the code of line to be added ?

methodRadius  
s(x);

sp.methodRa  
dius(x);

Nothing to  
add

Sphere.meth  
odRadius();

27 class One{

```
int var1;  
One (int x){  
    var1 = x;  
}
```

```
class Derived extends One{  
int var2;  
void display(){  
    System.out.println("var  
1="+var1+"var2="+var2);  
}
```

```
class Main{  
    public static void main(String[] args){  
        Derived obj = new Derived();  
        obj.display();  
    }  
}
```

consider the code above & select the proper output from the options.

0 , 0

compiles  
successfully  
but runtime  
error

compile error none of these

28 Consider the following code and choose the correct option:	Compiles and prints show()	Compiles and prints display()	Compiles but throws runtime exception	Compilation error
<pre>class Test{ private void display(){} System.out.println("Display()"); private static void show() { display(); System.out.println("show()"); public static void main(String arg[]){ show();}}</pre>				
29 Consider the following code and choose the best option:	Compilation error	Compiles and runs without any output	Compiles and display 2	Compiles and display 0
<pre>class Super{ int x; Super(){x=2;} class Sub extends Super { void displayX(){ System.out.print(x);} public static void main(String args[]){ new Sub().displayX();}}</pre>				
30 class One{ int var1; One (int x){ var1 = x; } class Derived extends One{ int var2; Derived(){ super(10); var2=10; } void display(){ System.out.println("var1="+var1+", var2="+var2); } class Main{ public static void main(String[] args){ Derived obj = new Derived(); obj.display(); } consider the code above & select the proper output from the options.	var1=10 , var2=10	0,0	compile error	runtime error
31 public class MyAr { static int i1; public static void main(String argv[]) { MyAr m = new MyAr(); m.amethod(); } public void amethod() { System.out.println(i1); } }	Compilation Error	Garbage Value	It is not possible to access a static variable in side of non static method	
What is the output of the program?				

32 What will be printed out if you attempt to compile and run the following code ?

```
public class AA {  
    public static void main(String[] args) {  
        int i = 9;  
        switch (i) {  
            default:  
                System.out.println("default");  
            case 0:  
                System.out.println("zero");  
                break;  
            case 1:  
                System.out.println("one");  
            case 2:  
                System.out.println("two");  
        }  
    }  
}
```

Compilation Error	default	default zero	default zero one two
-------------------	---------	--------------	-------------------------

33 Which statements, when inserted at (1), will not result in compile-time errors?

```
public class ThisUsage {  
    int planets;  
    static int suns;  
    public void gaze() {  
        int i;  
        // (1) INSERT STATEMENT HERE  
    }  
}
```

i = this.planets;	this = new i = this.suns;	ThisUsage();	this.suns = planets;
----------------------	------------------------------	--------------	-------------------------

34 Which modifier is used to control access to critical code in multi-threaded programs?

default	public	transient	synchronized
---------	--------	-----------	--------------

35 package QB;

```
class Meal {  
    Meal() {  
        System.out.println("Meal()");  
    }  
}  
class Cheese {  
    Cheese() {  
        System.out.println("Cheese()");  
    }  
}  
class Lunch extends Meal {  
    Lunch() {  
        System.out.println("Lunch()");  
    }  
}  
class PortableLunch extends Lunch {  
    PortableLunch() {  
        System.out.println("PortableLunch()");  
    }  
}  
class Sandwich extends PortableLunch {  
    private Cheese c = new Cheese();  
  
    public Sandwich() {  
        System.out.println("Sandwich()");  
    }  
}  
public class MyClass7 {  
    public static void main(String[] args) {  
        new Sandwich();  
    }  
}
```

36 Consider the following code and choose the correct option:

```
class A{ int a; A(int a){a=4;}}  
class B extends A{ B(){super(3);} void  
displayA(){  
System.out.print(a);}  
public static void main(String args[]){  
new B().displayA();}}
```

Meal()  
Lunch()  
PortableLunc  
h()  
Cheese()  
Sandwich()

compiles and  
display 0

Meal()  
Cheese()  
Lunch()  
PortableLunc  
h()  
Sandwich()  
Sandwich()

compilation  
error

Meal()  
Lunch()  
PortableLunc  
h()  
Sandwich()  
Cheese()

Compiles and  
display 4

Cheese()  
Sandwich()  
Meal()  
Lunch()  
PortableLunc  
h()

Compiles and  
display 3

37

Given the following code what will be output?

```
public class Pass{  
    static int j=20;  
    public static void main(String argv[]){  
        int i=10;  
        Pass p = new Pass();  
        p.amethod(i);  
        System.out.println(i);  
        System.out.println(j);  
    }  
  
    public void amethod(int x){  
        x=x*2;  
        j=j*2;  
    }  
}
```

Error:  
amethod  
parameter  
does not  
match  
variable  
The compiler  
will  
automatically  
change the  
private  
variable to a  
public  
variable

20 and 40      10 and 40      10, and 20

The compiler  
will find the  
error and will  
not make a  
.class file

The program  
will compile  
and run  
successfully

The program  
will compile  
successfully,  
but the .class  
file will not  
run correctly

38

What will happen if a main() method of a "testing" class tries to access a private instance variable of an object using dot notation?

```
39 11. class Mud {  
12. // insert code here  
13. System.out.println("hi");  
14. }  
15. }
```

And the following five fragments:

```
public static void main(String...a) {  
public static void main(String.* a) {  
public static void main(String... a) {  
public static void main(String[]... a) {  
public static void main(String...[] a) {
```

How many of the code fragments, inserted independently at line 12, compile?

0      1      2      3

<pre>40 class Order{     Order(){}     System.out.println("Cat"); } public static void main(String... Args){     Order obj = new Order();     System.out.println("Ant"); } static{     System.out.println("Dog"); } {     System.out.println("Man"); }</pre>	<p>consider the code above &amp; select the proper output from the options.</p>	<table border="1"> <tr> <td>Man</td><td>Dog</td><td>Cat</td><td>Cat</td><td>Ant</td><td>Dog</td><td>Dog</td><td>Man</td><td>Cat</td><td>Ant</td><td>compile error</td></tr> <tr> <td></td><td></td><td>Ant</td><td></td><td>Man</td><td></td><td></td><td></td><td></td><td></td><td></td></tr> </table>	Man	Dog	Cat	Cat	Ant	Dog	Dog	Man	Cat	Ant	compile error			Ant		Man						
Man	Dog	Cat	Cat	Ant	Dog	Dog	Man	Cat	Ant	compile error														
		Ant		Man																				
<pre>41 abstract class MineBase {     abstract void amethod();     static int i; }  public class Mine extends MineBase {     public static void main(String argv[]){         int[] ar=new int[5];         for(i=0;i &lt; ar.length;i++)             System.out.println(ar[i]);     } }</pre>	<p>A Sequence of 5 zero's will be printed like 0 0 0 0</p>	<p>A Sequence of 5 one's will be printed like 1 1 1 1</p>	<p>IndexOutOfBoundsException</p>	<p>Compiler Error occurs and to avoid them we need to declare Mine class as abstract</p>																				
<pre>42 public class Q {     public static void main(String argv[]) {         int anar[] = new int[] { 1, 2, 3 };         System.out.println(anar[1]);     } }</pre>	<p>Compiler Error: anar is referenced before it is initialized</p>	<p>Compiler Error: size of array must be 1 defined</p>																						
<p>43 A constructor may return value including class type</p>	<p>true</p>	<p>false</p>																						
<p>44 Consider the following code and choose the correct option:</p> <pre>package aj; class S{ int roll =23; private S(){} } package aj; class T { public static void main(String ar[]){     System.out.print(new S().roll);} }</pre>	<p>Compilation error</p>	<p>Compiles and display 0</p>	<p>Compiles and display 23</p>	<p>Compiles but no output</p>																				

45	public class c123 { private c123() { System.out.println("Hellow"); } public static void main(String args[]) { c123 o1 = new c123(); c213 o2 = new c213(); } }  class c213 { private c213() { System.out.println("Hello123"); } }		It is not possible to declare a constructor as private	Compilation Error	Runs without any output
	What is the output?	Hellow			
46	class MyClass1 { private int area(int side) { return(side * side); } public static void main(String args[ ]) { MyClass1 MC = new MyClass1( ); int area = MC.area(50); System.out.println(area); } }		Compilation error	Runtime Exception	2500      50
47	What would be the output?				
	public class MyAr { public static void main(String argv[]) { MyAr m = new MyAr(); m.amethod(); } public void amethod() { static int i1; System.out.println(i1); } }		Compile time error because i has not been initialized	Compilation and output of null	It is not possible to declare a static variable in side of non static method or instance method. Because Static variables are class level dependencie s.
	What is the Output of the Program?				

<pre>48 public class MyAr {     public static void main(String argv[]) {         MyAr m = new MyAr();         m.amethod();     }     public void amethod() {         final int i1;         System.out.println(i1);     } }</pre> <p>What is the Output of the Program?</p>	<p>Unresolved compilation problem: The local variable i1 may not have been initialized</p>	<p>Compilation and output of null</p>	<p>None of the given options</p>
<pre>49 public class c1 {     private c1()     {         System.out.println("Hello");     }     public static void main(String args[])     {         c1 o1=new c1();     } }</pre> <p>What is the output?</p>	<p>Hello</p>	<p>It is not possible to declare a constructor private</p>	<p>Can't create object because constructor is private</p>
<p>50 Which modifier indicates that the variable might be modified asynchronously, so that all threads will get the correct value of the variable.</p>	<p>synchronized</p>	<p>volatile</p>	<p>transient default</p>

51

```
class A {  
    int i, j;  
  
    A(int a, int b) {  
        i = a;  
        j = b;  
    }  
    void show() {  
        System.out.println("i and j: " + i + " " + j);  
    }  
}  
class B extends A {  
    int k;  
  
    B(int a, int b, int c) {  
        super(a, b);  
        k = c;  
    }  
    void show(String msg) {  
        System.out.println(msg + k);  
    }  
}  
class Override {  
    public static void main(String args[]) {  
        B subOb = new B(3, 5, 7);  
        subOb.show("This is k: "); // this calls  
        show() in B  
        subOb.show(); // this calls show() in A  
    }  
}
```

What would be the output?

This is j: 5 i  
and k: 3 7      This is i: 3 j  
and k: 5 7      This is i: 7 j  
and k: 3 5      This is k: 7 i  
and j: 3 7

52 Consider the following code and choose the correct option:

```
class X { int x; X(int x){x=2;}}  
class Y extends X{ Y(){}} void displayX()  
System.out.print(x);  
public static void main(String args[]){  
    new Y().displayX();}}
```

Compiles and  
display 2      Compiles and  
runs without  
any output      Compiles and  
display 0      Compilation  
error

53 class Order{  
 Order(){  
 System.out.println("Cat");  
 }  
 public static void main(String... Args){  
 Order obj = new Order();  
 System.out.println("Ant");  
 }  
 static{  
 System.out.println("Dog");  
 }  
}

consider the code above & select the proper output from the options.

Cat Ant Dog      Dog Cat Ant      Ant Cat Dog      none

54 What will be the result when you attempt to compile this program? <pre>public class Rand{     public static void main(String argv[]){         int iRand;         iRand = Math.random();         System.out.println(iRand);     } }</pre>	Compile time error referring to a cast problem	A random number between 1 and 10	A random number between 0 and 1	A compile time error as random being an undefined method
55 Choose the meta annotations. (Choose THREE)	Target	Retention	Deprecated	Documented
56 If no retention policy is specified for an annotation, then the default policy of _____ is used.	method	class	source	runtime
57 Select the variable which are in java.lang.annotation.RetentionPolicy class. (Choose THREE)	SOURCE	CLASS	RUNTIME	CONSTRUCTOR
58 Select the Uses of annotations. (Choose THREE)	Information For the Compiler	Information for the JVM	Compile time and deploytime processing	Runtime processing
59 All annotation types should manually extend the Annotation interface. State TRUE/FALSE	true	false		
60 Custom annotations can be created using	@interface	@inherit	@include	all the listed options
61 Given: 10. interface A { void x(); } 11. class B implements A { public void x() {} public void y() {} } 12. class C extends B { public void x() {} } And: 20. java.util.List<a> list = new java.util.ArrayList</a>(); 21. list.add(new B()); 22. list.add(new C()); 23. for (A a:list) { 24.     a.x(); 25.     a.y(); 26. }	Compilation fails because of an error in line 25	The code runs with no output.	An exception is thrown at runtime	Compilation fails because of an error in line 21
What is the result?				

62 Given:

```
public static Collection get() {  
    Collection sorted = new LinkedList();  
    sorted.add("B"); sorted.add("C");  
    sorted.add("A");  
    return sorted;  
}  
public static void main(String[] args) {  
    for (Object obj: get()) {  
        System.out.print(obj + ", ");  
    }  
}
```

What is the result?

A, B, C,

B, C, A,

Compilation fails.

An exception is thrown at runtime.

63 Which statement is true about the following program?

```
import java.util.ArrayList;  
import java.util.Collections;  
import java.util.List;  
public class WhatISThis {  
    public static void main(String[] na){  
        List<StringBuilder> list=new  
        ArrayList<StringBuilder>();  
        list.add(new StringBuilder("B"));  
        list.add(new StringBuilder("A"));  
        list.add(new StringBuilder("C"));  
        Collections.sort(list,Collections.reverseOrder());  
        System.out.println(list.subList(1,2));  
    }  
}
```

The program will compile and print the following output: [B]

The program will compile and print the following output: [B,A]

The program will compile and throw a runtime exception

The program will not compile

64 Consider the following code and choose the correct option:

```
public static void before() {  
    Set set = new TreeSet();  
    set.add("2");  
    set.add(3);  
    set.add("1");  
    Iterator it = set.iterator();  
    while (it.hasNext())  
        System.out.print(it.next() + " ");  
}
```

The before() method will print 1 2

The before() method will print 1 2 3

The before() method will throw an exception at runtime

The before() method will not compile

65 import java.util.StringTokenizer;  
class ST{  
 public static void main(String[] args){  
 String input = "Today is\$Holiday";  
 StringTokenizer st = new StringTokenizer(input,"\$");  
 while(st.hasMoreTokens()){  
 System.out.println(st.nextElement());  
 }  
 }  
}

Today is Holiday

Today is Holiday

Both

none of the listed options

	66 Given: <pre>public static Iterator reverse(List list) {     Collections.reverse(list);     return list.iterator(); } public static void main(String[] args) {     List list = new ArrayList();     list.add("1"); list.add("2"); list.add("3");     for (Object obj: reverse(list))         System.out.print(obj + ", "); }</pre> <p>What is the result?</p>	3, 2, 1,	1, 2, 3,	Compilation fails.	The code runs with no output.
67	Which collection class allows you to grow or shrink its size and provides indexed access to its elements, but its methods are not synchronized?	java.util.HashSet	java.util.LinkedHashSet	java.util.List	java.util.ArrayList
68	int indexOf(Object o) - What does this method return if the element is not found in the List?	null	-1		none of the 0 listed options
69	What is the result of attempting to compile and run the following code?  <pre>import java.util.Vector; import java.util.LinkedList; public class Test1{ public static void main(String[] args) { Integer int1 = new Integer(10); Vector vec1 = new Vector(); LinkedList list = new LinkedList(); vec1.add(int1); list.add(int1); if(vec1.equals(list))     System.out.println("equal"); else     System.out.println("not equal"); } } </pre> 1. The code will fail to compile. 2. Runtime error due to incompatible object comparison 3. Will run and print "equal". 4. Will run and print "not equal".		1	2	3
70	Consider the following code and choose the correct option:  <pre>class Test{     public static void main(String args[]){         Integer arr[]={3,4,3,2};         Set&lt;Integer&gt; s=new         TreeSet&lt;Integer&gt;(Arrays.asList(arr));         s.add(1);         for(Integer ele :s){             System.out.println(ele); } }}</pre>	Compilation error		prints 3,4,2,1, prints 1,2,3,4	Compiles but exception at runtime

71

Inorder to remove one element from the given Treeset, place the appropriate line of code

```
public class Main {  
    public static void main(String[] args) {  
        TreeSet<Integer> tSet = new  
        TreeSet<Integer>();  
        System.out.println("Size of TreeSet : " +  
        tSet.size());  
        tSet.add(new Integer("1"));  
        tSet.add(new Integer("2"));  
        tSet.add(new Integer("3"));  
        System.out.println(tSet.size());  
        // remove the one element from the Treeset  
        System.out.println("Size of TreeSet after  
removal : " + tSet.size());  
    }  
}
```

```
tSet.clear(ne w  
tSet.delete(ne w  
tSet.remove( new  
Integer("1")); Integer("1")); Integer("1"));  
tSet.drop(ne w  
Integer("1"));
```

72

Consider the code below & select the correct output from the options:

```
public class Test{  
    public static void main(String[] args) {  
        String  
        [colors={"orange","blue","red","green","ivory"};  
        Arrays.sort(colors);  
        int s1=Arrays.binarySearch(colors, "ivory");  
        int s2=Arrays.binarySearch(colors, "silver");  
        System.out.println(s1+" "+s2); }}}
```

2 -4                  3 -5                  2 -6                  3 -4

73 Consider the following code and choose the correct output:

```
class Test{  
    public static void main(String args[]){  
        TreeMap<Integer, String> hm=new  
        TreeMap<Integer, String>();  
        hm.put(2,"Two");  
        hm.put(4,"Four");  
        hm.put(1,"One");  
        hm.put(6,"Six");  
        hm.put(7,"Seven");  
        SortedMap<Integer, String>  
        sm=hm.subMap(2,7);  
        SortedMap<Integer, String>  
        sm2=sm.tailMap(4);  
        System.out.print(sm2);  
    }}}
```

{2=Two,  
4=Four,  
6=Six,  
7=Seven}  
{4=Four,  
6=Six,  
7=Seven}  
{4=Four,  
6=Six}  
{2=Two,  
4=Four,  
6=Six}

74 next() method of Scanner class will return

\_\_\_\_\_                  Integer                  Long                  int                  String

75

Given:

```
import java.util.Arrays;
import java.util.HashSet;
import java.util.Set;

public class MainClass {

    public static void main(String[] a) {
        String elements[] = { "A", "B", "C", "D", "E" };
        Set set = new
HashSet(Arrays.asList(elements));

        elements = new String[] { "A", "B", "C", "D"
    };
        Set set2 = new
HashSet(Arrays.asList(elements));

        System.out.println(set.equals(set2));
    }
}
```

What is the result of given code?

true

false

Compile time error

Runtime Exception

76 A)Property files help to decrease coupling  
B) DateFormat class allows you to format dates and times with customized styles.  
C) Calendar class allows to perform date calculation and conversion of dates and times between timezones.  
D) Vector class is not synchronized

A and B is TRUE

A and D is TRUE

A and C is TRUE

B and D is TRUE

77 Which interface does java.util.Hashtable implement?

Java.util.Map

Java.util.List

Java.util.Table

Java.util.Collection

78 Object get(Object key) - What does this method return if the key is not found in the Map?

0

-1 null

none of the listed options

79 Consider the following code and choose the correct option:

```
class Test{
    public static void main(String args[]){
        TreeSet<Integer> ts=new
TreeSet<Integer>();
        ts.add(1);
        ts.add(8);
        ts.add(6);
        ts.add(4);
        SortedSet<Integer> ss=ts.subSet(2, 10);
        ss.add(9);
        System.out.println(ts);
        System.out.println(ss);
    }
}
```

[1,4,6,8]  
[4,6,8,9]

[1,8,6,4]  
[8,6,4,9]

[1,4,6,8,9]  
[4,6,8,9]

[1,4,6,8,9]  
[4,6,8]

80 A) Iterator does not allow to insert elements during traversal

B) Iterator allows bidirectional navigation.

C) ListIterator allows insertion of elements during traversal

D) ListIterator does not support bidirectional navigation

81 static void sort(List list) method is part of \_\_\_\_\_

82 static int binarySearch(List list, Object key) is a method of \_\_\_\_\_

83 Which collection class allows you to access its elements by associating a key with an element's value, and provides synchronization?

84

Consider the following code and select the correct output:

```
import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;
public class Lists {
    public static void main(String[] args) {
        List<String> list=new ArrayList<String>();
        list.add("1");
        list.add("2");
        list.add(1, "3");
        List<String> list2=new LinkedList<String>(list);
        list.addAll(list2);
        list2=list.subList(2,5);
        list2.clear();
        System.out.println(list);
    }
}
```

A and B is TRUE

Collection interface

Vector class

A and D is TRUE

Collections class

ArrayList class

A and C is TRUE

Vector class

Collection interface

B and D is TRUE

ArrayList class

Collections class

java.util.SortedMap

java.util.TreeMap

java.util.TreeSet

java.util.HashTable

85 Given:

```
import java.util.*;
```

```
public class LetterASort{
    public static void main(String[] args) {
        ArrayList<String> strings = new
        ArrayList<String>();
        strings.add("aAaA");
        strings.add("AaA");
        strings.add("aAa");
        strings.add("AAaa");
        Collections.sort(strings);
        for (String s : strings) { System.out.print(s + "
"); }
    }
}
```

What is the result?

Compilation fails.

aAaA aAa  
AAaa AaA

AAaa AaA  
aAa aAAA

AaA AAaa  
aAaA aAa

86	A) It is a good practice to store heterogenous data in a TreeSet. B) HashSet has default initial capacity (16) and loadfactor(0.75) C)HashSet does not maintain order of Insertion D)TreeSet maintains order of Inserstion	A and B is TRUE	A and D is TRUE	A and C is TRUE
87	<pre>TreeSet&lt;String&gt; s = new TreeSet&lt;String&gt;(); TreeSet&lt;String&gt; subs = new TreeSet&lt;String&gt;(); s.add("a"); s.add("b"); s.add("c"); s.add("d"); s.add("e");  subs = (TreeSet)s.subSet("b", true, "d", true); s.add("g"); s.pollFirst(); s.pollFirst(); s.add("c2"); System.out.println(s.size() +" "+ subs.size());</pre>	The size of s is 4	The size of s is 5	The size of subs is 3
88	Consider the following code was executed on June 01, 1983. What will be the output? <pre>class Test{ public static void main(String args[]){     Date date=new Date();     SimpleDateFormat sd;     sd=new SimplpeDateFormat("E MMM dd yyyy");     System.out.print(sd.format(date));}}</pre>	Wed Jun 01 1983	244 JUN 01 1983	PST JUN 01 1983
89	Given: <pre>public class Venus { public static void main(String[] args) { int [] x = {1,2,3}; int y[] = {4,5,6}; new Venus().go(x,y); } void go(int[]... z) { for(int[] a : z) System.out.print(a[0]); } }</pre> <p>} What is the result?</p>	123	12	14

90

You wish to store a small amount of data and make it available for rapid access. You do not have a need for the data to be sorted, uniqueness is not an issue and the data will remain fairly static. Which data structure might be most suitable for this requirement?

- 1) TreeSet
- 2) HashMap
- 3) LinkedList
- 4) an array

1

2

3

4

91 What will be the output of following code?

```
class Test{  
    public static void main(String args[]){  
        TreeSet<Integer> ts=new  
        TreeSet<Integer>();  
        ts.add(2);  
        ts.add(3);  
        ts.add(7);  
        ts.add(5);  
        SortedSet<Integer> ss=ts.subSet(1,7);  
        ss.add(4);  
        ss.add(6);  
        System.out.print(ss);}}}
```

[2,3,7,5]

[2,3,7,5,4,6]

[2,3,4,5,6,7]

[2,3,4,5,6]

92

Consider the following code and choose the correct option:

```
class Data{ Integer data; Data(Integer  
d){data=d;}  
public boolean equals(Object o){return true;}  
public int hashCode(){return 1;}  
class Test{  
    public static void main(String ar[]){  
        Set<Data> s=new HashSet<Data>();  
        s.add(new Data(4));  
        s.add(new Data(2));  
        s.add(new Data(4));  
        s.add(new Data(1));  
        s.add(new Data(2));  
        System.out.print(s.size());}}
```

3

compilation  
5 error

Compiles but  
error at run  
time

93 Consider the code below & select the correct output from the options:

```
public class Test{  
public static void main(String[] args) {  
String num="";  
z: for(int x=0;x<3;x++)  
    for(int y=0;y<2;y++){  
        if(x==1) break;  
        if(x==2 && y==1) break z;  
        num=num+x+y;  
    }System.out.println(num);}}
```

0 0 0 1

0 0 0 1 2 0

0 0 0 1 2 0 2

1

Compilation  
error

94 Given:				
<pre>public class Test {     public enum Dogs {collie, harrier};     public static void main(String [] args) {         Dogs myDog = Dogs.collie;         switch (myDog) {             case collie:                 System.out.print("collie ");             case harrier:                 System.out.print("harrier ");         }     } }</pre> <p>What is the result?</p>	collie	harrier	Compilation fails.	collie harrier
95 Consider the following code and choose the correct output:				
<pre>class Test{     public static void main(String args[]){         boolean flag=true;         if(flag=false){             System.out.print("TRUE");         }else{             System.out.print("FALSE");         }     } }</pre>	true	false	compilation error	Compiles
96 Cosider the following code and choose the correct option:				
<pre>class Test{     public static void main(String args[]){         System.out.println(Integer.parseInt("21474836 48", 10));     } }</pre>	Compilation error	2.15E+09	NumberFormat atException	Compiles but no output
97				
<p>Given:</p> <pre>public class Test {     public enum Dogs {collie, harrier, shepherd};     public static void main(String [] args) {         Dogs myDog = Dogs.shepherd;         switch (myDog) {             case collie:                 System.out.print("collie ");             case default:                 System.out.print("retriever ");             case harrier:                 System.out.print("harrier ");         }     } }</pre> <p>What is the result?</p>	harrier	shepherd	retriever	Compilation fails.

98 Given:

```
static void myFunc()
{
    int i, s = 0;
    for (int j = 0; j < 7; j++) {
        i = 0;
        do {
            i++;
            s++;
        } while (i < j);
    }
    System.out.println(s);
}
```

} What would be the result

20

21

22

23

99

What is the range of the random number r generated by the code below?

int r = (int)(Math.floor(Math.random() \* 8)) + 2; 2 <= r <= 9 3 <= r <= 10 2<= r <= 10 3 <= r <= 9

100 class Test{

```
public static void main(String[] args) {
    int x=-1,y=-1;
    if(++x==++y)
        System.out.println("R.T. Ponting");
    else
        System.out.println("C.H. Gayle");
}
```

consider the code above & select the proper output from the options.

R.T.Ponting

C.H.Gayle

Compile error

none of the listed options

101 Given:

```
public class Breaker2 {
    static String o = "";
    public static void main(String[] args) {
        z:
        for(int x = 2; x < 7; x++) {
            if(x==3) continue;
            if(x==5) break z;
            o = o + x;
        }
        System.out.println(o);
    }
}
```

What is the result?

2

24

234

246

102 Consider the following code and choose the correct output:

```
class Test{
    public static void main(String args[]){
        int a=5;
        if(a=3){
            System.out.print("Three");}else{
            System.out.print("Five");}}}
```

Compilation error

Three

Five

Compiles but no output

103 Given:

```
public class Batman {  
    int squares = 81;  
    public static void main(String[] args) {  
        new Batman().go();  
    }  
    void go() {  
        incr(++squares);  
        System.out.println(squares);  
    }  
    void incr(int squares) { squares += 10; }  
}
```

What is the result?

104 public void foo( boolean a, boolean b)

```
{  
    if( a )  
    {  
        System.out.println("A"); /* Line 5 */  
    }  
    else if(a && b) /* Line 7 */  
    {  
        System.out.println( "A && B");  
    }  
    else /* Line 11 */  
    {  
        if ( !b )  
        {  
            System.out.println( "notB");  
        }  
        else  
        {  
            System.out.println( "ELSE" );  
        }  
    }  
}
```

What would be the result?

105 What is the value of 'n' after executing the following code?

```
int n = 10;  
int p = n + 5;  
int q = p - 10;  
int r = 2 * (p - q);  
switch(n)  
{  
    case p: n = n + 1;  
    case q: n = n + 2;  
    case r: n = n + 3;  
    default: n = n + 4;  
}
```

81

82

91

92

If a is true  
and b is false  
then the  
output is  
"notB"

If a is true  
and b is true  
then the  
output is "A  
&& B"

If a is false  
and b is false  
then the  
output is  
"ELSE"

If a is false  
and b is true  
then the  
output is  
"ELSE"

14

Compilation  
28 Error

10

106	<pre>public class While {     public void loop()     {         int x= 0;         while ( 1 ) /* Line 6 */         {             System.out.print("x plus one is " + (x + 1)); /* Line 8 */         }     } }</pre> <p>Which statement is true?</p>				
107	<p>Which of the following loop bodies DOES compute the product from 1 to 10 like (1 * 2 * 3 * 4 * 5 * 6 * 7 * 8 * 9 * 10)?</p> <pre>int s = 1; for (int i = 1; i &lt;= 10; i++) {     &lt;What to put here?&gt; }</pre>	There is a syntax error on line 1	There are syntax errors on lines 1 and 6	There are syntax errors on lines 1, 6, and 8	There is a syntax error on line 6
108	<p>Which of the following statements are true regarding wrapper classes? (Choose TWO)</p>	s += i * i;	s++;	s = s + s * i;	s *= i;
109	<p>Given:</p> <pre>class Atom {     Atom() { System.out.print("atom "); } } class Rock extends Atom {     Rock(String type) { System.out.print(type); } } public class Mountain extends Rock {     Mountain() {         super("granite ");         new Rock("granite ");     }     public static void main(String[] a) { new         Mountain(); } }</pre> <p>What is the result?</p>	String is a wrapper class	Double has a compareTo() method	Character has a intValue() method	Byte extends Number
		Compilation fails.	granite granite	atom granite granite	atom granite atom granite

110 What are the thing to be placed to complete the code?

```
class Wrap {  
    public static void main(String args[]) {  
  
        _____ iOb = _____  
        Integer(100);  
  
        int i = iOb.intValue();
```

```
        System.out.println(i + " " + iOb); // displays  
100 100
```

```
    }  
}  
}
```

111 public class SwitchTest

```
{  
    public static void main(String[] args)  
    {  
        System.out.println("value =" +  
switchIt(4));  
    }  
    public static int switchIt(int x)  
    {  
        int j = 1;  
        switch (x)  
        {  
            case 1: j++;  
            case 2: j++;  
            case 3: j++;  
            case 4: j++;  
            case 5: j++;  
            default: j++;  
        }  
        return j + x;  
    }  
}
```

What will be the output of the program?

int, int

Integer, new Integer, int

int, Integer

112 Given:

```
public class Barn {  
    public static void main(String[] args) {  
        new Barn().go("hi", 1);  
        new Barn().go("hi", "world", 2);  
    }  
    public void go(String... y, int x) {  
        System.out.print(y[y.length - 1] + " ");  
    }  
}
```

What is the result?

value = 8

value = 2

value = 4

value = 6

hi hi

hi world

world world

Compilation fails.

113 Consider the following code and choose the correct option:				
<pre>class Test{     public static void main(String args[]){         int x=034;         int y=12;         int ans=x+y;         System.out.println(ans);     } }</pre>	40	compilation error	Compiles but error at run time	
114				
<pre>11. double input = 314159.26; 12. NumberFormat nf = NumberFormat.getInstance(Locale.ITALIAN); 13. String b; 14. //Insert code here</pre> <p>Which code, inserted at line 14, sets the value of b to 314.159,26?</p>		b = b = nf.format( b = nf.equals( nf.parseObj input ); input ); ct( input );		
115 Consider the following code and choose the correct option:				
<pre>class Test{ public static void main(String ar[]){     TreeMap&lt;Integer,String&gt; tree = new TreeMap&lt;Integer,String&gt;();     tree.put(1, "one");     tree.put(2, "two");     tree.put(3, "three");     tree.put(4,"Four");     System.out.println(tree.higherKey(2));     System.out.println(tree.ceilingKey(2));     System.out.println(tree.floorKey(1));     System.out.println(tree.lowerKey(1)); }}</pre>	3 2 1 null	3 2 1 1	2 2 1 1	4 2 1 1
116 Consider the following code and choose the correct option:				
<pre>class Test{     public static void main(String args[]){         Long data=23;         System.out.println(data); }}</pre>	23	Compilation error	Compiles but error at run time	None of the listed options
117 class AutoBox {				
<pre>    public static void main(String args[]) {         int i = 10;         Integer iOb = 100;         i = iOb;         System.out.println(i + " " + iOb);     } } whether this code work properly, if so what would be the result?</pre>	No, Compilation error	No, Runtime error	Yes, 10, 100	Yes, 100, 100

118 Consider the following code and choose the correct option:	Compilation error	true	false	1
<pre>class Test{     public static void main(String args[]){         Long l=0;         System.out.println(l.equals(0));}}</pre>				
119 int l = 0; outer: while (true) {     l++;     inner:     for (int j = 0; j < 10; j++)     {         l += j;         if (j == 3)             continue inner;         break outer;     }     continue outer; } System.out.println(l);				
What will be the result?	3	2	4	1
120 what will be the result of attempting to compile and run the following class?	The code will fail to compile because the compiler will			
<pre>Public class IFTest{ public static void main(String[] args){ int i=10; if(i==10) if(&lt;10) System.out.println("a"); else System.out.println("b"); }}</pre>	The code will not be able to determine which if statement the if statement belongs to	compile correctly and display the letter a,when run	The code will compile correctly and display the letter b,when run	
121 What is the output of the following code :	good	good morning morning ....	compiler error	runtime error
<pre>class try1{ public static void main(String[] args) {     System.out.println("good");     while(false){         System.out.println("morning");     } }}</pre>				

122 Consider the following code and choose the correct output:

```
class Test{
    public static void main(String args[]){
        int num=3; switch(num){
            case 1: case 3: case 4: {
                System.out.println("bat man"); }
            case 2: case 5: {
                System.out.println("spider man"); }break; }
        }
    }
```

bat man

Compilation  
error

bat man  
spider man

spider man

123 Given:

```
int n = 10;
switch(n)
{
    case 10: n = n + 1;
    case 15: n = n + 2;
    case 20: n = n + 3;
    case 25: n = n + 4;
    case 30: n = n + 5;
}
System.out.println(n);
```

What is the value of 'n' after executing the following code?

23

32

Compilation  
Error

124 What will be the output of following code?

```
TreeSet map = new TreeSet();
map.add("one");
map.add("two");
map.add("three");
map.add("four");
map.add("one");
Iterator it = map.iterator();
while (it.hasNext() )
{
    System.out.print( it.next() + " " );
}
```

one two three four three two four one  
four one three two

one two three  
four one

125 public class Test {

```
public static void main(String [] args) {
    int x = 5;
    boolean b1 = true;
    boolean b2 = false;

    if ((x == 4) && !b2 )
        System.out.print("1 ");
    System.out.print("2 ");
    if ((b2 = true) && b1 )
        System.out.print("3 ");
    }
}
```

What is the result?

2

3 2 3

1 2 3

126	HashTable is a sub class of Dictionary	ArrayList is a sub class of Vector	LinkedList is a subclass of ArrayList	Stack is a subclass of Vector
Which of these statements are true?				
127 Given:				
<pre>import java.util.*; public class Explorer3 { public static void main(String[] args) { TreeSet&lt;Integer&gt; s = new TreeSet&lt;Integer&gt;(); TreeSet&lt;Integer&gt; subs = new TreeSet&lt;Integer&gt;(); for(int i = 606; i &lt; 613; i++) if(i%2 == 0) s.add(i); subs = (TreeSet)s.subSet(608, true, 611, true); subs.add(629); System.out.println(s + " " + subs); } }</pre> <p>What is the result?</p>	Compilation fails.	[608, 610, 612, 629] [608, 610]	An exception is thrown at runtime.	[608, 610, 629] [608, 610, 629]
128 What is the output :	good	bad	compile error	run time error
<pre>class try1{ public static void main(String[] args) { int x=1; if(x--) System.out.println("good"); else System.out.println("bad"); } }</pre>				
129 Consider the following code and choose the correct output:	brownie	default	compilation error	default
<pre>class Test{ public static void main(String args[]){ int num='b'; switch(num){ default :{ System.out.print("default");} case 100 : case 'b' : case 'c' : { System.out.println("brownie"); break;} case 200: case 'e': { System.out.println("pastry"); }break; } }}</pre>				

130 Given:

```
int a = 5;
int b = 5;
int c = 5;
if (a > 3)
if (b > 4)
if (c > 5)
c += 1;
else
c += 2;
else
c += 3;
c += 4;
```

What is the value of variable c after executing the following code?

3

5

7

11

131

Given:

```
Float pi = new Float(3.14f);
if (pi > 3) {
System.out.print("pi is bigger than 3. ");
}
else {
System.out.print("pi is not bigger than 3. ");
}
finally {
System.out.println("Have a nice day.");
}
```

What is the result?

Compilation fails.

pi is bigger than 3.

An exception occurs at runtime.

pi is bigger than 3. Have a nice day.

132 Given:

```
public void go() {
String o = "";
z:
for(int x = 0; x < 3; x++) {
for(int y = 0; y < 2; y++) {
if(x==1) break;
if(x==2 && y==1) break z;
o = o + x + y;
}
}
System.out.println(o);
}
```

What is the result when the go() method is invoked?

0 0

0 0 0 1

0 0 0 1 2 0

0 0 1 2 0 2  
1

133 Examine the following code:

```
int count = 1;
while ( _____ )
{
    System.out.print( count + " " );
    count = count + 1;
}
System.out.println( );
```

What condition should be used so that the code prints:

1 2 3 4 5 6 7 8

134 What will be the output of the program?

```
public class Switch2
{
    final static short x = 2;
    public static int y = 0;
    public static void main(String [] args)
    {
        for (int z=0; z < 3; z++)
        {
            switch (z)
            {
                case y: System.out.print("0 "); /* Line 11 */
                case x-1: System.out.print("1 "); /* Line 12 */
                case x: System.out.print("2 "); /* Line 13 */
            }
        }
    }
}
```

135 Given:

```
int x = 0;
int y = 10;
do {
    y--;
    ++x;
} while (x < 5);
System.out.print(x + "," + y);
```

What is the result?



count < 9

count+1 <= 8

count < 8

count != 8

0 1 2

0 1 2 1 2 2

Compilation fails at line 11

Compilation fails at line 12.

5,6

5,5

6,5

6,6

136 What is the output :	<pre>class Test{     public static void main(String[] args) {         int a=5,b=10,c=1;         if(a&gt;c){             System.out.println("success");         }         else{             break;         }     } }</pre>	success	runtime error	compiler error	none of the listed options
137 Consider the following code and choose the correct output:	<pre>public class Test{     public static void main(String[] args) {         int x = 0;         int y = 10;         do {             y--;             ++x;         } while (x &lt; 5);         System.out.print(x + "," + y);     } }</pre>	5,6	5,5	6,5	6,6
138 Consider the following code and choose the correct option:	<pre>class Test{     public static void main(String args[]){         int l=7;         Long L = (Long)l;         System.out.println(L); }}</pre>	7	Compilation error	Compiles but error at run time	None of the listed options
139 Given:	<pre>double height = 5.5; if(height-- &gt;= 5.0)     System.out.print("tall "); if(--height &gt;= 4.0)     System.out.print("average "); if(height-- &gt;= 3.0)     System.out.print("short "); else     System.out.print("very short "); }</pre>	tall	tall short	short	very short
What would be the Result?					
140 Consider the following code and choose the correct option:	<pre>class Test{     public static void main(String args[]){         String hexa = "0xFF";         int number = Integer.decode(hexa);         System.out.println(number); }}</pre>	Compilation error	1515	255	Compiles but error at run time

141 Consider the following code and choose the correct option:

```
int i = 1, j = -1;
switch (i)
{
    case 0, 1: j = 1;
    case 2: j = 2;
    default: j = 0;
}
System.out.println("j = " + j);
```

142 Which of the following statements about arrays is syntactically wrong?

143

What will be the output of following code?

```
import java.util.*;
class I
{
    public static void main (String[] args)
    {
        Object i = new ArrayList().iterator();
        System.out.print((i instanceof List)+",");
        System.out.print((i instanceof
Iterator)+",");
        System.out.print(i instanceof ListIterator);
    }
}
```

144

Given:

```
public static void test(String str) {
int check = 4;
if (check == str.length()) {
System.out.print(str.charAt(check - 1) +", ");
} else {
System.out.print(str.charAt(0) + ", ");
}
}
```

and the invocation:

```
test("four");
test("tee");
test("to");
```

What is the result?

145 What will be the output of the program?

```
int x = 3;
int y = 1;
if (x = y) /* Line 3 */
{
    System.out.println("x =" + x);
}
```

j = -1      j = 0      j = 1  
Person[] p =  
new Person[5];      Person p[5];      Person[] p [];

Compilation fails  
Person p[][] =  
new Person[2][];

Prints: false,      Prints: false,      Prints: false,  
false, false      false, true      true, false  
Prints: false,  
true, true

An exception  
is thrown at  
runtime.

r, t, t,      r, e, o,      Compilation fails.

x = 1      x = 3      Compilation fails.

The code  
runs with no  
output.

146

```
import java.util.SortedSet;
import java.util.TreeSet;

public class Main {

    public static void main(String[] args) {
        TreeSet<String> tSet = new
TreeSet<String>();
        tSet.add("1");
        tSet.add("2");
        tSet.add("3");
        tSet.add("4");
        tSet.add("5");
        SortedSet sortedSet =_____("3");
        System.out.println("Head Set Contains : " +
sortedSet);
    }
}
```

What is the missing method in the code to get the head set of the tree set?

147 Consider the following code and choose the correct output:

```
class Test{
    public static void main(String args[]){
        int num=3;  switch(num){
        default :{
            System.out.print("default");
        case 1: case 3: case 4: {
            System.out.println("apple"); break;
        case 2: case 5: {
            System.out.println("black berry"); }break; }
    }
}}
```

148 Consider the following code and choose the correct option:

```
class Test{
    public static void main(String args[]){
        Long L = null;  long l = L;
        System.out.println(L);
        System.out.println(l);
    }
}}
```



apple  
null 0

tSet.headSet tset.headset headSet HeadSet

compilation error  
default apple default

Compiles but error at run time  
Compilation error 0 null

149 What does the following code fragment write to the monitor?

```
int sum = 21;  
if ( sum != 20 )  
    System.out.print("You win ");  
else  
    System.out.print("You lose ");  
  
System.out.println("the prize.");
```

What does the code fragment prints?

150

Which statements are true about maps?  
(Choose TWO)

151 Which collection implementation is suitable for maintaining an ordered sequence of objects, when objects are frequently inserted in and removed from the middle of the sequence?

152

Choose TWO correct options:

153 What is the output :

```
class One{  
    public static void main(String[] args) {  
        int a=100;  
        if(a>10)  
            System.out.println("M.S.Dhoni");  
        else if(a>20)  
            System.out.println("Sachin");  
        else if(a>30)  
            System.out.println("Virat Kohli");  
    }  
}
```

	You win the prize	You lose the prize.	You win	You lose
150	The return type of the values() method is set	Changes made in the Set view returned by keySet() will be reflected in the original map	The Map interface extends the Collection interface	All keys in a map are unique
151	TreeMap	HashSet	Vector	LinkedList
152	OutputStream is the abstract superclass of all classes that represent an outputstream of bytes.	Subclasses of the class Reader are used to read character streams.	To write characters to an outputstream, you have to make use of the class CharacterOutputStream.	To write an object to a file, you use the class ObjectOutputStream.
153	M.S.Dhoni	M.S.Dhoni Sachin Virat Kohli	Virat Kohli	all of these

154

Which of the following statements is TRUE regarding a Java loop?

A continue statement doesn't transfer control to the test statement of the for loop

An overflow error can only occur in a loop

A loop may have multiple exit points

If a variable of type int overflows during the execution of a loop, it will cause an exception

155 switch(x)

```
{  
    default:  
        System.out.println("Hello");  
}
```

Which of the following are acceptable types for x?

- 1.byte
- 2.long
- 3.char
- 4.float
- 5.Short
- 6.Long

1 ,3 and 5

2 and 4

3 and 5

4 and 6

156

Which are true with respect to finally block?  
(Choose THREE)

Used to release the resources which are obtained in try block.

Writing finally block is optional.

When an exception occurs then a part of try block will execute one appropriate catch block

When no exception occurs then complete try block and finally block will execute but no catch block will execute.

157 What will happen when you attempt to

compile and run the following code?

```
public class Bground extends Thread{  
    public static void main(String argv[]){  
        Bground b = new Bground();  
        b.run();  
    }  
    public void start(){  
        for (int i = 0; i <10; i++){  
            System.out.println("Value of i =  
" + i);  
        }  
    }  
}
```

A compile time error indicating that no run method is defined for the Thread class

A run time error indicating that no run method is defined for the Thread class

Clean compile and at run time the values 0 to 9 are printed out

Clean compile but no output at runtime

158	Given:	<pre>public void testIfA() {     if (testIfB("True")) {         System.out.println("True");     } else {         System.out.println("Not true");     } }  public Boolean testIfB(String str) {     return Boolean.valueOf(str); }</pre>	What is the result when method testIfA is invoked?	true	Not true	An exception is thrown at runtime.	none
159				Both wait() and notify() must be called from a synchronized context.	The wait() method is overloaded to accept a duration	A thread will resume execution as soon as its sleep duration expires.	The notify() method is overloaded to accept a duration
160	public class MyProgram	<pre>{ public static void throwit() {     throw new RuntimeException(); } public static void main(String args[]) {     try     {         System.out.println("Hello world ");         throwit();         System.out.println("Done with try block ");     }     finally     {         System.out.println("Finally executing ");     } } }</pre>	which answer most closely indicates the behavior of the program?	The program will print Hello world, then will print that a RuntimeExce ption has occurred, then will print Finally executing	The program will print Hello world, then will print that a RuntimeExce ption has occurred, then will print Finally executing.	The program will print Hello world, then will print that a RuntimeExce ption has occurred, and then will print Finally executing.	The program will print Hello world, then will print Finally executing, then will print that a RuntimeExce ption has occurred.
161	If a method is capable of causing an exception that it does not handle, it must specify this behavior using throws so that callers of the method can guard themselves against such Exception			false	true		

162

- A) Checked Exception must be explicitly caught or propagated to the calling method
- B) If runtime system can not find an appropriate method to handle the exception, then the runtime system terminates and uses the default exception handler.

Only A is TRUE      Only B is TRUE      Both A and B is TRUE      Both A and B is FALSE

163 public class RTEExcept

```
{  
    public static void throwit ()  
    {  
        System.out.print("throwit ");  
        throw new RuntimeException();  
    }  
    public static void main(String [] args)  
    {  
        try  
        {  
            System.out.print("hello ");  
            throwit();  
        }  
        catch (Exception re )  
        {  
            System.out.print("caught ");  
        }  
        finally  
        {  
            System.out.print("finally ");  
        }  
        System.out.println("after ");  
    }  
}
```

hello throwit  
caught finally  
after      hello throwit  
RuntimeExce  
ption caught  
after      Compilation  
fails

```

164 class s implements Runnable
{
    int x, y;
    public void run()
    {
        for(int i = 0; i < 1000; i++)
            synchronized(this)
        {
            x = 12;
            y = 12;
        }
        System.out.print(x + " " + y + " ");
    }
    public static void main(String args[])
    {
        S run = new S();
        Thread t1 = new Thread(run);
        Thread t2 = new Thread(run);
        t1.start();
        t2.start();
    }
}

```

{ What is the output?

165

What is wrong with the following code?

```

Class MyException extends Exception{}
public class Test{
public void foo() {
try {
bar();
} finally {
baz();
} catch(MyException e) {}
}
public void bar() throws MyException {
throw new MyException();
}
public void baz() throws RuntimeException {
throw new RuntimeException();
}
}

```

DeadLock

Compilation  
Error

Cannot  
determine  
output.

prints 12 12  
12 12

Since the  
method foo()  
does not  
catch the  
exception  
generated by  
the method A try block  
baz(), it must cannot be  
declare the followed by  
RuntimeExce both a catch  
ption in a and a finally  
throws clause block  
An empty catch block is  
not allowed

A catch block  
cannot follow  
a finally block

166 Consider the following code and choose the correct option:

```
class Test{  
    static void test() throws RuntimeException {  
        try { System.out.print("test ");  
        throw new RuntimeException();  
    } catch (Exception ex) {  
        System.out.print("exception ");}  
    } public static void main(String[] args) {  
        try { test(); } catch (RuntimeException ex) {  
            System.out.print("runtime ");}  
        System.out.print("end"); } }
```

test end

test runtime  
end

test  
exception  
runtime end

test  
exception  
end

167

A method declaring that it throws a certain exception class may throw instances of any subclass of that exception class

If an exception is not caught in a method, the method will terminate and normal execution will resume.

An overriding method must declare that it throws the same exception classes as the method it overrides.

The main() method of a program can declare that it throws checked exception

Choose TWO correct options:

168 Which four can be thrown using the throw statement?

- 1.Error
- 2.Event
- 3.Object
- 4.Throwable
- 5.Exception
- 6.RuntimeException

1, 2, 3 and 4    2, 3, 4 and 5    1, 4, 5 and 6

2, 4, 5 and 6

169 class X implements Runnable

```
{  
    public static void main(String args[])  
    {  
        /* Missing code? */  
    }  
    public void run() {}  
}
```

Which of the following line of code is suitable to start a thread ?

Thread t =  
new  
Thread(X);

Thread t =  
new  
Thread(X);

X run = new  
X(); Thread t =  
new  
Thread(run);

Thread t =  
new  
Thread();  
x.run();

170

Given:

```
class X { public void foo() {  
    System.out.print("X "); } }
```

```
public class SubB extends X {  
    public void foo() throws RuntimeException {  
        super.foo();  
        if (true) throw new RuntimeException();  
        System.out.print("B ");  
    }  
    public static void main(String[] args) {  
        new SubB().foo();  
    }  
}
```

What is the result?

X, followed  
by an  
Exception.

No output,  
and an  
Exception is  
thrown.

X, followed  
by an  
Exception,  
followed by  
B.

none

171

What will the output of following code?

```
try  
{  
    int x = 0;  
    int y = 5 / x;  
}  
catch (Exception e)  
{  
    System.out.println("Exception");  
}  
catch (ArithmaticException ae)  
{  
    System.out.println(" Arithmatic Exception");  
}  
System.out.println("finished");
```

172 Which of the following methods are static?

173

Which of the following statements regarding static methods are correct? (2 answers)

finished  
start()

static  
methods are  
difficult to  
maintain,  
because you  
can not  
change their  
implementation.

Exception  
join()  
static

methods can  
be called  
using an  
object  
reference to  
an object of  
the class in  
which this  
method is  
defined.

compilation  
fails  
yield()

static  
methods are  
always  
public,  
because they  
are defined at  
class-level.

ArithmaticExc  
eption  
sleep()

static  
methods do  
not have  
direct access  
to non-static  
methods  
which are  
defined  
inside the  
same class.

174	Consider the following code and choose the correct option:	
175		
176		
177		
178		
179		
180		
181		
182		
183		
184		
185		
186		
187		
188		
189		
190		
191		
192		
193		
194		
195		
196		
197		
198		
199		
200		
201		
202		
203		
204		
205		
206		
207		
208		
209		
210		
211		
212		
213		
214		
215		
216		
217		
218		
219		
220		
221		
222		
223		
224		
225		
226		
227		
228		
229		
230		
231		
232		
233		
234		
235		
236		
237		
238		
239		
240		
241		
242		
243		
244		
245		
246		
247		
248		
249		
250		
251		
252		
253		
254		
255		
256		
257		
258		
259		
260		
261		
262		
263		
264		
265		
266		
267		
268		
269		
270		
271		
272		
273		
274		
275		
276		
277		
278		
279		
280		
281		
282		
283		
284		
285		
286		
287		
288		
289		
290		
291		
292		
293		
294		
295		
296		
297		
298		
299		
300		
301		
302		
303		
304		
305		
306		
307		
308		
309		
310		
311		
312		
313		
314		
315		
316		
317		
318		
319		
320		
321		
322		

```

178 class Trial{
    public static void main(String[] args){
        try{
            System.out.println("One");
            int y = 2 / 0;
            System.out.println("Two");
        }
        catch(RuntimeException ex){
            System.out.println("Catch");
        }
        finally{
            System.out.println("Finally");
        }
    }
}

```

One Two  
Catch Finally One Catch Finally One Two  
Catch

179 Which digit, and in what order, will be printed when the following program is run?

```

Public class MyClass {
    public static void main(String[] args) {
        int k=0;
        try {
            int i=5/k;
        }
        catch(ArithmaticException e) {
            System.out.println("1");
        }
        catch(RuntimeException e) {
            System.out.println("2");
            return;
        }
        catch(Exception e) {
            System.out.println("3");
        }
        finally{
            System.out.println("4");
        }
        System.out.println("5");
    }
}

```

The program will only print 5  
The program will only print 1 and 4 in order  
The program will only print 1,2 and 4 in order  
The program will only print 1 ,4 and 5 in order

180

```

class Trial{
    public static void main(String[] args){
        try{
            System.out.println("Java is portable");
        } } }

```

Java is portable  
We cannot have a try block without a catch block  
We cannot have a try block without a catch / finally block  
Nothing is displayed

```

181 class Animal { public String noise() { return
    "peep"; } }
    class Dog extends Animal {
        public String noise() { return "bark"; }
    }
    class Cat extends Animal {
        public String noise() { return "meow"; }
    }
    class try1{
        public static void main(String[] args){
            Animal animal = new Dog();
            Cat cat = (Cat)animal;
            System.out.println(cat.noise());
        }
    }

```

consider the code above & select the proper output from the options.

182 Given:

```

class X implements Runnable
{
    public static void main(String args[])
    {
        /* Some code */
    }
    public void run() {}
}

```

Which of the following line of code is suitable to start a thread ?

183

Which statement is true?

184 Consider the following code and choose the correct option:

```

class Test{
    static void display(){
        throw new RuntimeException();
    }
    public static void main(String args[]){
        try{display();
        }catch(Exception e){}
        catch(RuntimeException re){}
        finally{System.out.println("exit");}
    }
}

```

bark  
X run = new X(); Thread t = new Thread(run); t.start();

meow  
Thread t = new Thread(); Thread(X); x.run();

Compilation fails

An exception is thrown at runtime.

A static method cannot be synchronized .

Variables can be protected from concurrent access problems by marking them with the synchronized keyword.

When a thread sleeps, it releases its locks

exit

Compiles and no output

Compiles but exception at runtime

185

Given:

```
public class ExceptionTest
{
    class TestException extends Exception {}
    public void runTest() throws TestException
    {}
    public void test() /* Line X */
    {
        runTest();
    }
}
```

At Line X, which code is necessary to make the code compile?

No code is necessary	throws Exception	throw Exception	throws RuntimeException
Implement java.lang.Runnable and implement the run() method.	Extend java.lang.Thread and override the run() method.	Implement java.lang.Thread and override the start() run() method.	Extend java.lang.Runnable and override the start() method.

186

Which two can be used to create a new Thread?

An Error that might be thrown in a try block.	Multiple catch statements must have at least one corresponding catch block.	Except in case of VM shutdown, if a try block starts to execute, a corresponding finally block will always start to execute.

Choose the correct option:

188 class PropagateException{  
 public static void main(String[] args){  
 try{  
 method();  
 System.out.println("method() called");  
 }  
 catch(ArithmeticException ex){  
 System.out.println("Arithmatic Exception");  
 }  
 catch(RuntimeException re){  
 System.out.println("Runtime Exception");  
 }  
 static void method(){  
 int y = 2 / 0;  
 }  
 consider the code above & select the proper output from the options.

Arithmetic Exception	Runtime Exception	Runtime Exception	compilation error

189 Given:

```
static void test() {  
    try {  
        String x = null;  
        System.out.print(x.toString() + " ");  
    }  
    finally { System.out.print("finally "); }  
}  
public static void main(String[] args) {  
    try { test(); }  
    catch (Exception ex) {  
        System.out.print("exception "); }  
}
```

What is the result?

190 Given two programs:

```
1. package pkgA;  
2. public class Abc {  
3.     int a = 5;  
4.     protected int b = 6;  
5.     public int c = 7;  
6. }  
  
3. package pkgB;  
4. import pkgA.*;  
5. public class Def {  
6.     public static void main(String[] args) {  
7.         Abc f = new Abc();  
8.         System.out.print(" " + f.a);  
9.         System.out.print(" " + f.b);  
10.        System.out.print(" " + f.c);  
11.    }  
12. }
```

What is the result when the second program is run? (Choose all that apply)

	null	Compilation fails.	finally exception	finally
	Compilation fails with an error on line 9	Compilation fails with an error on line 5 followed by an exception 7	Compilation fails with an error on line 7	Compilation fails with an error on line 8

191

Consider the following code:

```
System.out.print("Start ");
try
{
    System.out.print("Hello world");
    throw new FileNotFoundException();
}
System.out.print(" Catch Here "); /* Line 7 */
catch(EOFException e)
{
    System.out.print("End of file exception");
}
catch(FileNotFoundException e)
{
    System.out.print("File not found");
}
```

given that EOFException and FileNotFoundException are both subclasses of IOException. If this block of code is pasted in a method, choose the best option.

192

Which of the following statements is true?

193 Consider the following code and choose the correct option:

```
int array[] = new int[10];
array[-1] = 0;
```

The code will not compile.

catch(X x)  
can catch  
subclasses of  
X where X is  
a subclass of  
Exception.

The Error  
class is a  
RuntimeExce  
ption.

Code output:  
Start Hello  
world File Not  
Found

Code output:  
Start Hello  
world End of  
file exception.

Any statement  
that can  
throw an  
Exception

Error must be  
enclosed in a  
try block.

Code output:

Start Hello  
world Catch  
Here File not  
found.

Any statement  
that can  
throw an  
Exception

must be  
enclosed in a  
try block.

compiles  
successfully

does not  
compile

runtime error  
none of the  
listed options

194 What will be the output of the program?

```
public class RTEexcept
{
    public static void throwit ()
    {
        System.out.print("throwit ");
        throw new RuntimeException();
    }
    public static void main(String [] args)
    {
        try
        {
            System.out.print("hello ");
            throwit();
        }
        catch (Exception re )
        {
            System.out.print("caught ");
        }
        finally
        {
            System.out.print("finally ");
        }
        System.out.println("after ");
    }
}
```

hello throwit caught	Compilation fails	hello throwit RuntimeExce ption caught after	hello throwit caught finally after
volatile	synchronized	final	private

195 What is the keyword to use when the access of a method has to be restricted to only one thread at a time

196 Consider the following code and choose the correct option:

```
class Test{
    public static void parse(String str) {
        try { int num = Integer.parseInt(str);
        } catch (NumberFormatException nfe) {
            num = 0; } finally {
        System.out.println(num);
    } } public static void main(String[] args) {
        parse("one"); }
```

NumberForm atException thrown at 0 runtime	Compilation fails	ParseException thrown at runtime
---	----------------------	--

197 public static void parse(String str) {  
try {  
float f = Float.parseFloat(str);  
} catch (NumberFormatException nfe) {  
f = 0;  
} finally {  
System.out.println(f);  
}  
}  
public static void main(String[] args) {  
parse("invalid");  
}

A  
NumberForm  
ParseExcepti  
on is thrown  
by the parse  
method at  
runtime.

<p>198 Given the following program, which statements are true? (Choose TWO)</p> <pre>Public class Exception {     public static void main(String[] args) {         try {             if(args.length == 0) return;             System.out.println(args[0]);         }finally {             System.out.println("The end");         }     } }</pre>	<p>If run with no arguments, the program will produce no output</p>	<p>If run with one argument, the program will print the given argument followed by "The end"</p>	<p>If run with one arguments, the program will simply print the given argument</p>
<p>199 Which can appropriately be thrown by a programmer using Java SE technology to create a desktop application?</p>	<p>ClassCastException</p>	<p>NullPointerException</p>	<p>NoClassDefFoundError</p>
<p>200 Which of the following is a checked exception?</p>	<p>ArithmeticException</p>	<p>IOException</p>	<p>NumberFormatException</p>
<p>201 Given:</p> <pre>11. class A { 12.     public void process() { System.out.print("A,"); 13.     class B extends A { 14.         public void process() throws IOException { 15.             super.process(); 16.             System.out.print("B,"); 17.             throw new IOException(); 18. } 19.     public static void main(String[] args) { 20.         try { new B().process(); } 21.         catch (IOException e) { System.out.println("Exception"); 22.     } What is the result?</pre>	<p>Exception</p>	<p>A,B,Exception</p>	<p>Compilation fails because of an error in line 20.</p>
<p>202</p>	<p>The notifyAll() method must be called from a synchronized context</p>	<p>To call sleep(), a thread must own the lock on the object</p>	<p>The notify() method is defined in class java.lang.Thread</p>
<p>Which statement is true?</p>	<p>The notify() method causes a thread to immediately release its locks.</p>		
<p>203 class Trial{</p> <pre>public static void main(String[] args){ try{ System.out.println("Try Block"); } finally{ System.out.println("Finally Block"); } }</pre>	<p>Try Block</p>	<p>Try Block Finally Block</p>	<p>Finally Block Try Block</p>

204 consider the code & choose the correct output:  
`class Threads2 implements Runnable {`

```
public void run() {  
    System.out.println("run.");  
    throw new RuntimeException("Problem");  
}  
public static void main(String[] args) {  
    Thread t = new Thread(new Threads2());  
    t.start();  
    System.out.println("End of method.");  
}
```

205 The exceptions for which the compiler doesn't enforce the handle or declare rule

206 Consider the code below & select the correct output from the options:

```
public class Test{  
    Integer i;  
    int x;  
    Test(int y){  
        x=i+y;  
        System.out.println(  
    }  
}
```

```
public static void main(String[] args) {  
    new Test(new Integer(5));  
}  
} // class
```

207 Given:

```
public class TestSeven extends Thread {  
    private static int x;  
    public synchronized void doThings() {  
        int current = x;  
        current++;  
        x = current;  
    }  
    public void run() {  
        doThings();  
    }  
}
```

Which statement is true?

	End of method.	End of method. run.
run		
java.lang.Run	java.lang.Run	java.lang.Run
timeException	timeException	timeException
n: Problem	n: Problem	n: Problem
Unchecked exceptions	Exception	all of these

Compilation error	Compiles but error at run time
-------------------	--------------------------------

Synchronizing the run() method would make the class thread-safe.

Declaring the doThings() method as static would make the class thread-safe.

End of  
method. run.  
java.lang.Run  
timeExceptio  
n: Problem

10

An exception  
is thrown at  
runtime.

208 Consider the following code and choose the correct option:

```
class Test{  
    static void display(){  
        throw new RuntimeException();  
    }  
    public static void main(String args[]){  
        try{ display(); }catch(Exception e){  
            throw new NullPointerException();}  
        finally{try{ display(); }  
            catch(NullPointerException e){  
                System.out.println("caught");}  
                System.out.println("exit");}}}
```

caught exit    exit

Compilation fails

Compiles but exception at runtime

An object will not be garbage collected as long as it is possible for a live thread to access it through a reference.

209

An object is deleted as soon as there are no more references that denote the object. The finalize() method will eventually be called on every object. The finalize() method will never be called more than once on an object.

Which statements describe guaranteed behaviour of the garbage collection and finalization mechanisms? (Choose TWO)

210 Which statement is true?

- A. A class's finalize() method CANNOT be invoked explicitly.
- B. super.finalize() is called implicitly by any overriding finalize() method.
- C. The finalize() method for a given object is called no more than once by the garbage collector.
- D. The order in which finalize() is called on two objects is based on the order in which the two objects became finalizable.

A

B

C

D

211

Which of the following allows a programmer to destroy an object x?

x.delete()

x.finalize()

Runtime.getRuntime().gc()

Only the garbage collection system can destroy an object.

```

212 class X2
{
    public X2 x;
    public static void main(String [] args)
    {
        X2 x2 = new X2(); /* Line 6 */
        X2 x3 = new X2(); /* Line 7 */
        x2.x = x3;
        x3.x = x2;
        x2 = new X2();
        x3 = x2; /* Line 11 */
    }
}

```

after line 11 runs, how many objects are eligible for garbage collection?

0                  1                  2                  3

213 Given :

```

public class MainOne {
    public static void main(String args[]) {
        String str = "this is java";
        System.out.println(removeChar(str,'s'));
    }
}

```

```

public static String removeChar(String s,
char c) {
    String r = "";
    for (int i = 0; i < s.length(); i++) {
        if (s.charAt(i) != c)
            r += s.charAt(i);
    }
    return r;
}

```

} What would be the result?

214

How can you force garbage collection of an object?

This is java	Thi is java Call System.gc()	This i java	Thi i java
Set all references to the object to new values(null, for example).	passing in a reference to the object to be garbage collected	Call System.gc()	Garbage collection cannot be forced

215 Consider the following code and choose the correct option:

```
public class X {  
    public static void main(String [] args) {  
        X x = new X();  
        X x2 = m1(x); /* Line 6 */  
        X x4 = new X();  
        x2 = x4; /* Line 8 */  
        doComplexStuff(); }  
    static X m1(X mx) {  
        mx = new X();  
        return mx; }  
}
```

After line 8 runs. how many objects are eligible for garbage collection?

216 interface interface\_1 {

```
    void f1();  
}  
class Class_1 implements interface_1 {  
    void f1() {  
        System.out.println("From F1 function in  
Class_1 Class");  
    }  
}  
public class Demo1 {  
    public static void main(String args[]) {  
        Class_1 o11 = new Class_1();  
        o11.f1();  
    }  
}
```

217 Given:

```
class A {  
    final void meth() {  
        System.out.println("This is a final  
method.");  
    }  
}  
class B extends A {  
    void meth() {  
        System.out.println("Illegal!");  
    }  
}  
class MyClass8{  
    public static void main(String[] args) {  
        A a = new A();  
        a.meth();  
        B b = new B();  
        b.meth();  
    }  
}
```

What would be the result?

0	1	2	3	
From F1 function in Class_1 Class	Create an object for Interface only	Compile time error	Runtime Error	

This is a final method	This is a final method	Some error message	Compilation error	illegal Some error message
illegal	illegal	Some error message	Compilation error	

218 Which Man class properly represents the relationship "Man has a best friend who is a Dog"?

- A)class Man extends Dog { }
- B)class Man implements Dog { }
- C)class Man { private BestFriend dog; }
- D)class Man { private Dog bestFriend; }

A

B

C

D

219 What will be the output of the program?

```
class SuperClass
{
    public Integer getLength()
    {
        return new Integer(4);
    }
}

public class SubClass extends SuperClass
{
    public Long getLength()
    {
        return new Long(5);
    }

    public static void main(String[] args)
    {
        SuperClass sp = new SuperClass();
        SubClass sb = new SubClass();
        System.out.println(
            sp.getLength().toString() + "," +
            sub.getLength().toString());
    }
}
```

4, 4

4, 5

5, 4

Compilation fails

220

Consider the code below & select the correct output from the options:

```
abstract class Ab{ public int getN(){return 0;}}
class Bc extends Ab{ public int getN(){return
7;}}
class Cd extends Bc { public int getN(){return
47;}}
class Test{
    public static void main(String[] args) {
        Cd cd=new Cd();
        Bc bc=new Cd();
        Ab ab=new Cd();
        System.out.println(cd.getN()+" "+
bc.getN()+" "+ab.getN());}}
```

0 0 0

47 7 0

Compilation error

47 47 47

```

221 interface A{}
    class B implements A{}
    class C extends B{}
    public class Test extends C{
        public static void main(String[] args) {
            C c=new C();
            /* Line6 */}
        
```

Which code, inserted at line 6, will cause a java.lang.ClassCastException?

B b=c;      A a2=(B)c;      C c2=(C)(B)c;      A a1=(Test)c;

222 Given :

What would be the result of compiling and running the following program?

```

// Filename: MyClass.java
public class MyClass {
    public static void main(String[] args) {
        C c = new C();
        System.out.println(c.max(13, 29));
    }
}

class A {
    int max(int x, int y) { if (x>y) return x; else
        return y; }
}

class B extends A{
    int max(int x, int y) { return super.max(y, x) - 10; }
}

class C extends B {
    int max(int x, int y) { return super.max(x+10, y+10); }
}

```

The code will fail to compile because the max() method in B passes the arguments in the call to a super.max(y, x) in the wrong order.

The code will fail to compile because a call to a max() method is ambiguous.

The code will compile and print 23,

The code will compile and print 29, when run.

223 The concept of multiple inheritance is implemented in Java by

- (A) extending two or more classes
- (B) extending one class and implementing one or more interfaces
- (C) implementing two or more interfaces
- (D) all of these

(A)      (A) & (C)      (D)      (B) & (C)

224

Given:

```

interface DoMath
{
    double getArea(int r);
}

interface MathPlus
{
    double getVolume(int b, int h);
}

/* Missing Statements ? */
Select the correct missing statements.

```

abstract class AllMath implements DoMath,

interface AllMath extends DoMath {

class AllMath implements MathPlus {

public double getArea(int rad) { return rad \* rad \* 3.14; } }

double getArea(int r);

double getVol(int x, int y); }

double MathPlus {

getArea(int rad); }

225 Consider the following code and choose the correct option:

```
class A{
    void display(byte a, byte b){
        System.out.println("sum of byte"+(a+b));
    }
    void display(int a, int b){
        System.out.println("sum of int"+(a+b));
    }
    public static void main(String[] args) {
        new A().display(3, 4);
    }
}
```

sum of byte 7      Compilation error

sum of int7

Compiles but  
error at  
runtime

226

Consider the following code and choose the correct option:

```
interface Output{
    void display();
    void show();
}
class Screen implements Output{
    void display(){ System.out.println("display");}
}
public static void main(String[] args) {
    new Screen().display();
}
```

display

Compilation  
error

Compiles but  
error at run  
time

Runs but no  
output

227

```
class Animal {
    void makeNoise() {System.out.println("generic
noise");}
}
class Dog extends Animal {
    void makeNoise() {System.out.println("bark");}
    void playDead() { System.out.println("roll
over");}
}
class CastTest2 {
    public static void main(String [] args) {
        Dog a = (Dog) new Animal();
        a.makeNoise();
    }
}
```

consider the code above & select the proper output from the options.

run time error generic noise bark

compile error

228

Consider the following code and choose the correct option:

```
interface employee{  
    void saldetails();  
    void perdetails();  
}  
  
abstract class perEmp implements employee{  
    public void perdetails(){  
        System.out.println("per details"); }  
    class Programmer extends perEmp{  
        public void saldetails(){  
            perdetails();  
            System.out.println("sal details"); }  
        public static void main(String[] args) {  
            perEmp emp=new Programmer();  
            emp.saldetails(); }  
}
```

sal details

sal details  
per details

compilation  
error

per details  
sal details

229 Consider the code below & select the correct output from the options:

```
class A{  
    static int sq(int n){  
        return n*n; }  
}  
  
public class Test extends A{  
    static int sq(int n){  
        return super.sq(n); }  
    public static void main(String[] args) {  
        System.out.println(new Test().sq(3)); }  
}
```

3 Compilation  
error

Compiles but  
error at run  
time

9

230

Given:

```
public static void main( String[] args ) {  
    SomeInterface x; ... }  
Can an interface name be used as the type of  
a variable
```

No—a  
variable must  
always be an  
object  
reference  
type

No—a  
variable must  
always be an  
object  
reference  
type or a  
primitive type

No—a  
variable must  
always be a  
primitive type

Yes—the  
variable can  
refer to any  
object whose  
class  
implements  
the interface

231 Consider the following code and choose the correct option:

```
interface A{  
    int i=3;  
}  
  
interface B{  
    int i=4;  
}  
  
class Test implements A,B{  
    public static void main(String[] args) {  
        System.out.println(i);  
    }  
}
```

3

compilation  
error

Compiles but  
error at  
runtime

232 Given the following classes and declarations, which statements are true?

```
// Classes  
class A {  
    private int i;  
    public void f() { /* ... */ }  
    public void g() { /* ... */ }  
}  
class B extends A{  
    public int j;  
    public void g() { /* ... */ }  
}  
  
// Declarations:  
A a = new A();  
B b = new B();
```

Select the three correct answers.

233 Which declaration can be inserted at (1) without causing a compilation error?

```
interface MyConstants {  
    int r = 42;  
    int s = 69;  
    // (1) INSERT CODE HERE  
}
```

234 What is the output for the following code:

```
abstract class One{  
    private abstract void test();  
}  
class Two extends One{  
    void test(){  
        System.out.println("hello");  
    }  
}  
class Test{  
    public static void main(String[] args){  
        Two obj = new Two();  
        obj.test();  
    }  
}
```

The B class is a subclass of A.

int total =  
total + r + s;

run time exception

The statement b.f(); is legal

final double circumference = 2 \*  
Math.PI \* r; CODE = 31337;

compile time error

The statement a.j = 5; is legal

protected int

hello

The statement a.g(); is legal

int AREA = r \* s;

hellohello

235

Consider the code below & select the correct output from the options:

```
class Money {  
    private String country = "Canada";  
    public String getC() { return country; } }  
class Yen extends Money {  
    public String getC() { return super.country; }  
    public static void main(String[] args) {  
        System.out.print(new Yen().getC()); } }
```

Canada

Compiles but  
error at run  
time

null

236

When we use both implements & extends keywords in a single java program then what is the order of keywords to follow?

we must use always extends and later we must use implements keyword.

we must use always implements and later we must use extends keyword.

we can use in any order its not at all a problem

implements can't be used together

237 Consider the code below & select the correct output from the options:

```
1. public class Mountain {  
2. protected int height(int x) { return 0; }  
3. }  
4. class Alps extends Mountain {  
5. // insert code here  
6. }
```

Which five methods, inserted independently at line 5, will compile? (Choose three.)

- A. public int height(int x) { return 0; }
- B. private int height(int x) { return 0; }
- C. private int height(long x) { return 0; }
- D. protected long height(long x) { return 0; }
- E. protected long height(int x) { return 0; }

A,B,E

A,C,D

B,D,E

C,D,E

238 Given:

```
interface DeclareStuff {  
public static final int Easy = 3;  
void doStuff(int t);}  
public class TestDeclare implements  
DeclareStuff {  
public static void main(String [] args) {  
int x = 5;  
new TestDeclare().doStuff(++x);  
}  
void doStuff(int s) {  
s += Easy + ++s;  
System.out.println("s " + s);  
}  
}
```

) What is the result?

s 14

s 16

s 10

Compilation fails.

239 Given:

```
interface A { public void methodA(); }  
interface B { public void methodB(); }  
interface C extends A,B{ public void  
methodC(); } //Line 3  
class D implements B {  
public void methodB() {} //Line 5  
}  
class E extends D implements C { //Line 7  
public void methodA() {}  
public void methodB() {} //Line 9  
public void methodC() {}  
}  
What would be the result?
```

Compilation fails, due to an error in line 3

If you define D e = (D)  
(new E()), then e.methodB()  
invokes the version of methodB()  
defined at line 9

Compilation fails, due to an error in line 7

If you define D e = (D)  
(new E()), then e.methodB()  
invokes the version of methodB()  
defined at line 5

240	Which of the following statements is true regarding the super() method?	It can only be used in the parent's constructor	Only one child class can use it	It must be used in the last statement of the constructor. It must be used in the first statement of the constructor.
241	Consider the following code and choose the correct option: <pre>interface Output{     void display();     void show(); }  class Screen implements Output{     void show() {System.out.println("show");}     void display(){ System.out.println("display");} }public static void main(String[] args) {     new Screen().display();}</pre>	display	Compilation error	Compiles but error at run time Runs but no output
242	Consider the following code and choose the correct option: <pre>class A{     void display(){ System.out.println("Hello A");} }  class B extends A{     void display(){         System.out.println("Hello B"); } }  public class Test {     public static void main(String[] args) {         B b=(B) new A();         b.display(); }}</pre>	Hello A	Compilation error	Hello B Compiles but error at runtime
243	Consider the following code: <pre>// Class declarations: class Super {} class Sub extends Super {}  // Reference declarations: Super x; Sub y;</pre> Which of the following statements is correct for the code: y = (Sub) x?	Illegal at compile time	Definitely legal at runtime, but the cast operator (Sub) is not strictly needed.	Definitely legal at runtime, and the cast operator (Sub) is needed.
244	Given: 11. class ClassA {} 12. class ClassB extends ClassA {} 13. class ClassC extends ClassA {} and: 21. ClassA p0 = new ClassA(); 22. ClassB p1 = new ClassB(); 23. ClassC p2 = new ClassC(); 24. ClassA p3 = new ClassB(); 25. ClassA p4 = new ClassC(); Which TWO are valid? (Choose two.)	p0 = p1; p2 = p4;	p1 = (ClassB)p3;	p1 = p2;

245 Consider the following code and choose the correct option:

```
abstract class Car{  
    abstract void accelerate();  
}  
class Lamborghini extends Car{  
    @Override  
    void accelerate() {  
        System.out.println("90 mph"); }  
    void nitroBooster(){  
        System.out.print("150 mph"); }  
    public static void main(String[] args) {  
        Car mycar=new Lamborghini();  
        Lamborghini lambo=(Lamborghini) mycar;  
        lambo.nitroBooster();}}
```

150 mph	Compilation error	90 mph	Compiles but error at runtime
---------	-------------------	--------	-------------------------------

246 Consider the following code and choose the correct option:

```
class A{  
    void display(){ System.out.println("Hello A"); }  
}  
class B extends A{  
    void display(){  
        System.out.println("Hello B"); }  
}  
public class Test {  
    public static void main(String[] args) {  
        A a=new B();  
        B b= (B)a;  
        b.display(); }}
```

Hello A	Compilation error	Hello B	Compiles but error at runtime
---------	-------------------	---------	-------------------------------

247

A class Animal has a subclass Mammal.  
Which of the following is true:

Because of single inheritance, Mammal can have no subclasses	Because of single inheritance, Mammal can have no other parent than Animal	Because of single inheritance, Animal can have only one subclass	Because of single inheritance, Mammal can have no siblings.
--	--	--	---

248

```
class Animal {  
void makeNoise() {System.out.println("generic  
noise"); }  
}  
class Dog extends Animal {  
void makeNoise() {System.out.println("bark"); }  
void playDead() { System.out.println("roll  
over"); }  
}  
class CastTest2 {  
public static void main(String [] args) {  
Animal a = new Dog();  
a.makeNoise();  
}  
}
```

consider the code above & select the proper  
output from the options.

run time error generic noise bark compile error

249 What will be the result when you try to  
compile and run the following code?

```
class Base1 {  
Base1() {  
int i = 100;  
System.out.println(i);  
}  
}  
  
public class Pri1 extends Base1 {  
static int i = 200;  
  
public static void main(String argv[]) {  
Pri1 p = new Pri1();  
System.out.println(i);  
}  
}
```

Error at  
compile time 200 100 followed  
by 200 100

250 What is the output :

```
interface A{  
void method1();  
void method2();  
}  
class Test implements A{  
public void method1(){  
System.out.println("hello");}  
class RunTest{  
public static void main(String[] args){  
Test obj = new Test();  
obj.method1();  
}}
```

hello compile error runtime error none

251 Given the following classes and declarations, which statements are true?

```
// Classes  
class Foo {  
    private int i;  
    public void f() { /* ... */ }  
    public void g() { /* ... */ }  
}  
class Bar extends Foo {  
    public int j;  
    public void g() { /* ... */ }  
}  
  
// Declarations:  
Foo a = new Foo();  
Bar b = new Bar();
```

252 Given a derived class method which overrides one of it's base class methods. With derived class object you can invoke the overridden base method using:

253 Consider the following code and choose the correct option:

```
abstract class Car{  
    abstract void accelerate();  
}  
class Lamborghini extends Car{  
    @Override  
    void accelerate() {  
        System.out.println("90 mph");  
    }  
    void nitroBooster(){  
        System.out.print("150 mph"); }  
    public static void main(String[] args) {  
        Car mycar=new Lamborghini();  
        mycar.nitroBooster(); }}
```

254

Given:

```
class Pizza {  
    java.util.ArrayList toppings;  
    public final void addTopping(String topping) {  
        toppings.add(topping);  
    }  
    public class PepperoniPizza extends Pizza {  
        public void addTopping(String topping) {  
            System.out.println("Cannot add Toppings");  
        }  
    }  
    public static void main(String[] args) {  
        Pizza pizza = new PepperoniPizza();  
        pizza.addTopping("Mushrooms");  
    }  
}
```

What is the result?

The Bar class is a subclass of Foo.	super keyword	Compilation error	Compilation fails.
The statement a.j = 5; is legal. by creating an instance of the base class	this keyword	Compiles but error at run time	Cannot add Toppings

The statement a.j = 5; is legal.

by creating an instance of the base class

The statement b.f(); is legal.

cannot call a.g(); because it is overridden in derived class

The statement a.g(); is legal.

because it is overridden in derived class

90 mph

150 mph

A NullPointerException is thrown

255 Consider the following code and choose the correct option:

```
interface console{
    int line=10;
    void print();}

class a implements console{
    void print(){
        System.out.print("A");
    }

    public static void main(String ar[]){
        new a().print();
    }
}
```

256

Which of these field declarations are legal in an interface? (Choose all applicable)

257 Given :

```
Day d;
BirthDay bd = new BirthDay("Raj", 25);
d = bd; // Line X
```

Where Birthday is a subclass of Day. State whether the code given at Line X is correct:

258

Select the correct statement:

259

Choose the correct declaration of variable in an interface:

260 Consider the following code and choose the correct option:

```
abstract class Fun{
    void time(){
        System.out.println("Fun Time"); }

    class Run extends Fun{
        void time(){
            System.out.println("Fun Run"); }

        public static void main(String[] args) {
            Fun f1=new Run();
            f1.time(); }}
```

	A	Compilation error	Compiles but error at run time	Runs but no output
255		public int answer = 42;	final static int answer = 42;	private final static int answer = 42;
256		No—there must always be an exact match between the variable and the object.	No—but a object of parent type can be assigned to a variable of child type.	Yes—an object can be assigned to a reference variable.
257				Yes—any object can be assigned to any reference variable.
258		A super() or this() call must always be provided explicitly as the first statement in the body of a constructor.	If both a subclass and its superclass do not have any declared constructors, the implicit default constructor of the subclass will call super() when run.	If neither super() nor this() is declared as the first statement in the body of a constructor, this() will implicitly be inserted as the first statement.
259		public final data type varaibale=inti alization;	static final data type static data type variable;	final data type varaiablename =intialization;
260	Fun Time	Compilation error	Fun Run	Compiles but error at runtime

261

```
interface Vehicle{  
void drive();  
}  
final class TwoWheeler implements Vehicle{  
int wheels = 2;  
public void drive(){  
System.out.println("Bicycle");  
}  
}  
class ThreeWheeler extends TwoWheeler{  
public void drive(){  
System.out.println("Auto");  
}  
}  
class Test{  
public static void main(String[] args){  
ThreeWheeler obj = new ThreeWheeler();  
obj.drive();  
}  
}
```

consider the code above & select the proper output from the options.

Auto

Bicycle Auto compile error runtime error

262

Consider the following code and choose the correct option:

```
interface employee{  
void saldetails();  
void perdetails();  
}  
abstract class perEmp implements employee{  
public void perdetails(){  
System.out.println("per details"); }  
class Programmer extends perEmp{  
public static void main(String[] args) {  
perEmp emp=new Programmer();  
emp.saldetails(); }  
}
```

263 All data members in an interface are by default

sal details  
abstract and final

sal details  
per details  
public and abstract

compilation  
error  
public ,static  
and final

per details  
sal details  
default and abstract

264 Consider the following code and choose the correct option:

```
interface console{  
int line;  
void print();}  
class a implements console{  
public void print(){  
System.out.print("A");}  
public static void main(String ar[]){  
new a().print();}}
```

A

Compiles but  
error at run  
time

Runs but no  
output

265

Which of the following is correct for an abstract class. (Choose TWO)

An abstract class is one which contains some general purpose methods and some undefined methods.

An abstract class is one which contains only static methods.

Abstract class can be declared final.

266

Which of the following defines a legal abstract class?

abstract class Vehicle {  
 abstract void display(); }  
class abstract Vehicle {  
 abstract void display(); }  
System.out.println("Car"); }}

267 Consider the code below & select the correct output from the options:

```
class Mountain{  
    int height;  
    protected Mountain(int x) { height=x; }  
    public int getH(){return height;}}
```

```
class Alps extends Mountain{  
    public Alps(int h){ super(h); }  
    public Alps(){ this(100); }  
    public static void main(String[] args) {  
        System.out.println(new Alps().getH());  
    }  
}
```

268 Consider the given code and select the correct output:

```
class SomeException {  
}  
  
class A {  
    public void doSomething() {}  
}  
  
class B extends A {  
    public void doSomething() throws  
        SomeException {}  
}
```

Compiles but error at run time

100

Compiles but error at run time

Compiles but no output

Compilation of class A will fail.  
Compilation of class B will succeed.

Compilation of both classes A & B will fail.

Compilation of both classes will succeed.

Compilation of class A will succeed.

269

Is it possible if a class definition implements two interfaces, each of which has the same definition for the constant?

No—if a class implements several interfaces, each constant must be defined in only one interface

No—a class may not implement more than one interface

Yes—either of the two variables can be accessed through : interfaceName.variableName

Yes—since the definitions are the same it will not matter

270

Select the correct statement:

271 Consider the following code and choose the correct option:

```
class A{
    void display(){ System.out.println("Hello A");}
}
class B extends A{
    void display(){
        System.out.println("Hello B"); }
}
public class Test {
    public static void main(String[] args) {
        A a=new B();
        B b= a;
        b.display(); }}
```

Private methods cannot be overridden in subclasses

A subclass can override any method in a superclass

An overriding method can declare that it throws checked exceptions that are not thrown by the method it is overriding

The parameter list of an overriding method can be a subset of the parameter list of the method that it is overriding

272

Which of the following option gives one possible use of the statement 'the name of the public class should match with its file name'?

To maintain the uniform standard

Compilation error

Hello B

Compiles but error at runtime

Helps the compiler to find the source file that corresponds to a class, when it does not find a class file while compiling

Helps JVM to find and execute the classes

Helps Javadoc to build the Java Documentati on easily

273

Which of the following statement gives the use of CLASSPATH?

Holds the location of Core Java Class Library (Bootstrap classes)	Holds the location of Java Extension Library	Holds the location of User Defined classes, packages and JARs	Holds the location of Java Software
---	--	---	-------------------------------------

274

Which of the following are true about packages? (Choose 2)

Packages can contain only Java Source files	Packages can contain both Classes and Interfaces (Compiled Classes)	Packages can contain non-java elements such as images, xml files etc.	Sub packages should be declared as private in order to deny importing them
---	---	---	--

275 Which of the following options give the valid argument types for main() method? (Choose 2)

String [][]args	String args[]	String[] args[]	String[] args
-----------------	---------------	-----------------	---------------

276

Which of the following options give the valid package names? (Choose 3)

dollarpack.\$p	_score.pack.	p@ckage.sub
ack.\$\$pack	\$\$.\$. \$\$.	p@ckage.inn

277

Which of the following statements are true regarding java.lang.Object class? (Choose 2)

Object class is an abstract class	Object class cannot be instantiated directly	Object class has the core methods for thread synchronization	Implementation in Collection framework
-----------------------------------	--	--	--

278

The term 'Java Platform' refers to \_\_\_\_\_.

Java Compiler (Javac)	Java Runtime Environment (JRE)	Database Connectivity (JDBC)	Java Debugger
-----------------------	--------------------------------	------------------------------	---------------

279

Which of the following methods are needed for loading a database driver in JDBC?

registerDriver () method	Class.forName ()	Class.forName ()	getConnectio n
--------------------------	------------------	------------------	----------------

280

how to register driver class in the memory?

Using the static method registerDriver () method which is available in DriverManager Class.	Either forName() or registerDriver ()	None of the given options
---	---------------------------------------	---------------------------

281	Give Code snippet: <pre>// Somecode ResultSet rs = st.executeQuery("SELECT * FROM survey");  while (rs.next()) {     String name = rs.getString("name");     System.out.println(name); }  rs.close(); // somecode</pre> <p>} What should be imported related to ResultSet?</p>	java.sql.ResultSet	java.sql.Drivers	java.sql.DriverManager	java.sql.Connection
282	Consider the following code & select the correct option for output. <pre>String sql ="select empno,ename from emp"; PreparedStatement pst=cn.prepareStatement(sql); System.out.println(pst.toString()); ResultSet rs=pst.executeQuery(); System.out.println(rs.getString(1)+ " "+rs.getString(2));</pre>	will show first employee record	Compilation error	Compiles but error at run time	Compiles but no output
283	Which of the following methods finds the maximum number of connections that a specific driver can obtain?	Connection.getMaxConnections	ResultSetMetaData.getMaximumConnections	DatabaseMetaData.getMaxConnections	Database.getMaxConnections
284	By default all JDBC transactions are autocommit. State TRUE/FALSE.	true	false		
285	getConnection() is method available in?	DriverManager Class	Driver Interface	ResultSet Interface	Statement Interface
286	A) By default, all JDBC transactions are auto commit B) PreparedStatement suitable for dynamic sql and requires one time compilation C) with JDBC it is possible to fetch information about the database	Only A and B is TRUE	Only B and C is True	Both A and C is TRUE	All are TRUE
287	What is the use of wasNull() in ResultSet interface?	There is no such method in ResultSet interface	It returns true when last read column contain SQL NULL else returns false	It returns int value as mentioned below: > 0 if many columns Contain Null Value < 0 if no column contains Null Value = 0 if one column contains Null value	none of the listed options

288	Given :	public class MoreEndings { public static void main(String[] args) throws Exception { Class driverClass = Class.forName("sun.jdbc.odbc.JdbcOdbcDrive r"); DriverManager.registerDriver((Driver) driverClass.newInstance()); // Some code } Inorder to compile & execute this code, what should we import?	r	java.sql.Drive r	java.sql.Drive rManager	java.sql.Data Source
289	Which of the following method can be used to execute to execute all type of queries i.e. either Selection or Updation SQL Queries?	executeAll() L()	executeAllSQ L()	execute()	executeQuer y()	executeQuer y()
290	Which method will return boolean when we try to execute SQL Query from a JDBC program?	executeUpda te()	executeSQL( )	execute()	executeQuer y()	executeQuer y()
291	Consider the following code & select the correct output. <pre>String sql ="select rollno, name from student"; PreparedStatement pst=cn.prepareStatement(sql); System.out.println(pst.toString()); ResultSet rs=pst.executeQuery(); while(rs.next()){     System.out.println(rs.getString(3)); }</pre>	will show only name	Compilation error	will show city	Compiles but error at run time	
292	It is possible to insert/update record in a table by using ResultSet. State TRUE/FALSE	true	false			
293	What is the default type of ResultSet in JDBC applications?	Read Only, Forward Only	Updatable, Forward only	Read only, Scroll Sensitive	Updatable, Scroll sensitive	
294	An application can connect to different Databases at the same time. State TRUE/FALSE.	true	false			
295	A) It is not possible to execute select query with execute() method B) CallableStatement can executes store procedures only but not functions	Both A and B is FALSE	Only A is TRUE	Only B is TRUE	Both A and B is TRUE	
296	A) When one use callablestatement, in that case only parameters are send over network not sql query. B) In preparestatement sql query will compile for first time only	Both A and B is FALSE	Both A and B is TRUE	Only A is TRUE	Only B is TRUE	

297 Consider the code below & select the correct output from the options:

```
String sql ="select * from ?";
String table=" txyz ";
PreparedStatement
pst=cn.prepareStatement(sql);
pst.setString(1,table );
ResultSet rs=pst.executeQuery();
while(rs.next()){
System.out.println(rs.getString(1)); }
```

will show all row of first column

Compilation error

Compiles but error at run time

Compiles but run without output

298

Sylvy wants to develop Student management system, which requires frequent insert operation about student details. In order to insert student record which statement interface will give good performance

Statement

CallableStatement

PreparedStatement

RowSet

299 class CreateFile{

```
public static void main(String[] args) {
try {
File directory = new File("c"); //Line 13
File file = new File(directory,"myFile");
if(!file.exists()) {
file.createNewFile(); //Line 16
}
catch(IOException e) {
e.printStackTrace }
}}
```

If the current direcory does not consists of directory "c", Which statements are true ?  
(Choose TWO)

Line 16 is never executed

An exception is thrown at runtime

Line 13

Line 13 creates a directory named "c" in the file system.

300

1) Driver 2) Connection

3) ResultSet 4) Statement

ResultSetMet aData 5) Statement 6)

DriverManag er 7) PreparedStat

Connection 8) Callablestate

ResultSetMet aData 5) Statement 6)

DriverManag er 7) PreparedStat

Connection 8) Callablestate

ResultSetMet aData 5) Statement 6)

DriverManag er 7) PreparedStat

Connection 8) Callablestate

ResultSetMet aData 5) Statement 6)

DriverManag er 5) Class

DataBaseMet aData

DataBaseMet aData

Which of the following options contains only JDBC interfaces?

All of the given options

301 Consider the code below & select the correct output from the options:

```
public class Test {  
    public static void main(String [] args) {  
        int x = 5;  
        boolean b1 = true;  
        boolean b2 = false;  
        if ((x == 4) && !b2 )  
            System.out.print("1 ");  
        System.out.print("2 ");  
        if ((b2 = true) && b1 )  
            System.out.print("3 ");  
    }  
}
```

302 Which three are legal array declarations?  
(Choose THREE)

303 Consider the given code and select the correct output:

```
class Test{  
    public static void main(String[] args){  
        int num1 = 012;  
        int num2 = 0x110;  
        int sum =num1+=num2;  
        System.out.println("Ans = "+sum);  
    }  
}
```

304 Say that class Rodent has a child class Rat and another child class Mouse. Class Mouse has a child class PocketMouse. Examine the following

```
Rodent rod;  
Rat rat = new Rat();  
Mouse mos = new Mouse();  
PocketMouse pkt = new PocketMouse();
```

Which one of the following will cause a compiler error?

305 Consider the code below & select the correct output from the options:

```
class Test{  
    public static void main(String[] args) {  
        parse("Four");  
        static void parse(String s){  
            try {  
                double d=Double.parseDouble(s);  
            }catch(NumberFormatException nfe){  
                d=0.0; }finally{  
                System.out.println(d); } }  
    }  
}
```

2 3  
int []  
myScores [];

1 3  
char []  
myChars;

2  
int [6]  
myScores;

3  
Dog myDogs  
[];

26

Compiles but  
error at run  
time

282

Compilation  
error

rod = mos

pkt = rat

pkt = null

rod = rat

Compilation  
0 error

A  
NumberForm  
ParseExcepti  
on is thrown  
by the parse  
method at  
runtime

A  
atException  
is thrown by  
the parse  
method at  
runtime

306 Consider the code below & select the correct output from the options:

```
class A{
    public int a=7;
    public void add(){
        this.a+=2; System.out.print("a"); }}
```

```
public class Test extends A{
    public int a=2;
    public void add(){
        this.a+=2; System.out.print("t"); }
    public static void main(String[] args) {
        A a =new Test();
        a.add();
        System.out.print(a.a); }}
```

307 What will be the output of the program?

```
public class CommandArgsTwo
{
    public static void main(String [] argh)
    {
        int x;
        x = argh.length;
        for (int y = 1; y <= x; y++)
        {
            System.out.print(" " + argh[y]);
        }
    }
}
```

and the command-line invocation is

> java CommandArgsTwo 1 2 3

t 7

t 9

a 9

Compilation error

308

What will be the result of the following program?

```
public class Init {
    String title;
    boolean published;
    static int total;
    static double maxPrice;
    public static void main(String[] args) {
        Init initMe = new Init();
        double price;
        if (true)
            price = 100.00;
        System.out.println("|" + initMe.title + "|" +
                           initMe.published + "|" +
                           Init.total + "|" + Init.maxPrice + "|" +
                           price+ "|");
    }
}
```

0 1 2

2 3

0 0 0

An exception is thrown at runtime

The program will compile, and print |null|false|0|0. |null|true|0|0. 0|0.0|, when run

The program will compile, and print |null|true|0|0. |null|true|0|0. 0|100.0|, when run

The program will compile, and print |null|true|0|0. |false|0|0.0|0. 0|100.0|, when run

The program will compile, and print |null|false|0|0. 0|100.0|, when run

309 Here is the general syntax for method definition:

```
accessModifier returnType methodName(  
parameterList )  
{  
    Java statements  
  
    return returnValue;  
}
```

What is true for the returnType and the returnValue?

310 Consider the following code and choose the correct option:

```
class Test{  
    class A{ static int x=3; }  
    static void display(){  
        System.out.println(A.x); }  
    public static void main(String[] args) {  
        display(); }}
```

311 Which of the following lines of code will compile without warning or error?

- 1) float f=1.3;
- 2) char c="a";
- 3) byte b=257;
- 4) boolean b=null;
- 5) int i=10;

312 Consider the following code and choose the correct option:

```
class Test{  
    interface Y{  
        void display(); }  
    public static void main(String[] args) {  
        new Y(){  
            public void display(){  
                System.out.println("Hello World"); }  
            }.display(); }}
```

313 Consider the following code and choose the correct option:

```
class Test{  
    static class A{  
        interface X{  
            int z=4; } }  
    static void display(){  
        System.out.println(A.X.z); }  
    public static void main(String[] args) {  
        display(); }}
```

The  
returnValue  
can be any  
type, but will  
be

The  
returnValue  
must be  
exactly the  
same type as  
the  
returnType

automatically  
converted to  
returnType  
when the  
method  
returns to the  
caller.

If the  
returnType is  
void then the  
returnValue  
can be any  
type

The  
returnValue  
must be the  
same type as  
the  
returnType,  
or be of a  
type that can  
be converted  
to returnType  
without loss  
of  
information.

Compilation  
3 error

Compiles but  
error at run  
time

0

Line 3

Line 1, Line  
3, Line 5

Line 1, Line 5

Line 5

Hello World

Compilation  
error

Compiles but  
error at run  
time

Compiles but  
run without  
output

Compilation  
4 error

Compiles but  
error at run  
time

314

What is the output of the following program?

```
public class MyClass
{
    public static void main( String[] args )
    {
        private static final int value = 9;
        float total;
        total = value + value / 2;
        System.out.println( total );
    }
}
```

315 Which of the given options is similar to the following code:

```
value += sum++ ;
```

316 What will happen if you attempt to compile and run the following code?

```
Integer ten=new Integer(10);
Long nine=new Long (9);
System.out.println(ten + nine);
int i=1;
System.out.println(i + ten);
```

317 Identify the statements that are correct:

- (A) int a = 13, a>>2 = 3
- (B) int b = -8, b>>1 = -4
- (C) int a = 13, a>>>2 = 3
- (D) int b = -8, b>>>1 = -4

318 Consider the following code:

```
int x, y, z;
y = 1;
z = 5;
x = 0 - (++y) + z++;
After execution of this, what will be the values
```

of x, y and z?

value = value + sum; sum = sum + 1;	sum = sum + 1; value = value + sum;	value = value + sum; + ++sum;
---	---	-------------------------------------

0

13.5

13 Compilation  
Error

19 followed by 11	19 followed by 20	Compile time error	10 followed by 1
----------------------	----------------------	-----------------------	---------------------

(A), (B) & (C)	(D)	(C) & (D)	(A) & (B)
-------------------	-----	-----------	-----------

319 Here is the general syntax for method definition:

```
accessModifier returnType methodName(  
parameterList )  
{  
    Java statements  
  
    return returnValue;  
}
```

What is true for the accessModifier?

x = -7, y = 1, z = 5	x = 3, y = 2, z = 6	x = 4, y = 1, z = 5	x = 4, y = 2, z = 6
-------------------------	------------------------	------------------------	------------------------

It can be  
omitted, but if  
not omitted

there are

several

choices,

including

private and

public

The access  
modifier must  
agree with

the type of

the return

value

It can be  
omitted, but if  
not omitted it

must be

private or

public

320 What will be the output of the program?

```
public class CommandArgs
{
    public static void main(String [] args)
    {
        String s1 = args[1];
        String s2 = args[2];
        String s3 = args[3];
        String s4 = args[4];
        System.out.print(" args[2] = " + s2);
    }
}
```

and the command-line invocation is

> java CommandArgs 1 2 3 4

args[2] = 2      args[2] = 3      args[2] = null      An exception  
is thrown at  
runtime

321 Consider the following code snippet:

int i = 10;

int n = ++i%5;

What are the values of i and n after the code  
is executed?

10, 1      11, 1      10,0      11,0  
int [] myList = {"1", "2", "3"};      int [] myList = {5, 8, 2};      int myList [] = {4, 9, 7, 0};      int myList [] = {4, 3, 7};

322

Which will legally declare, construct, and  
initialize an array?

323 Consider the code below & select the correct  
output from the options:

```
public class Test {
    public static void main(String[] args) {
        int x=5;
        Test t=new Test();
        t.disp(x);
        System.out.println("main X="+x);
    }
    void disp(int x) {
        System.out.println("disp X = "+x++);
    }
}
```

disp X = 6      disp X = 5      disp X = 5      Compilation  
main X=6      main X=5      main X=6      error

324 How many objects and reference variables are  
created by the following lines of code?

```
Employee emp1, emp2;
emp1 = new Employee();
Employee emp3 = new Employee();
```

Two objects  
and three  
reference  
variables.

Three objects  
and two  
reference  
variables.

Four objects  
and two  
reference  
variables.

Two objects  
and two  
reference  
variables.

325 A) The purpose of the method overriding is to  
perform different operation, though input  
remains the same.

B) one of the important Object Oriented  
principle is the code reusability that can be  
achieved using abstraction

Only A is  
TRUE      Only B is  
True      Both A and B  
is True      Both A and B  
is FALSE

326	class Test{ public static void main(String[] args){ byte b=(byte) (45 << 1); b+=4; System.out.println(b); }} What should be the output for the code written above?	48		Compiles but error at run time	Compilation error
327	What is the value of y when the code below is executed?  int a = 4; int b = (int)Math.ceil(a % 3 + a / 3.0);	1	2		3
328	Consider the following code and choose the correct option:  class Test{ class A{ interface X{ int z=4;  } static void display(){ System.out.println(new A().X.z); } } public static void main(String[] args) { display(); } }	0	Compilation error	Compiles but error at run time	4
329	Consider the code below & select the correct output from the options:  public class Test { public static void main(String[] args) { String[] elements = { "for", "tea", "too" }; String first = (elements.length > 0) ?elements[0] : null; System.out.println(first); } }		The variable first is set to null.	The variable first is set to elements[0].	Compiles but error at runtime
330	Given the following piece of code:  public class Test { public static void main(String args[]) { int i = 0, j = 5 ; for( ; (i < 3) && (j++ < 10) ; i++ ) { System.out.print(" " + i + " " + j ); } System.out.print(" " + i + " " + j ); } } what will be the output?	0 6 1 7 2 8 3 8	0 6 1 7 2 8 3 9	0 5 1 5 2 5 3 5	compilation fails
331	Given  class MybitShift { public static void main(String [] args) { int a = 0x5000000; System.out.print(a + " and "); a = a >>> 25; System.out.println(a); } }	83886080 and -2	2 and 83886080	2 and - 83886080	83886080 and 2

332 Consider the code below & select the correct output from the options:

```
public class Test {  
    int squares = 81;  
    public static void main(String[] args) {  
        new Test().go(); }  
    void go() {  
        incr(++squares);  
        System.out.println(squares); }  
    void incr(int squares) { squares += 10; } }
```

92

Compilation  
91 error

82

333 class C{  
 public static void main (String[] args) {  
 byte b1=33; //1  
 b1++; //2  
 byte b2=55; //3  
 b2=b1+1; //4  
 System.out.println(b1+" "+b2);  
 }  
}

Consider the code above & select the correct output.

334 What will be the output of the program ?

```
public class Test {  
    public static void main(String [] args)  
    {  
        signed int x = 10;  
        for (int y=0; y<5; y++, x--)  
            System.out.print(x + " , ");  
    }  
}
```

10, 9, 8, 7, 6, 9, 8, 7, 6, 5,

Compilation  
fails

An exception  
is thrown at  
runtime

335 1. public class LineUp {  
2. public static void main(String[] args) {  
3. double d = 12.345;  
4. // insert code here  
5. }  
6. }

Which code fragment, inserted at line 4,  
produces the output | 12.345|?

- A. System.out.printf("|%7f| \n", d);
- B. System.out.printf("|%3.7f| \n", d);
- C. System.out.printf("|%7.3d| \n", d);
- D. System.out.printf("|%7.3f| \n", d);

A

B

C

D

336	Consider the following code and choose the correct option: <pre>class Test{     interface Y{         void display();     }     public static void main(String[] args) {         Y y=new Y(){             public void display(){                 System.out.println("Hello World"); } };         y.display(); }}</pre>	Hello World	Compilation error	Compiles but error at run time	Compiles but run without output
337	<pre>class Test{     public static void main(String[] args){         int var;         var = var +1;         System.out.println("var =" +var);     } }</pre> <p>consider the code above &amp; select the proper output from the options.</p>	compiles and runs with no output	var = 1	does not compile	run time error
338	State the class relationship that is being implemented by the following code: <pre>class Employee {     private int empid;     private String ename;     public double getBonus()     {         Accounts acc = new Accounts();         return acc.calculateBonus();     } }</pre> <pre>class Accounts {     public double calculateBonus(){//method's code} }</pre>	Simple Aggregation	Association	Dependency	Composition
339	Given classes A, B, and C, where B extends A, and C extends B, and where all classes implement the instance method void doIt(). How can the doIt() method in A be called from an instance method in C?	It is not possible	super.doIt()	his.super.doIt ((A) () this).doIt();	
340	Which of the following will declare an array and initialize it with five numbers?	int [] a = {23,22,21,20, 19};	int a [] = new int[5];	int [5] array;	
341	Which of the following are correct variable names? (Choose TWO)	int #ss;	int 1ah;	int _;	int \$abc;

342 What is the output of the following:

```
int a = 0;  
int b = 10;
```

```
a = --b ;  
System.out.println("a: " + a + " b: " + b );
```

a: 9 b:11      a: 10 b: 9      a: 9 b:9      a: 0 b:9

343 As per the following code fragment, what is the value of a?

```
String s;  
int a;  
s = "Foolish boy.";  
a = s.indexOf("fool");
```

-1      0      4 random value

344 Consider the following code snippet:

```
int i = 10;  
int n = i++%5;
```

What are the values of i and n after the code is executed?

10, 1      11, 1      10,0      11,0

345 Consider the following code and choose the correct output:

```
int value = 0;  
int count = 1;  
value = count++ ;  
System.out.println("value: "+ value + " count: " + count);
```

value: 0      value: 0  
count: 0      count: 1  
value: 1      value: 1  
count: 1      count: 2

346 Consider the following code and select the correct output:

```
class Test{  
interface Y{  
void display(); }  
public static void main(String[] args) {  
new Y(){  
public void display(){  
System.out.println("Hello World"); } };  
}}
```

Hello World      Compilation error      Compiles but error at run time      Compiles but run without output

347

What is the output of the following program?

```
public class demo {  
public static void main(String[] args) {  
int arr[5];  
for (int i = 0; i < arr.length; i++) {  
arr[i] = arr[i] + 10;  
}  
for (int j = 0; j < arr.length; j++)  
System.out.println(arr[j]);  
}
```

A sequence of five 10's are printed      A sequence of Garbage Values are printed      compile time Error      Compiles but no output

348 Which of the following methods registers a thread in a thread scheduler?

run();      construct();      start();      register();

<pre>349 class PingPong2 {     synchronized void hit(long n) {         for(int i = 1; i &lt; 3; i++)             System.out.print(n + "-" + i + " ");     } }  public class Tester implements Runnable {     static PingPong2 pp2 = new PingPong2();     public static void main(String[] args) {         new Thread(new Tester()).start();         new Thread(new Tester()).start();     }     public void run() {         pp2.hit(Thread.currentThread().getId());     } } Which statement is true?</pre>	The output could be 5-1 6-1 6-2 5-2	The output could be 6-1 6-2 5-1 5-2	The output could be 6-1 5-2 6-2 5-1	The output could be 6-1 6-2 5-1 7-1
<p>350 Consider the following code and choose the correct option:</p> <pre>class Cthread extends Thread{     public void run(){         System.out.print("Hi");     }     public static void main (String args[]){         Cthread th1=new Cthread();         th1.run();         th1.start();         th1.run();     } }</pre>	will print Hi twice and throws Exception at run time	will print Hi Thrice	Compilation error	will print Hi once
<p>351 class Cthread extends Thread{</p> <pre>public void run(){     System.out.print("Hi"); } public static void main (String args[]){     Cthread th1=new Cthread();     th1.run();     th1.start();     th1.start(); }</pre>	will start two thread	will print Hi Once	will not print	will print Hi twice and throws exception at runtime
<p>352 Consider the following code and choose the correct option:</p> <pre>class Cthread extends Thread{     Cthread(){start();}     public void run(){         System.out.print("Hi");     }     public static void main (String args[]){         Cthread th1=new Cthread();         Cthread th2=new Cthread();     } }</pre>	will create two child threads and display Hi twice	compilation error	will not create any child thread	will display Hi once
<p>353 Which of the following methods are defined in class Thread? (Choose TWO)</p>	start()	wait()	notify()	run()

354 The following block of code creates a Thread using a Runnable target:

```
Runnable target = new MyRunnable();
Thread myThread = new Thread(target);
```

Which of the following classes can be used to create the target, so that the preceding code compiles correctly?

355

Which of the following statements can be used to create a new Thread? (Choose TWO)

356 What will be the output of the program?

```
class MyThread extends Thread
{
    MyThread() {}
    MyThread(Runnable r) {super(r);}
    public void run()
    {
        System.out.print("Inside Thread ");
    }
}
class MyRunnable implements Runnable
{
    public void run()
    {
        System.out.print(" Inside Runnable");
    }
}
class Test
{
    public static void main(String[] args)
    {
        new MyThread().start();
        new MyThread(new
MyRunnable()).start();
    }
}
```

- 357 A) Multiple processes share same memory location  
B) Switching from one thread to another is easier than switching from one process to another  
C) Thread makes it possible to maximize resource utilization  
D) Process is a light weight program

public class MyRunnable implements Runnable{public void run(){}}

Extend java.lang.Thread and override the run() method.

public class MyRunnable extends Runnable{public void run(){}}

Extend java.lang.Runnable and override the start() method.

public class MyRunnable implements Runnable{void run();}

Implement java.lang.Thread and implement the run() method.

public class MyRunnable extends Object{public void run();}

Implement java.lang.Runnable and implement the run() method

Prints "Inside Thread Inside Thread"

Does not compile

Prints "Inside Thread Inside Runnable"

Throws exception at runtime

All are FALSE

Only B and C is TRUE

Only A and B is TRUE

Only C and D is TRUE

358 A) Exception is the superclass of all errors  
and exceptions in the java language

B) RuntimeException and its subclasses are  
unchecked exception.

359 What will be the output of the program?

```
class MyThread extends Thread
{
    public static void main(String [] args)
    {
        MyThread t = new MyThread();
        t.start();
        System.out.print("one. ");
        t.start();
        System.out.print("two. ");
    }
    public void run()
    {
        System.out.print("Thread ");
    }
}
```

360 Consider the following code and choose the  
correct option:

```
class A implements Runnable{ int k;
public void run(){}
k++;
}
public static void main(String args[]){
A a1=new A();
a1.run();}
```

361 Given:

```
public class Threads4 {
public static void main (String[] args) {
new Threads4().go();
}
public void go() {
Runnable r = new Runnable() {
public void run() {
System.out.print("run");
}
};
Thread t = new Thread(r);
t.start();
t.start();
}
}
What is the result?
```

Only A is  
TRUE

Only B is  
TRUE

Both A and B  
are TRUE

Both A and B  
are FALSE

Compilation  
fails

An exception  
occurs at  
runtime.

It prints  
"Thread one.  
Thread two."

The output  
cannot be  
determined.

It will start a  
new thread

compilation  
error

Compiles but  
throws run  
time  
Exception

a1 is not a  
Thread

Compilation  
fails.

An exception  
is thrown at  
runtime.

The code  
executes  
normally and  
prints "run".

The code  
executes  
normally, but  
nothing is  
printed.

<pre>362 class Thread2 {     public static void main(String[] args) {         new Thread2().go(); }     public void go(){         Runnable rn=new Runnable(){             public void run(){                 System.out.println("Good Day.."); } };         Thread t=new Thread(rn);         t.start();     }}</pre> <p>what should be the correct output for the code written above?</p>	<p>Compilation fails.</p>	<p>An exception is thrown at runtime.</p>	<p>The code executes normally and prints "Good Day.."</p>	<p>prints Good Day.. Twice</p>
<pre>363 public class MyRunnable implements Runnable {     {         public void run()         {             // some code here         }     } }</pre> <p>which of these will create and start this thread?</p>	<p>new Runnable(MyRunnable).start();</p>	<p>new Thread(MyRunnable).run();</p>	<p>new MyRunnable().start();</p>	<p>new Thread(new MyRunnable()).start();</p>
<p>364 Consider the following code and choose the correct option:</p> <pre>class Nthread extends Thread{     public void run(){         System.out.print("Hi");}     public static void main(String args[]){         Nthread th1=new Nthread();         Nthread th2=new Nthread();     }}</pre>	<p>Will create two child threads and display Hi twice</p>	<p>compilation error</p>	<p>will not create any child thread</p>	<p>will display Hi once</p>
<p>365 Assume the following method is properly synchronized and called from a thread A on an object B:</p>	<p>After the lock on B is released, or after two seconds.</p>	<p>Two seconds after thread A is notified.</p>	<p>Two seconds after lock B is released.</p>	
<p>wait(2000);</p> <p>After calling this method, when will the thread A become a candidate to get another turn at the CPU?</p>	<p>After thread A is notified, or after two seconds.</p>	<p>Interrupt class</p>	<p>none of the listed options</p>	
<p>366 wait(), notify() and notifyAll() methods belong to _____</p>	<p>Object class</p>	<p>Thread class</p>	<p>Compilation error</p>	
<p>367 Consider the following code and choose the correct option:</p>	<pre>class Test {     public static void main(String[] args) {         new Test().display("hi", 1);         new Test().display("hi", "world", 2);         public void display(String... s, int x) {             System.out.print(s[s.length-x] + " "); } }</pre>	<p>hi hi</p>	<p>hi world</p>	<p>world</p>

368 Consider the following code and choose the correct option:

```
public class Test {  
    public static void main(String[] args) {  
        String name="Anthony Gomes";  
        int a=111;  
        System.out.println(name.indexOf(a)); }}
```

4

2

Compilation  
6 error

369 Given:

```
String test = "This is a test";  
String[] tokens = test.split("\\s");  
System.out.println(tokens.length);
```

What is the result?

0

1

Compilation  
4 fails.

370

Consider the following code and choose the correct option:

```
public class Test {  
    public static void main(String[] args) {  
        String data="78";  
        System.out.println(data.append("abc")); }}
```

78abc

abc78

Compilation  
error

Compiles but  
exception at  
run time

371 Consider the following code and choose the correct option:

```
public class Test {  
    public static void main(String[] args) {  
        String name="ALDPR7882E";  
        System.out.println(name.endsWith("E") &  
            name.matches("[A-Z]{5}[0-9]{4}[A-Z]"));}}
```

false

true

0

1

372 Examine this code:

```
String stringA = "Hello ";  
String stringB = " World";  
String stringC = " Java";  
String result;  
Which of the following puts a reference to  
"Hello World Java" in result?
```

result =  
stringA.concat(  
 stringB.concat(  
 stringC ));

result.concat(  
 stringA,  
 stringB,  
 stringC );

result+stringA  
+stringB+stringC;  
A).concat(St  
ringB).concat(  
StringC)

373 For two string objects obj1 and obj2:

- A) Use of obj1 == obj2 tests whether two String object references refer to the same object
- B) obj1.equals(obj2) compares the sequence of characters in obj1 and obj2.

Only A is  
TRUE

Only B is  
TRUE

Both A and B  
is TRUE

Both A and B  
is FALSE

374 What is the result of the following:

```
String ring = "One ring to rule them all,\n";  
String find = "One ring to find them."  
  
if ( ring.startsWith("One") &&  
find.startsWith("One") )  
    System.out.println( ring+find );  
else  
    System.out.println( "Different Starts" );
```

One ring to  
rule them all,  
One ring to  
find them.

One ring to  
rule them all,  
One ring to  
find them.

One ring to  
rule them  
all,\n One  
ring to find  
them.

Different  
Starts

375

Consider the following code and choose the correct option:

```
class MyClass {  
    String str1="str1";  
    String str2 ="str2";  
    String str3="str3";  
    str1.concat(str2);  
    System.out.println(str3.concat(str1));  
}
```

The code will fail to compile because the expression str3.concat(st r1) will not result in a valid argument for the println() method

The program will print

The program will print

The program will print

376 Given:

```
public class Theory {  
    public static void main(String[] args) {  
        String s1 = "abc";  
        String s2 = s1;  
        s1 += "d";  
        System.out.println(s1 + " " + s2 + " " +  
            (s1==s2));  
  
        StringBuffer sb1 = new StringBuffer("abc");  
        StringBuffer sb2 = sb1;  
        sb1.append("d");  
        System.out.println(sb1 + " " + sb2 + " " +  
            (sb1==sb2));  
    }  
}
```

Which are true? (Choose all that apply.)

The second line of output is abcd abcd  
true

The first line of output is abc abc  
false

The first line of output is abcd abc  
false

The second line of output is abcd abc  
false

377 class StringManipulation{  
 public static void main(String[] args){  
 String str = new String("Cognizant");  
 str.concat(" Technology");  
 StringBuffer sbf = new StringBuffer(" Solutions");  
 System.out.println(str+sbf);  
 }  
}

consider the code above & select the proper output from the options.

Cognizant  
Technology  
Solutions

Cognizant  
Technology

Cognizant  
Solutions

Technology  
Solutions

378 What does this code write:

```
 StringTokenizer stuff = new StringTokenizer(  
    "abc def+ghi", "+");  
 System.out.println( stuff.nextToken() );  
 System.out.println( stuff.nextToken() );
```

abc def

abc def ghi

abc def +

abc def +ghi

379 Consider the following code and choose the correct option:

```
public class Test {  
    public static void main(String[] args) {  
        StringBuffer sb = new  
        StringBuffer("antarctica");  
        sb.delete(0,6);  
        System.out.println(sb); }}
```

tica anta

Compilation error

Complies but exception at run time

380 Consider the following code and choose the correct option:

```
public class Test {  
    public static void main(String[] args) {  
        String name="vikaramaditya";  
        System.out.println(name.substring(2,  
            5).toUpperCase().charAt(2));}}
```

K A R I

381 Consider the following code and choose the correct option:

```
public class Test {  
    public static void main(String[] args) {  
        StringBuffer sb = new  
        StringBuffer("antarctica");  
        sb.reverse();  
        sb.replace(2, 7, "c");  
        sb.delete(0,2);  
        System.out.println(sb); }}
```

acctna iccratna ctna tna

382 Consider the following code and choose the correct option:

```
class Test {  
    public static void main(String args[]) {  
        String s1 = "abc";  
        String s2 = "def";  
        String s3 = s1.concat(s2.toUpperCase());  
        System.out.println(s1+s2+s3); }}
```

abcabcDEFD abcdefabcDE  
abcdefabcdef EF F

383 What will be the result when you attempt to compile and run the following code?.

```
public class Conv {  
    public static void main(String argv[]){  
        Conv c=new Conv();  
        String s=new String("ello");  
        c.amethod(s);  
  
        public void amethod(String s){  
            char c='H';  
            c+=s;  
            System.out.println(c);  
        }  
    }
```

Compilation and output the string "Hello"

Compilation and output the string "ello"

Compilation and output the string elloH

Compile time error

none of the listed options

384 Consider the following code and choose the correct option:

```
public class Test {  
    public static void main(String[] args) {  
        String name="Anthony Gomes";  
        System.out.println(name.replace('n',  
            name.charAt(3)).compareTo(name));  
    }  
}
```

-6

6

Compilation  
0 error

385 Consider the following code and choose the correct option:

```
class Test {  
    public static void main(String args[]) {  
        String name=new String("batman");  
        int ibegin=1;  
        char iend=3;  
        System.out.println(name.substring(ibegin,  
            iend));  
    }  
}
```

bat

at

atm

Compilation  
error

386 Consider the following code and choose the correct option:

```
public class Test {  
    public static void main(String[] args) {  
        StringBuffer sb=new  
        StringBuffer("YamunaRiver");  
        System.out.println(sb.capacity());  
    }  
}
```

10

27

24

11

387 Consider the following code and choose the correct option:

```
public class Test {  
    public static void main(String[] args) {  
        StringBuffer sb = new  
        StringBuffer("antarctica");  
        sb.reverse();  
        sb.insert(4, 'r');  
        sb.replace(2, 4, "c");  
        System.out.println(sb);  
    }  
}
```

acitcratna

acitrcratna

accircratna

accrcratna

388 A)A string buffer is a mutable sequence of characters.

B) sequece of characters in the string buffer can not be changed.

389 Examine this code:

```
String stringA = "Wild";  
String stringB = " Irish";  
String stringC = " Rose";  
String result;
```

Which of the following puts a reference to "Wild Irish Rose" in result?

Only A is  
TRUE

Only B is  
TRUE

Both A and B  
is TRUE

Both A and B  
is FALSE

```
result =  
stringA.concat(  
t(  
stringB.concat(  
t( stringC ) );
```

```
result.concat(  
stringA,  
stringB,  
stringC );
```

```
result+stringA  
+stringB+stri  
ngC;
```

```
concat(String  
A).concat(Sti  
ngB).concat(Sti  
ngC);
```

390 Consider the following code and choose the correct option:

```
class Test {  
    public static void main(String[] args) {  
        new Test().display(1,"hi");  
        new Test().display(2,"hi", "world" );  
        public void display(int x,String... s) {  
            System.out.print(s[x] + " ");  
        }  
    }  
}
```

hi hi	hi world	world	Compilation error
-------	----------	-------	-------------------

391 Consider the following code and choose the correct option:

```
public class Test {  
    public static void main(String[] args) {  
        String name="vikaramaditya";  
        System.out.println(name.codePointAt(2)+name  
        .charAt(3));  
    }  
}
```

203	204	205	Compilation error
-----	-----	-----	-------------------

392 Consider the following code and choose the correct option:

```
public class Test {  
    public static void main(String[] args) {  
        String data="7882";  
        data+=32; System.out.println(data);  
    }  
}
```

7914	Compiles but exception at run time	788232	Compilation error
------	------------------------------------	--------	-------------------

393 Which code can be inserted at Line X to print "Equal"?

```
public class EqTest{  
    public static void main(String argv[]){  
        EqTest e=new EqTest();  
    }  
}
```

```
EqTest(){  
    String s="Java";  
    String s2="java";  
    // Line X  
    {  
        System.out.println("Equal");  
    }else  
    {  
        System.out.println("Not equal");  
    }  
}  
if(s==s2) if(s.equals(s2)) if(s.equalsIgnoreCase(s2)) if(s.noCaseMatch(s2))
```

```

394 import java.io.*;
public class MyClass implements Serializable
{
private int a;
public int getA() { return a; }
public MyClass(int a){this.a=a; }
private void writeObject( ObjectOutputStream
s)
throws IOException {
// insert code here
}
}

```

Which code fragment, inserted at line 15, will allow Foo objects to be correctly serialized and deserialized?

395

Which of the following opens the file "myData.stuff" for output first deleting any file with that name?

396

```

import java.io.*;
public class MyClass implements Serializable
{

private Tree tree = new Tree();

public static void main(String [] args) {
MyClass mc= new MyClass();
try {
FileOutputStream fs = new
FileOutputStream("MyClass.ser");
ObjectOutputStream os = new
 ObjectOutputStream(fs);
os.writeObject(mc); os.close();
} catch (Exception ex) { ex.printStackTrace(); }
}
}

```

s.writeInt(x);	s.serialize(x);	s.defaultWriteObject();	s.writeObject(x);
FileOutputStream fos = new FileOutputStream("myData.stuff");	FileOutputStream fos = new FileOutputStream("myData.stuff", true);	DataOutputStream dos = new DataOutputStream(new BufferedOutputStream("myData.stuff"));	FileOutputStream fos = new FileOutputStream("myData.stuff", true);

Compilation fails	An exception is thrown at runtime	An instance of MyClass is serialized	A instance of MyClass and an instance of Tree are both serialized
-------------------	-----------------------------------	--------------------------------------	---

<p>397 Consider the following code and choose the correct option:</p> <pre>class std implements Serializable{     int call; std(int c){call=c;}     int getCall(){return call;} } public class Test{     public static void main(String[] args) throws IOException {     File file=new File("d:/std.txt");     FileOutputStream fos=new FileOutputStream(file);     ObjectOutputStream oos=new ObjectOutputStream(fos);     std s1=new std(10);     oos.writeObject(s1);     oos.close(); }}</pre>	<p>the state of the object s1 will be stored to file std.txt</p>	<p>Compilation error</p>	<p>Compiles but error at runtime</p>	<p>the state of the object s1 will not be stored to the file.</p>
<p>398 Consider the following code and choose the correct option:</p> <pre>public class Test {     public static void main(String[] args) throws IOException {     File file=new File("D:/jlist.lst");     byte buffer[]=new byte[(int)file.length()+1];     FileInputStream fis=new FileInputStream(file);     int ch=0;     while((ch=fis.read())!=-1){         System.out.print(ch); }}</pre>	<p>reads data from file one byte at a time and display it on the console.</p>	<p>Compilation error</p>	<p>reads data from file named jlist.lst in byte form and ascii value</p>	<p>Compiles but error at runtime</p>
<p>399 Consider the following code and choose the correct option:</p> <pre>public class Test {     public static void main(String[] args) throws IOException {     File file=new File("D:/jlist.lst");     byte buffer[]=new byte[(int)file.length()+1];     FileInputStream fis=new FileInputStream(file);     int ch=0;     while((ch=fis.read())!=-1){         System.out.print((char)ch); }}</pre>	<p>reads data from file one byte at a time and display it on the console.</p>	<p>Compilation error</p>	<p>reads data from file named jlist.lst in byte form and display garbage value</p>	<p>Compiles but error at runtime</p>
<p>400 Consider the following code and choose the correct option:</p> <pre>public class Test {     public static void main(String[] args) {     File file=new File("d:/prj/lib");     file.mkdirs();}}</pre>	<p>creates directory d:/prj/lib</p>	<p>Compilation error</p>	<p>Compiles but error at runtime</p>	<p>Compiles and executes but directory is not created</p>

401	Consider the following code and choose the correct option: <pre>public class Test {     public static void main(String[] args) throws IOException {         String data="Confidential info";         byte buffer[]={data.getBytes()};         FileOutputStream fos=new FileOutputStream("d:/temp");         for(byte d : buffer){             fos.write(d); } }}</pre>	writes data to file in byte form.	Compilation error	writes data to the file in character form.	Compiles but error at runtime
402	Given : <pre>import java.io.*; public class ReadingFor {     public static void main(String[] args) {         String s;         try {             FileReader fr = new FileReader("myfile.txt");             BufferedReader br = new BufferedReader(fr);             while((s = br.readLine()) != null)             System.out.println(s);             br.flush();         } catch (IOException e) {             System.out.println("io error");         }     } }</pre> <p>And given that myfile.txt contains the following two lines of data: ab cd</p> <p>What is the result?</p>	ab	Compilation Error	ab cd	a b c d
403	Consider the following code and choose the correct option: <pre>class std{     int call; std(int c){call=c;}     int getCall(){return call;} } public class Test{     public static void main(String[] args) throws IOException {         File file=new File("d:/std.txt");         FileOutputStream fos=new FileOutputStream(file);         ObjectOutputStream oos=new ObjectOutputStream(fos);         std s1=new std(10);         oos.writeObject(s1);         oos.close();     }}</pre>	the state of the object s1 will be stored to file std.txt	Compilation error	Compiles but error at runtime	the state of the object s1 will not be stored to the file.

404 Consider the following code and choose the correct option:

```
public class Test {  
    public static void main(String[] args) {  
        File file=new File("D:/jlist.lst");  
        byte buffer[]=new byte[(int)file.length()];  
        FileInputStream fis=new  
FileInputStream(file);  
        fis.read(buffer);  
        System.out.println(buffer);  
    }  
}
```

reads data from file named jlist.lst in byte form and display it on console.

Compilation error

reads data from file named jlist.lst in byte form and display garbage value

Compiles but error at runtime

405 Consider the following code and choose the correct option:

```
public class Test {  
    public static void main(String[] args) throws  
IOException {  
    File file=new File("D:/jlist.lst");  
    byte buffer[]=new byte[(int)file.length()];  
    FileInputStream fis=new  
FileInputStream(file);  
    fis.read(buffer);  
    System.out.println(new String(buffer)); } }
```

reads data from file named jlist.lst in byte form and display it on console.

Compilation error

reads data from file named jlist.lst in byte form and display garbage value

Compiles but error at runtime

406

What happens when the constructor for FileInputStream fails to open a file for reading?

throws a DataFormatException

throws a FileNotFoundException

throws an ArrayIndexOutOfBoundsException

returns null

407 Consider the following code and choose the correct option:

```
public class Test {  
    public static void main(String[] args) {  
        File file=new File("d:/prj,d:/lib");  
        file.mkdirs(); }}
```

creates directories names prj and lib in d: drive

Compilation error

Compiles but error at run time

Compiles and executes but directories are not created

408 Consider the following code and choose the correct output:

```
public class Person{  
    public void talk(){ System.out.print("I am a  
Person "); }  
  
public class Student extends Person {  
    public void talk(){ System.out.print("I am a  
Student "); }  
}
```

what is the result of this piece of code:

```
public class Test{  
    public static void main(String args[]){  
        Person p = new Student();  
        p.talk();  
    } }
```

I am a Person

I am a Student

I am a Person I am a Student

I am a Student I am a Person

409	Which of these are two legal ways of accessing a File named "file.tst" for reading. Select the correct option: A)FileReader fr = new FileReader("file.tst"); B)InputStream fr = new FileInputStream("file.tst"); C)InputStreamReader isr = new InputStreamReader(fr, "UTF8"); D)FileReader fr = new FileReader("file.tst", "UTF8");	A,D	B,C	C,D
410	What is the DataOutputStream method that writes double precision floating point values to a stream?	writeBytes()	writeFloat()	write()
411	Consider the following code and choose the correct option: <pre>public class Test{     public static void main(String[] args) {         File dir = new File("dir");         dir.mkdir();         File f1 = new File(dir, "f1.txt"); try {             f1.createNewFile(); } catch (IOException e) {         }         File newDir = new File("newDir");         dir.renameTo(newDir); }</pre>	The file system has a new empty directory named newDir;	The file system has a new empty directory named newDir;	The file system has a directory named dir, containing a file f1.txt
412	Consider the following code and choose the correct option: <pre>public class Test {     public static void main(String[] args) throws IOException {     File file=new File("d:/data");     byte buffer[]=new byte[(int)file.length()+1];     FileInputStream fis=new     FileInputStream(file);     fis.read(buffer);     FileWriter fw=new FileWriter("d:/temp.txt");     fw.write(new String(buffer));}}</pre>	Transfer content of file data to the temp.txt	Compilation error	Compiles but error at runtime Compiles and runs but content not transferred to the temp.txt

413

```

import java.io.EOFException;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
public class MoreEndings {
    public static void main(String[] args) {
        try {
            FileInputStream fis = new
                FileInputStream("seq.txt");
            InputStreamReader isr = new
                InputStreamReader(fis);
            int i = isr.read();
            while (i != -1) {
                System.out.print((char)i + "|");
                i = isr.read();
            }
        } catch (FileNotFoundException fnf) {
            System.out.println("File not found");
        } catch (EOFException eofe) {
            System.out.println("End of stream");
        } catch (IOException ioe) {
            System.out.println("Input error");
        }
    }
}

```

Assume that the file "seq.txt" exists in the current directory, has the required access permissions, and contains the string "Hello".

Which statement about the program is true?

414 Consider the following code and choose the correct option:

```

public class Test{
    public static void main(String[] args) throws
        IOException {
        File file = new File("d:/temp.txt");
        FileReader reader=new FileReader(file);
        reader.skip(7); int ch;
        while((ch=reader.read())!=-1){
            System.out.print((char)ch); } }
}

```

415

A file is readable but not writable on the file system of the host platform. What will be the result of calling the method canWrite() on a File object representing this file?

416

Which of following set of functions are example of method overloading

The program will not compile because a certain unchecked exception is not caught.

Skip the first seven characters and then starts reading file and display it on console

A SecurityException is thrown  
void add(int x,int y)  
char add(float x,int y)  
add(int x,int y)  
char add(float y)

The program will compile and print H|e|||o|Input error.

Compilation error

The program will compile and print H|e|||o|End of stream.

Compiles and runs without output

The program will compile, print H|e|||o|, and then terminate normally.

Compiles but error at runtime

The file is modified from being

unwritable to being

writable.

void add(int

x,int y)

void sum(double

x,double y)

417

What is the advantage of runtime polymorphism?

Efficient utilization of memory at runtime	Code reuse	Code flexibility at runtime	avoiding method name confusion at runtime
--	------------	-----------------------------	---

418

Which of the following is an example of IS A relationship?

Ford - Car	Microprocessor - Computer	Tea - Cup	Driver - Car
------------	---------------------------	-----------	--------------

419 Which of the following is not a valid relation between classes?

Inheritance	Segmentation	Instantiation	Composition
-------------	--------------	---------------	-------------

420 Which of the following is not an attribute of object?

State	Behaviour	Inheritance	Identity
-------	-----------	-------------	----------

QuestionText	Choice1	Choice2	Choice3	Choice4	Choice5	Grade1	Grade2	Grade3	Grade4	Grade5	AnswerDescript	QuestionMedia	AnswerMedia	Author	Reviewer	Is Numeric
Carefully read the question & answer accordingly: int void increment( int i ) { i++; }  int main() { for ( i = 0; i < 10; increment( i ) ) { } printf("i=%d\n", i); return 0; }  What is the output of given code snippet?	loop will go infinite state.	i=10	i=9	i=11		1	0	0	0	0	TEXT	TEXT				
Carefully read the question & answer accordingly: void main () { int i,j=5; i=5+j++; if(i) print ("%d", i); }  What is the output of given code snippet?		1	None of these		compile time error	0	0	1	0	0	TEXT	TEXT				
Carefully read the question and answer accordingly. int main() { int num,factorial; printf("Enter a number:\n"); scanf("%d", &num); factorial=1; while (num>0) { factorial=factorial*num; --num; } if(factorial==num) { getch(); return 0; } } identify a correct option which will give the factorial number for given input.	factorial=factorial+num; ++num;	factorial=fact orial*num; -- num;	factorial=fact orial*num; -- num;	factorial=fact orial*num; -- num;		0	1	0	0	0	TEXT	TEXT				
Carefully read the question & answer accordingly: void main() { int i = 0; for (i=0;i<20; i++) { switch(i) { case 0 : i += 4; case 1 : i += 1; case 5 : i += 7; default : i += 4; break; } printf("%d ", i); } }  What is the output of given code snippet?	4 1 4 4 4 7 and i=6 to 19 it will print 4	12 17 22	4 11 16 21	16 21		0	0	0	1	0	TEXT	TEXT				
Carefully read the question and answer accordingly: Select the correct option from the below code snippet. for(i=1;i<n;j++) { printf ("Enter num %d:"); scanf ("%d", &num); if(num<0) { continue; } if(num>0) { sum+=num; } } jump: if(i<0) { average=sum/(i-1); printf ("Average: %f",average); }	goto jump; sum+=sum+num;	Exit;	continue jump: sum+=sum+n m;	break: sum+=sum+n m;		1	0	0	0	0	TEXT	TEXT				
Carefully read the question & answer accordingly: int main() { int n; for (n = 9; n!=0; n--) printf("%d", n--); return 1; }  What is the output of given code snippet?	9 8 7 6 5 4 3 2 1	Error "negative value can not return to main()"	Infinite loop	9 7 5 3 1		0	0	1	0	0	TEXT	TEXT				
Carefully read the question & answer accordingly: int inc_value(int a) { a=20; return 5+ (++a); } void main() { int a=10;c; while (a>0) { c=inc_value(a); printf("%d",c); a-=10; } }  What is the output of given code snippet?	Infinite	25	16	26	None of these options	0	0	0	1	0	TEXT	TEXT				
Carefully read the question & answer accordingly: What is your observation on the below code? int main() { int i,num,product=1; for(i=1;i<4;i++) { printf("Enter num %d:",i); scanf ("%d", &i); if(i!=0) continue; product*=num; } printf("product=%d",product); return 0; }  What is the output of the below code?	Loop will not be executed more than 4 times	Next iteration will be considered when zero is given	Infinite Loop when 0 is given continuously	None of the listed options	Loop will be terminated when zero is given	0.5	0.5	0	0	0	TEXT	TEXT				
Carefully read the question and answer accordingly. What is the output of the below program?  void main() { int d,i,j; d=1; for(i=1;i<5 ;i++) for(j=1;j<5 ;j++) if((i+j)%3 == 0) d= d*1; else printf("%d", d); getch(); }  Carefully read the question & answer accordingly: void main()	10	13	5	20		0	0	1	0	0	TEXT	TEXT				
Carefully read the question & answer accordingly: void main() { int i = 0; for(i=0;i<10;) switch (i) { case '0': printf("Case1"); break; case '1': printf("Case2"); break; default: printf("Default Case"); } }  What is the output of given code snippet?	case2	case1	Default Case	loop will go infinite state.	None of these option.	0	0	1	0	0	TEXT	TEXT				
Carefully read the question & answer accordingly: void main() { int counter = 5,number = 10; do { number/=counter; } while(counter--); printf ("%d", number); }  What is the output of the given code snippet?	Divide error	Garbage value	O	2		1	0	0	0	0	TEXT	TEXT				

	Carefully read the question & answer accordingly: void main() { int list[6]; for(j = 0; j < 6; j++) { list[j] = 3 * j + 4; if(j % 3 == 0) { list[j] = list[j] - 2; } } for(i=0;j<i;j++) printf("%d",list[i]); } What is the output of given code snippet?	4 7 10 11 16 19	19 15 18 19 24 27	None of these options.	2 7 10 11 16 19	0 0 0 1	TEXT	TEXT
	Carefully read the question & answer accordingly: void main() { int choice = 20, arr[] = {10, 20, 30}; switch (choice) { case arr[0]: printf("Morning"); case arr[1]: printf("Evening"); case arr[2]: printf("Night"); default:printf("After Noon"); } }	Night Morning	After Noon	Evening	Compile-time error	0 0 0 1	TEXT	TEXT
	Carefully read the question & answer accordingly: void main() { int i,j; clrscr(); for(i=0;i<12;i++) { for(j=1;j<=2;j++) { if(i==j) { cout<<"*"; printf(" %d",i); } } } }	1 2 2 1	1 1 1 1	2 1 2 2	1 2 1 2	1 0 0 0	TEXT	TEXT
	Carefully read the question & answer accordingly: void main() { int i,j; clrscr(); for(i=0;i<12;i++) { for(j=1;j<=2;j++) { if(i==j) { cout<<"*"; printf(" %d",i); } } } }	10 11 11 10	10 11 19 12 11	10 11 11 12	10 11 11 12	1 0 0 0	TEXT	TEXT
	Carefully read the question & answer accordingly: What is the output of the below code void main() { int i = 1, j = 1; for ( ; ; ) { if (i > 5) break; else j++; printf ("%d %d", i, j); i++; } }	Run time error	2 5	2 4	Compile Time Error	0 1 0 0	TEXT	TEXT
	Carefully read the question & answer accordingly: void main() { int i = 3; switch() { case 1: print("All The Best"); case 2: print("Case1"); break; case 2: print("Case2"); break; default: print("default Case"); } }	None of these option	All The Best	Compile-time error		0 1 0 0	TEXT	TEXT
	Carefully read the question & answer accordingly: What is your observation on the below code snippet? switch(operator) { case '+': printf("num1+num2=%2f",num1+num2); getch(); break; case '-': printf("num1-num2=%2f",num1-num2); getch(); break; default: printf("Operator is not correct"); getch(); break; }	Default block will also be executed when the operator '+' is given	Run time error - Break is missing in case statement.	Compile-time error while executing with '-' Operator	Default block will also be executed when the operator '-' is given	0 0 0 0 1	TEXT	TEXT
	Carefully read the question & answer accordingly: What is the output of below code? int main() { int a = 60; print("\n%d",a << 1); }	240	120	480	60	0 1 0 0	TEXT	TEXT
	Carefully read the question & answer accordingly: void main() { FILE *fp; char str[20]; fp=fopen("MSG.txt","a"); fputs("You're alive.Do something!",fp); fclose(fp); fp=fopen("MSG.txt","r"); fgets(str,10,fp); printf("%s",str); }	You're all	None of these options. file_pointer.close() method is used to closing the file.	ve. Do something!	You're all	0 0 0 1	TEXT	TEXT
	Carefully read the question & answer accordingly: Choose the correct option? w+b and wb+ are same	w+b and wb+ are same	reets(); returns null on error	None of the listed options are true		1 0 0 0	TEXT	TEXT
	Carefully read the question & answer accordingly: void main() { FILE *fp; char str[20]; fp=fopen("abc.txt","w"); fputs("I am a boy\n",fp); fclose(fp); fp=fopen("abc.txt","r"); gets(str,10,fp); printf("%s",str); }	I am a boy and cursor will go to new line.	I am a boy\n I am a boy	I am a bo		0 0 0 1	TEXT	TEXT

Carefully read the question & answer accordingly: Which of the following operations can be performed on the file "MyFile.txt" using the below code?					Read and Write							
FILE *fp; fp = fopen("MyFile.txt", "+r");	Archive	Read	Append	Write	Read and Write	0	0	0	0	1	TEXT	TEXT
Carefully read the question & answer accordingly: Which of the following function can be used to terminate the main function from another function safely?	All Of the Options	return(expr);	abort();	exit(expr);		0	0	0	1		TEXT	TEXT
Carefully read the question & answer accordingly: What is the output of below code:												
void main() { FILE *fp; fopen("XYZ.doc", "w"); fputs("Hello", fp); fclose(fp); }	None of these.	Compile-time error	XY file will be black	XY2 file will contain "Hello"		0	0	0	1		TEXT	TEXT
Carefully read the question & answer accordingly: The Content of a file will be lost if it is opened in :	w+ mode	w mode	a mode	c mode		1	0	0	0		TEXT	TEXT
Carefully read the question & answer accordingly: void main() { FILE *fp; char str[2]; fp=fopen("Believe.txt", "r"); fgets(str, 4, fp); fp=fopen("Believe.txt", "r"); fgets(str, 7, fp); printf("%s", str); }	char str[2]; size need to be increased	fp=fopen ("Believe.txt", "r"). need to be removed.	fp=fopen ("Believe.txt", "r"). need to be used.	fp=fopen ("Believe.txt", "r"). need to be replaced with fgets(str, fp);		0	0	1	0		TEXT	TEXT
Carefully read the question & answer accordingly: void main() { char str[2]; FILE *fp; fp=fopen("life.txt", "r"); fgets(str, 4, fp); printf("%s", str); } /Life.txt contains: Life is beautiful. What is the output of above code snippet?	Lif	L	Li	Life		1	0	0	0		TEXT	TEXT
Carefully read the question & answer accordingly: The Function sprint works like printf but operates on the string	string	pointer	none of the listed options	data in a file		1	0	0	0		TEXT	TEXT
Carefully read the question and answer accordingly: Which of the following data structure is linear data structure?	Graphs	Tree	Arrays	None of the listed options		0	0	1	0		TEXT	TEXT
Carefully read the question and answer accordingly: The way a card game player arranges his cards as he picks them up one by one is an example of :	Bubble Sort	Merge sort	selection Sort	insertion sort		0	0	0	1		TEXT	TEXT
Carefully read the question and answer accordingly: Which of the following options are correct?		A tree which does not have any node other than root node has depth of zero.	A tree which does not have any node other than root node has exactly one branch or degree.	None of the options.		0.5	0.5	0	0		TEXT	TEXT
Carefully read the question and answer accordingly: Where do we use the Arrow in the flowchart?		An arrow coming from one symbol and ending at another symbol represents that control passes to the symbol the arrow points to.	Represents Input / Output symbol	Represents Predefined Process symbol	Represents Stored data symbol	1	0	0	0		TEXT	TEXT
Carefully read the question and answer accordingly: One of the best application of Stack is	Breadth First Search	Recursion	Array	Radix sort		0	1	0	0		TEXT	TEXT
Carefully read the question and answer accordingly: The if ... else keyword in a pseudocode is similar to the symbol in a flowchart.	Subroutine	Annotation	Decision	Process	Procedure	0	0	1	0	0	TEXT	TEXT
Carefully read the question and answer accordingly: Which of the options is false :		For a serial search to work the data in the array must not be alphabetic or numeric order.	A serial search begins with the first Array element.	A serial search continues searching element by element either until a match is found or until the end of array is found.	A serial search is useful when the amount of data that needs to be searched is large.	0	0	0	1		TEXT	TEXT
Carefully read the question and answer accordingly: Which of the following name does not relate to stacks?	Sorting	LIFO list	Push-down lists	None of the options	FIFO lists	0	0	0	1		TEXT	TEXT
Carefully read the question and answer accordingly: The operation of processing each element in the list is known as		Inserting	Merging	Traversal		0	0	0	1		TEXT	TEXT
Carefully read the question & answer accordingly: Algorithm and Flow chart help us to	Know the memory capacity	Identify the base of number system	Direct the output to a printer	Specify the problem completely and clearly		0	0	0	1		TEXT	TEXT
Carefully read the question & answer accordingly: int main() { int a=0,b=0; display(a,b); getch(); return 0; } int display(int a, int b) { a=(b75)*9; printf("%d%d",a,b); return 0; }	Run Time Error	75.84	84.75	84.84		0	0	1	0		TEXT	TEXT
Carefully read the question & answer accordingly: Find the output of the below program:						0	0	0	1		TEXT	TEXT
void fun(); int i = 10; void main() { int i = 20; printf("%d",i); i = 30; fun(); } void fun() { printf("%d",i); }	20 30	10 30	20 20	20 10		0	0	0	1		TEXT	TEXT
Carefully read the question & answer accordingly: What will be output if you compile and execute the following c code? #define _2 void main() { int o; o=_2; printf("%d",o); }	Compile error	133	29	343		0	0	1	0		TEXT	TEXT
Carefully read the question & answer accordingly: The keyword used to transfer control from a function back to the calling function is	goto	switch	return	goback		0	0	1	0		TEXT	TEXT

Carefully read the question & answer accordingly: void can be used	All of the listed options.	As a datatype of a function that does not returns any value to its calling environment.	as the name of a variable	In an expression		0	1	0	0	TEXT	TEXT		
Carefully read the question & answer accordingly: What will be the output if you compile and execute the following c code? #include <stdio.h> void main() { int i; float f; printf("%d", i); }	106	12	18	10		1	0	0	0	TEXT	TEXT		
Carefully read the question & answer accordingly: What are your observations in the below code? float average(float a[],n) { int i; float avg, sum=0.0; for(i=1; i<n;i++) { printf("%f", a[i]); sum+=a[i]; } avg =(sum/n); printf("average = %f",avg); return avg; }	This code will throw an Error	This code will be executed properly	Datatype of array a[] should not be given	Data Type of n is missing		0.5	0	0	0.5	TEXT	TEXT		
Carefully read the question & answer accordingly: int printvalues(float); void main() { float f=2.5; float value; value = printvalues(); printf("%f",value); } int printvalues(float f) { switch(f) { case 2.3 : printf("Value is 2.3"); case 2.5 : printf("Value is 2.5"); default : printf("invalid case"); } } What will be the output of above program?	Value is 2.5	None of the listed options	Value is 2.5, invalid case	This will produce error		0	0	0	1	TEXT	TEXT		
Carefully read the question & answer accordingly: Features for accessing a variable through its address is desirable because	A function can return more than one value.	Call by reference can be simulated.	All of the options.	Excessive use of global variable can be avoided		0	0	1	0	TEXT	TEXT		
Carefully read the question & answer accordingly: Any program must contain atleast :	One section to get input data	One display block.	One function	One Looping constructs		0	0	1	0	TEXT	TEXT		
Carefully read the question & answer accordingly: Which is true about array ?	None of the given options	Does not store in contiguous locations	Automatic array bounds checking	No need to worry about the allocation and de- allocation of arrays		1	0	0	0	TEXT	TEXT		
Carefully read the question & answer accordingly: Which of the following function is more appropriate for reading in a multi-word string?	printf(); puts();	scanf();	gets();			0	0	0	1	TEXT	TEXT		
Carefully read the question and answer accordingly. What is the output in the below code snippet? void main() { int twod[3][3]; int i, j; clrscr(); for(i=0; i<3; i++) for(j=0; j<3; j++) twod[i][j] = i; for(i=0; i<3; i++) for(j=0; j<3; j++) printf("%d", twod[i] [j]); printf("\n"); }	0 0 0 1 2 4	1 1 0 1 1 0 2 4	0 0 0 1 2 0 2 4	1 1 0 1 2 0 2 4		0	0	1	0	TEXT	TEXT		
Carefully read the question & answer accordingly: What will happen if in a program, you assign a value to an array element whose subscript exceeds the size of array?	The program may crash if some important data gets overwritten.	The compiler would report an error.	The element will be set to 0.	The array size would appropriately grow.		1	0	0	0	TEXT	TEXT		
Carefully read the question & answer accordingly: What will be the output of following program? void main() { float b=7.4; printf("%d", (int)b);	9	None of the listed options	8	7	74	0	0	0	1	0	TEXT	TEXT	
Carefully read the question & answer accordingly: void main() { int arr[3][3]={{1,2},{3,4,5},{5,4}}; printf("%d %d %d",arr[0][2].arr[1][2].arr[2][1]); getch();	Compilation error	Runtime error	0.5,5	0.5,4	2,4,5	0	0	0	1	0	TEXT	TEXT	
What is the output of the given code snippet?	Value of elements in array	Base address of the array	First element of the array	Address of the last element of array		0	1	0	0	TEXT	TEXT		
Carefully read the question & answer accordingly: If you pass an array as an argument to a function, what actually gets passed?										TEXT	TEXT		
Carefully read the question & answer accordingly: What will be the output when you execute following code? void main() { char arr[11] = "Welcome To Programming"; printf("%s",arr); }	Welcome To	None of the given options	Welcome To P	Welcome	Compilation error	0	0	0	0	1	TEXT	TEXT	
Carefully read the question & answer accordingly: Size of the array need not be specified. When?	All of the options.	Initialization is a part of definition	It is a formal parameter	It is declaration		0	0.5	0.5	0	TEXT	TEXT		
Carefully read the question & answer accordingly: What will be output of the below program?										TEXT	TEXT		
int main() { char str[] = "Programming"; printf("%s", &str[2]); printf("%s", str); printf("%s", &str); return(0); }	programming Programming > Garbage Value >	Program Compiled with Syntax Errors	programming Programming Programming	Programming programming Programming		0	0	1	0	TEXT	TEXT		
Carefully read the question and answer accordingly. The term _____ means a separate physical computer system that host a part of the application architecture.	Layer	tier	None of the listed options	both		0	1	0	0	TEXT	TEXT		
Carefully read the question and answer accordingly. Which of the applications has most increased business usage over the internet?	World Wide Web (WWW)	None of the listed options	Electronic Data Interchange (EDI)	Extensible Markup Language (XML)	All the listed options	1	0	0	0	TEXT	TEXT		
Carefully read the question and answer accordingly. Which of the following DO NOT allow multiple applications to participate in a transaction?	Routers	Switches	Gateways	Firewalls	Net Beans.	0	0	1	0	TEXT	TEXT		
Carefully read the question and answer accordingly. In the Client-Server architecture, the component that makes a service request is	Network	Process	Server	Protocol	Client	0	0	0	0	1	TEXT	TEXT	
Carefully read the question and answer accordingly. One of the major disadvantages of a Client Server architecture	Lack of robustness	None of the listed options	All the listed options	Traffic congestion	Complex business logic involved	0.5	0	0	0.5	0	TEXT	TEXT	
Carefully read the question and answer accordingly. Every computer on the Internet that contains a Web site must have a _____	Business logic program	Database Server program	None of the listed options	Web Server program	File Server program	0	0	0	1	0	TEXT	TEXT	
Carefully read the question and answer accordingly. In 3-tier client/server applications, the business logic lies at _____	The client	Firewall	Middle Tier	Divided between client and server alternatively	The Database Server	0	0	1	0	0	TEXT	TEXT	

		Connects to several servers at the same time	Interacts with end user with GUI	Initiates a request	Processes requests	0	0.33	0.33	0.33	0	TEXT	TEXT
Carefully read the question and answer accordingly. Which of these are the functionalities of a Client	All the listed options	Real Simple Syndication (RSS)	IM (Instant Messaging).	Voice Over IP (VOIP), IPTV	None of the listed options	0	0	1	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. Call-free charges across the Internet between PCs and phones.	Time Server	Database Server	Transaction Server	File Server		1	0	0	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. Choose one from the following which is not a possible kind of server.												
Carefully read the question and answer accordingly. What are the benefits of employing Client-Server Computing in business ?	All the listed options		Better Sharing of critical database resources and other application services among clients throughout the network.	Faster Response and flexibility to changing environment of business world outside.	It is Easy to update and modernise system, both hardware and software as companies expanded and has new requirements.	1	0	0	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. Which of these are the functionalities of a Server	Processes and serves client requests	Wait for client requests	Interacts with end user with GUI	All the listed options	Accepts connections from large number of clients	0.33	0.33	0	0	0.33	TEXT	TEXT
Carefully read the question and answer accordingly. An N-tiered architecture _____ choose all answers applicable												
Carefully read the question and answer accordingly. The database server holds the following _____ and	Database Management systems	Database instances	All the given options		uses three sets of computers in which the client is responsible for presentation logic, one set of servers are responsible for data access logic and data storage, and application logic.	0	0.5	0.5	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. The computer that accepts, prioritizes, and processes print jobs is known as _____	print server	print client	mainframe	printer file server.	uses three sets of computers in which the clients are responsible for presentation logic, one set of servers are responsible for data access logic and data storage	0.5	0.5	0	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. In 3-tier computing, the request/response protocol between the clients and servers is _____	HTTP	POP3	NNTT	SMTP	MOM	1	0	0	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. One major disadvantage of an n-tiered client-server architecture is that it is much more difficult to program and test software	false	true				0	1				TEXT	TEXT
Carefully read the question and answer accordingly. A company _____ can be used to provide shared content for staff only	intranet	None of the listed options	extranet	opname	Internet	1	0	0	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. In the client/server architecture, the component that processes the request and sends the response is _____	Client	Network	Server	O.S.	Protocol	0	0	1	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. An advantage of the three-tier architecture over the two-tier architecture is _____	Better control over the data	Better performance	Easier maintenance	All the listed options		0	1	0	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. Which excel function is used to calculate the rates of return?	POWER(5^6)	POWER(5,6)	POWER(5^6)	POWER(5^6)		0	1	0	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. Workspace denotes _____	Group of worksheets	Statistical	Mathematical	All of the listed options	Financial Group of workbooks	0	0	0	1	0	TEXT	TEXT
Carefully read the question and answer accordingly. Choose an example of a function from the given list	C1:C5	ADD(C1:C5)		AVG(C1:C5)	C1:C5	0	0	0	1	0	TEXT	TEXT
Carefully read the question and answer accordingly. Choose an action from the given list to rename a worksheet	Worksheet cannot be renamed	Single Click on the Worksheet tab by holding CTRL key and type the new name	Double click on the Worksheet tab and type the new name	Single Click on the Worksheet tab by holding SHIFT key and type the new name	Option 2 -- SUM(G1:G9, H1:H9)	0	0	1	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. Choose the correct syntax of the SUM function in excel	All the listed options	Option 3 -- SUM(G1:H9)	option 1 -- SUM(G1,H1)	Both 2 and 3	None of the listed options	1	0	0	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. Choose an example of a formula from the given list	G1:G2	SUM(G1+G2)	ADD(G1:G2)	G1+G2		0	0	0	1	0	TEXT	TEXT
Carefully read the question and answer accordingly. What function in excel arranges row data in a column and column data in a row ?	Rows	Columns	Transpose	Index		0	0	1	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. Which chart is used to show the proportions of how one or more parts relate to one another	Column Chart	Pie chart	All the given options	Line Chart		0	1	0	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. Choose the types of conditions on which conditional formatting can be applied	Cell value based conditions	Function based conditions	All the listed options	Formulas based conditions		0.5	0	0	0.5	0	TEXT	TEXT
Carefully read the question and answer accordingly. Which excel function is used to display the current date in a cell?	now()	today()	time()	date()		0	1	0	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. Choose the function to find out the highest number among the 15 values entered in the column C	HIGH(C1:C15)	MAX(C1: C15)	MAX(C1, C15)	HIGH(C1: C15)		0	1	0	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. How do you reference the cell D7 on Sheet3 from Sheet2?	Sheet5!D7	Sheet5:D7	Sheet2:D7	Sheet2!D7		1	0	0	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. Choose the Order of operation in which Excel performs calculations in a formula	Comparison,Subtraction, Multiplication, Exponentiation	Multiplication, Exponentiation, Subtraction, Comparison	Subtraction, Comparison, Multiplication, Exponentiation	Exponentiation, Multiplication, Subtraction, Comparison		0	1	0	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. Choose the correct statements related to VLOOKUP function	VLOOKUP function can be used on a series of data in single column	VLOOKUP	The data in the table should be sorted in ascending order, while using VLOOKUP function	The data in the table need not be sorted, while using VLOOKUP function		0	0.5	0.5	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. Choose the types of data used in Excel	Both Options 1 and 2	Both Options 1 and 3	Option 2 -- Text, Values and Formulas	Option 3 -- Text, Values and Charts	Option 1 -- Text, Values and Functions	1	0	0	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. Excel stores dates and times as numbers, which enables its usage as functions and formulas. State True or False	false	true				0	1				TEXT	TEXT
Carefully read the question and answer accordingly. Which is the key used to select multiple non-adjacent cells in a worksheet, while clicking on them	Shift Key	CTRL Key	ALT key	None of the listed options	CTRL+Shift Key	0	1	0	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. Choose the correct syntax of the AVERAGE function in excel	Both Options 2 and 3	All the listed options	Option 1 --- AVERAGE (G1:H1)	Option 3 --- AVERAGE (G1:G9,H1:H9)	Option 2 -- AVERAGE (G1:H9)	0	1	0	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. What does the Error Value #NAME? specify in an excel?	All the listed options	A Function name is misspelled	A column is not wide enough to display the name	A value of the name is misspelled		0	1	0	0	0	TEXT	TEXT

Carefully read the question and answer accordingly. Choose the option to fit long addresses of multiple lines in a single cell?	Merge option	Text Wrap option Value_when_false	All the listed options None of the listed options	Shrink to Fit option		0	1	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. Which of these is not an argument of the IF function?	Logical test	only options 1 and 3	Two columns will be inserted after column L	Value_if_false option 1 --> option 2 --> option 3 -->	Value_if_false option 1 --> option 2 --> option 3 -->	0	1	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. Which set of charts can chart what can produce?	All the listed options	Two columns will be inserted after column N	Two columns will be inserted after column O	Bar charts Line graphs Pie charts	Bar charts Line graphs Pie charts	1	0	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. What is the resultant of the below action? Select both the columns M and N in a worksheet and choose -> Insert -> Insert Sheet Columns	Two columns will be inserted after column L	It counts cells with multiple criteria	It sums two cells with multiple arguments	All the listed options	It sums cells with values or labels	1	0	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. What is the function of COUNTIFS?	Whole number	All the listed options	Custom	Decimal Date	SUMIF (Sum_range, Criteria1, Criteria2, Range)	0	0	1	0	0	TEXT
Carefully read the question and answer accordingly. Choose the correct syntax of the SUMIF function	SUMIF(Number1, Number2...)	None of the listed options	SUMIF (Range, Criteria1, Sum_Range)			0	0	1	0	TEXT	TEXT
Carefully read the question and answer accordingly. K14-K28 Indicates values of :	Cells K14 through K28	Cells K14 and K28 only	Cells K15 and K27	None of the listed options		1	0	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. The method used to get data from a different worksheet is called as _____.	Accessing	Verifying	Referencing	Functioning		0	0	1	0	TEXT	TEXT
Carefully read the question and answer accordingly. What is the relationship of the below action? Select both rows 3 and 4 in a worksheet and choose -> Insert -> Insert Sheet Rows	Two rows will be inserted after Row 2	Two rows will be inserted after Row 5	Two rows will be inserted after Row 4	<a href="http://www.example.com">example</a>	Two rows will be inserted after Row 3	1	0	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. What is the correct HTML syntax for creating a hyperlink?	<a name="">A</a>	<ol>	<li>	<ol>	<a>B</a>	0	0	1	0	TEXT	TEXT
Carefully read the question and answer accordingly. How can you make a list that lists the items with numbers?	<ul>	<ol>	<li>	<ol>	<ol>	1	0	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. What does the Web browser assume while using a heading tag in a html document?	Heading information is to appear in bold letters	Heading information is to appear in its own line		Heading information has a hyperlink	Heading information is shown as a size six	0	1	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. If you don't want the frame windows to be resizable, simply add what to the lines ?	None of the listed options	desiresize	noresize	same	<embed>	0	0	1	0	TEXT	TEXT
Carefully read the question and answer accordingly. Choose the right syntax to embed "audio files" in HTML	<embed music="song.mid" width="500" height="10" >	0	0	1	0	TEXT	TEXT				
Carefully read the question and answer accordingly. Which is the correct syntax to create an Arabic numeric list?	<li type="1">	<ol type="1">	<li type="1">	<ol type="1">	<li type="1">	0	1	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. When creating a Web document, what format is used to express an image's height and width?	Dots per inch	Pixels	Inches	Centimeters		0	1	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. how many "browser safe colors" are there?	216	16 Million	256	None of the given options		0	0	1	0	TEXT	TEXT
Carefully read the question and answer accordingly. Identify the correct code for font,color and size.	<html><body><h1 style="font-family:verdana;">A</h1><p style="font-family:arial;color:red;font-size:20px;">A paragraph</p></body></html>	0	0	1	0	TEXT	TEXT				
Carefully read the question and answer accordingly. What does a HTML tag do?	None of the listed options	Refresh your content	Redirect to a new domain	rewrite url		0	0	1	0	TEXT	TEXT
Carefully read the question and answer accordingly. In which color is Anchor displayed by default?	hr	br	href	align		0	0	1	0	TEXT	TEXT
Carefully read the question and answer accordingly. What is the another way to make text bold besides <b>?	It specifies formatting and layout instructions for your web page	None of the listed options	It connects your web site to its operating environment.	It hides programming instructions from view.		1	0	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. Rather than using Hspace and Vspace can you use which of the following to add spacing to your image ?	green	blue	pink	red		0	1	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. How can you create an e-mail link in html?	All the listed options	<strong>	<fat>	<dark>		0	1	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. In which of the following is the mailto: tag used?	None of the listed options	height and width	1x1 pixel transparent image	align=+2		0	0	1	0	TEXT	TEXT
Carefully read the question and answer accordingly. How can you open a link in a new browser window?	3	2	1		<a href="mailto:a@b.com">a@b</a>	0	0	1	0	TEXT	TEXT
Carefully read the question and answer accordingly. Which of the following is the most important tool for adding colors to certain areas of the page rather than the entire background ?	cellspacing	cellpadding	row span	col span		0	0	1	0	TEXT	TEXT
Carefully read the question and answer accordingly. Which hml tag is used to display Preformatted texts?	<pre text=></pre text=>	<pre></pre>	<pre></pre>	<pre format=></pre format=>		0	1	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. what is the use of forms in html?	To provide animation effects	To display contents of email	To collect users input	None of the listed options		0	0	1	0	TEXT	TEXT
Carefully read the question and answer accordingly. Which of the following is a plain ASCII text file with embedded HTML commands?	Web document	Action document	Transaction document	Web Server		1	0	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. What is the most important tool for adding colors to certain areas of the page rather than the entire background ?	Images	Tables	Forts	None of the listed options		0	1	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. How can you open a link in a new browser window?	<a href="url" new>	None of the listed options	<a href="url" target="new" >		<a href="url" target="new" >	0	0	1	0	TEXT	TEXT
Carefully read the question and answer accordingly. Which link is used to start a list at the count of 3?	None of the listed options	None of the listed options	<ol begin="3">	<ol start="3">	<ol begin="3">	0	0	0	1	TEXT	TEXT
Carefully read the question and answer accordingly. What is the correct syntax to add alternative text for an image?		None of the listed options				0	0	0	1	TEXT	TEXT
Carefully read the question and answer accordingly. Using Hspace will add what to your image ?	Height to all sides	None of the listed options	Space to the left and right	Space to the top and bottom		0	0	1	0	TEXT	TEXT
Carefully read the question and answer accordingly. Which of the following is correct to align H1 tag to left alignment?	None of the given options	<h1 tag align = "left"></h1>	<h1 align = "left"></h1>	<h1 align = "left"></h1>		0	0	0	1	TEXT	TEXT
Carefully read the question and answer accordingly. Which tag would be used to display power in expression (A+B)2 ?	<p>	<SUP>	<SUB>	<b>		0	1	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. Choose the correct HTML syntax to left-align the content inside a table cell	<td align="left">	<tdleft>	<td leftalign>	<td valign="left">		1	0	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. A computer Program that translates one program instruction at a time into machine language is called _____	Interpreter	Compiler	CPU	Simulator		1	0	0	0	TEXT	TEXT



Carefully read the question and answer accordingly. State whether True or False. Context is made up of the contents of its registers and the memory that it is addressing.	true	false			1	0				TEXT	TEXT		
Carefully read the question and answer accordingly. In task manager of Windows 7 the unique process identifier is denoted as _____.	ProcessID	ProcID	PID	Disk and Execution Monitor	ProcessID	Deadlock & Executor Monitor	0	0	1	0	TEXT	TEXT	
Carefully read the question and answer accordingly. Daemon stands for _____.	Driver and Execution Monitor	None of the options				all of the specified option	0	0	1	0	TEXT	TEXT	
Carefully read the question and answer accordingly. In task manager of Windows 7 the unique process identifier is denoted as _____.	cooperative	interrogative			preemptive		0	0	0	1	TEXT	TEXT	
Carefully read the question and answer accordingly. In UNIX which command is used to list a job's Process ID?	"ps -l"	"job -l"	"jobs -l"	"jb -l"			0	0	1	0	TEXT	TEXT	
Carefully read the question and answer accordingly. Which command is used to bring a background process to the foreground?	"fg"	"fg"	"bf"	"bg"			0	1	0	0	TEXT	TEXT	
Carefully read the question and answer accordingly. Which command in UNIX is used to get process ID?	"pid"	"fetchpid"	"getid"	"getpid"			0	0	0	1	TEXT	TEXT	
Carefully read the question and answer accordingly. The total number of tasks that can run at any time does not depend on which of the following factors?	Network speed	Program Size	CPU Speed	Memory Size			1	0	0	0	TEXT	TEXT	
Carefully read the question and answer accordingly. State whether True or False. Multitasking is beneficial in critical applications involving huge business transactions.	false	true					1	0			TEXT	TEXT	
Carefully read the question and answer accordingly. State whether True or False. Time Slice is the period of time for which a process is allowed to run uninterrupted.	false	true					0	1			TEXT	TEXT	
Carefully read the question and answer accordingly. UNIX and Windows 7 uses which type of multitasking?	cooperative	preemptive			all of the specified option	interrogative	0	1	0	0	TEXT	TEXT	
Carefully read the question and answer accordingly. It implies that more than one CPU is involved.	None of the options	Multitasking	Multiprocessing	Multiprogramming			0	0	1	0	TEXT	TEXT	
Carefully read the question and answer accordingly. State whether True or False. A daemon is a long-running background process that answers requests for services	false	true					0	1			TEXT	TEXT	
Carefully read the question and answer accordingly. In UNIX, multitasking, each program can control the CPU for as long as it needs it.	interrogative	cooperative			all of the specified option	preemptive	0	1	0	0	TEXT	TEXT	
Carefully read the question and answer accordingly. State whether True or False.	false	true					0	1			TEXT	TEXT	
The Unix kernel maintains track of many processes at once, dividing its time between the jobs submitted to it.	ps	ps	prcs	prc	ps - ef   grep username	ps - ef   grep username	1	0	0	0	TEXT	TEXT	
Carefully read the question and answer accordingly. Which command lists all running processes in UNIX?	ps - ef   grep username	prc - ef   grep username	ps - ef   grep username	prc - ef   grep username			1	0	0	0	TEXT	TEXT	
Carefully read the question and answer accordingly. Which command is used to list down all the processes owned by you?	ps - ef   grep username	ps - ef   grep username	ps - ef   grep username	ps - ef   grep username			0	1	0	0	TEXT	TEXT	
Carefully read the question and answer accordingly. In UNIX which command is used to get information on CPU Model, CPU MHz, CPU Cores, Address size etc?	"proc/cpu"	"proc/sysinfo"	"proc/cpufreq"	"proc/cpudetails"			0	0	1	0	TEXT	TEXT	
Carefully read the question and answer accordingly. Which option is used to get long listing details of all running processes in UNIX?	"-d"	"-l"	"-d"	"-e"			0	0	1	0	TEXT	TEXT	
Carefully read the question and answer accordingly. UNIX lets us run more than one program inside the terminal, which is called as _____.	Control Job	Control MultiJob	MultiJob Control	Job Control	Simple Multiprogramming	Symmetric Multiprocessing	0	0	0	1	TEXT	TEXT	
Carefully read the question and answer accordingly. With respect to Android OS what does SMP stands for?	Simple Multiprocessing						0	0	0	1	TEXT	TEXT	
Carefully read the question and answer accordingly. Which command is used to view the disk usage in UNIX?	du	cd	df	rm			0	0	1	0	TEXT	TEXT	
Carefully read the question and answer accordingly. Which of the following are valid types of physical storage used in UNIX? (Choose three)	Tertiary	Auxiliary	Secondary	Central	Primary		0	0.5	0	0.5	0	TEXT	TEXT
Carefully read the question and answer accordingly. How can we view the Kernel Paged & NonPaged memory in windows 7 operating system?	Task Manager – Performance	Task Manager – Processes	Task Manager – Application	Task Manager – Cpu Usage			1	0	0	0	TEXT	TEXT	
Carefully read the question and answer accordingly. Which command is used to view disk usage by a directory or file in UNIX?	rm	cd	df	du			0	0	0	1	TEXT	TEXT	
Carefully read the question and answer accordingly. Which type of Memory allocation assigns consecutive memory blocks to a process	Contiguous memory allocation	Paged Memory Management	None of the options.	Virtual Memory			1	0	0	0	TEXT	TEXT	
Carefully read the question and answer accordingly. The process of translating virtual addresses into real addresses is called _____.	None of the options.	paging	swapping	mapping			0	0	0	1	TEXT	TEXT	
Carefully read the question and answer accordingly. The _____ command will display a continually updating report of system resource usage in UNIX.	top	df	du	rm			1	0	0	0	TEXT	TEXT	
Carefully read the question and answer accordingly. Each process on 32-bit Microsoft Windows has its own virtual address space and variables addressing up to _____ of memory.	32 gigabytes	8 gigabytes	16 gigabytes	4 gigabytes			0	0	0	1	TEXT	TEXT	
Carefully read the question and answer accordingly. Which type of Memory allocation uses a space that is larger than the actual amount of RAM present by temporarily transferring some contents from RAM to a disk.	Paged Memory Management	Virtual Memory	None of the options.	Contiguous memory allocation			0	1	0	0	TEXT	TEXT	
Carefully read the question and answer accordingly. A block of External Storage is called as _____.	None of the options.	Pages	Frames	Slots			0	0	0	1	TEXT	TEXT	
Carefully read the question and answer accordingly. A block of Virtual Storage is called as _____.	None of the options.	Frames	Slots	Pages			0	0	0	1	TEXT	TEXT	
Carefully read the question and answer accordingly. The _____ of a process is the collection of pages in the virtual address space of the process that are currently resident in physical memory.	working set	paging set	group set	map set			1	0	0	0	TEXT	TEXT	
Carefully read the question and answer accordingly. The Operating System is responsible for allocating primary memory to processes, and for assisting the programmer in loading and storing the contents of the primary memory is termed as _____.	None of the options.	Storage Management	Memory Management	Data Management			0	0	1	0	TEXT	TEXT	
Carefully read the question and answer accordingly. _____ is an area of memory used for dynamic memory allocation. It handles the dynamic memory needs of a program.	Thread	Array	Stack	Heap			0	0	0	1	TEXT	TEXT	
Carefully read the question and answer accordingly. _____ is portion of memory, meant to store the local variables and process registers of an application program.	Heap	Array	Stack	Thread			0	0	1	0	TEXT	TEXT	
Carefully read the question and answer accordingly. State whether True or False. Primary memory is a limited resource that cannot contain all active processes in the system.	false	true					0	1			TEXT	TEXT	
Carefully read the question and answer accordingly. With respect to Virtual Storage in zOS: A block of Internal Storage is called as _____.	Slots	None of the options.	Frames	Pages			0	0	1	0	TEXT	TEXT	
Carefully read the question and answer accordingly. State whether True or False. In UNIX operating system du command along with -h option is used to obtain the size of files in human readable form.	true	false					0	1			TEXT	TEXT	
Carefully read the question and answer accordingly. Which type of Memory allocation supports paging that causes every logical address to be translated to a physical address.	Virtual Memory	Contiguous memory allocation	Paged Memory Management	None of the options.			0	0	1	0	TEXT	TEXT	
Carefully read the question and answer accordingly. There are _____ ways of memory allocations.	two	five	six	three			0	0	0	1	TEXT	TEXT	
Carefully read the question and answer accordingly. In Operating System (OS) PCB Stands for _____.	Process control Blocks	None of the options	Program control Blocks	Program content Blocks			1	0	0	0	TEXT	TEXT	

Carefully read the question and answer accordingly. A programmer interacts with the _____.	Operating System and its Utilities	Computer Hardware and Operating Systems	Computer Hardware and its Utilities	Computer Hardware and Applications		1	0	0	0	TEXT	TEXT	
Carefully read the question and answer accordingly. Which of the following is a type of operating system?	Distributed	All of them	Multi-user	Embedded	Real-time	0	1	0	0	TEXT	TEXT	
Carefully read the question and answer accordingly. Which component of Operating System (OS) is providing CPU Scheduling, Process management, Memory management and communication between hardware and software components?	Shell	Kernel	Debugger	Command Processor		0	1	0	0	TEXT	TEXT	
Carefully read the question and answer accordingly. State whether True or False. Performance monitors fall under utilities of Operating System (OS).	true	false				1	0			TEXT	TEXT	
Carefully read the question and answer accordingly. In Operating System (OS) which program interprets the command typed by user?	Compiler	Debugger	Kernel	Shell		0	0	0	1	TEXT	TEXT	
Carefully read the question and answer accordingly. There are _____ components in the Structure of an Operating System (OS).	two	one	four	three		0	0	0	1	TEXT	TEXT	
Carefully read the question and answer accordingly. State whether True or False. Editors fall under utilities of Operating System (OS).	false	true				0	1			TEXT	TEXT	
Carefully read the question and answer accordingly. Deadlock handling mechanism fails under what key function of Operating System (OS) ?	I/O System Management	Memory Management	Process Management	File Management		0	0	1	0	TEXT	TEXT	
Carefully read the question and answer accordingly. With respect to z/OS what does ISPF Stands for?	Interactive System Productivity Facility	Intelligent Software Productivity Facility	Intelligent System Productivity Facility	None of the listed options	Interactive Software Productivity Facility	1	0	0	0	TEXT	TEXT	
Carefully read the question and answer accordingly. State whether True or False. Operating System (OS) ensures security by validating the users while any process is invoked.	false	true				0	1			TEXT	TEXT	
Carefully read the question and answer accordingly. Which of the following runs on computer hardware and serve as platform for other software to run on?	System Software	Application Software	Operating System	All of them		0	0	1	0	TEXT	TEXT	
Carefully read the question and answer accordingly. State whether True or False. Operating System (OS) creates a process when an instance of a program is loaded into the main memory.	false	true				0	1			TEXT	TEXT	
Carefully read the question and answer accordingly. Which of the following is a type of operating system used in mobile phones?	Batch Processing	RTOS	Distributed	Interactive Processing		0	0	0	1	TEXT	TEXT	
Carefully read the question and answer accordingly. Which of the following are NOT components in the Structure of an Operating System (OS)?	Kernel	Debugger	Compiler	Command Processor	Shell	0	0.5	0.5	0	TEXT	TEXT	
Carefully read the question and answer accordingly. Which of the following is a type of operating system used in stock and billing systems?	Distributed	Interactive Processing	RTOS	none of the listed options	Batch Processing	0	0	0	1	TEXT	TEXT	
Carefully read the question and answer accordingly. Operating System is a _____ Software.	both System & Application	Application	Operating System	System		0	0	0	1	TEXT	TEXT	
Carefully read the question and answer accordingly. State whether True or False. A set of threads can be deadlocked if each process in the set is waiting for an event that only another process in the set can cause.	false	true				0	1			TEXT	TEXT	
Carefully read the question and answer accordingly. With respect to z/OS what does TSO Stands for?	Timer Set Option	Time Slicing Option	Non of the options	Time Sharing Option		0	0	0	1	TEXT	TEXT	
Carefully read the question and answer accordingly. Choose the correct order from top to bottom.	I. APPS II. OPERATING SYSTEM III. COMPUTER HARDWARE	I. APPS II. OPERATING SYSTEM III. COMPUTER HARDWARE	I. APPS II. OPERATING SYSTEM III. COMPUTER HARDWARE	None of the listed options		0	0	1	0	TEXT	TEXT	
Carefully read the question and answer accordingly. Which of the following is not considered as key function of Operating System (OS).	Memory Management	File Management	I/O System Management	Process Management		0	0	0	0	1	TEXT	TEXT
Carefully read the question and answer accordingly. State whether True or False. z/OS is an operating system developed by IBM for mainframe computers.	true	false				1	0			TEXT	TEXT	
Carefully read the question and answer accordingly. State whether True or False. In GUI the user selects a picture or icons available to perform a task.	true	false				1	0			TEXT	TEXT	
Carefully read the question and answer accordingly. State which statement below are true.	Statement I : An event-driven system design of RTOS switches the tasks based on their priorities. Statement II : A time-slicing system design of RTOS switch tasks based on clock interrupts.	Both Statement I & II are FALSE	Both Statement I & II are TRUE	Both Statement I & II are TRUE		0	0	0	1	TEXT	TEXT	
Carefully read the question and answer accordingly. _____ is an interface between the user and Operating System (OS).	Shell	Kernel	Debugger	Command Processor		0	0	0	1	TEXT	TEXT	
Carefully read the question and answer accordingly. State whether True or False. z/OS is NOT a Multitasking Operating System (OS).	false	true				1	0			TEXT	TEXT	
Carefully read the question and answer accordingly. Which of the following falls under Command Processor?	both CLI & GUI	GUI	CLI	None of the listed options		1	0	0	0	TEXT	TEXT	
Carefully read the question and answer accordingly. Scheduling is the process of messages sent to a processor.	sharing	controlling & prioritizing	controlling & sharing	prioritizing	controlling	0	1	0	0	TEXT	TEXT	
Carefully read the question and answer accordingly. Which of the following are most common reasons for a context switch?	When a running thread needs to wait, it relinquishes the remainder of its time slice.	all of the specified option	A thread with a higher priority has become ready to run.	A running thread needs to wait.	The time slice has elapsed.	0	1	0	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. _____ uses multiple queues to select the next process, out of the processes in memory, to get a time quantum.	High level (memory) scheduler	Low level (CPU) scheduler	both High level (memory) & Low level (CPU) scheduler	None of the listed options		0	1	0	0	TEXT	TEXT	
Carefully read the question and answer accordingly. State whether True or False. In Windows 7 the Task Scheduler wizard is used to create a task that opens the program automatically according to the schedule you choose.	false	true				0	1			TEXT	TEXT	
Carefully read the question and answer accordingly. The steps in a context switch are randomly arranged below. Identify the correct order.	I. Push the thread that was finished executing at the end of the queue for its priority II. Save the context of the thread that just finished executing. III. Remove the thread at the head of the queue, load its context, and execute it. IV. Find the highest priority queue that contains ready threads.	I -> III -> IV -> II	II -> I -> IV -> III	II -> I -> III -> IV -> I	II -> III -> IV -> I	0	1	0	0	TEXT	TEXT	
Carefully read the question and answer accordingly. In Windows 7 the priority levels range from _____.	None of the options	zero (lowest) to 10 (highest).	zero (lowest) to 15 (highest).	zero (lowest) to 31 (highest).		0	0	0	1	TEXT	TEXT	
Carefully read the question and answer accordingly. The scheduler maintains a _____ of executable threads for each priority level.	queue	stack	None of the options	heap		1	0	0	0	TEXT	TEXT	
Carefully read the question and answer accordingly. Which of the following is not goals of scheduling?	Make sure each process is completed within a reasonable time frame	Eliminate highs & lows in the processor's workload	To maintain a constant amount of work for the processor.	Make sure each process completes specific amount of memory.		0	0	0	1	TEXT	TEXT	
Carefully read the question and answer accordingly. State whether True or False. Processes scheduling in which the scheduler selects tasks to run based on their priority is called Priority Scheduling.	Two	one	Four	Three		1	0	0	0	TEXT	TEXT	
Carefully read the question and answer accordingly. In z/OS there are a maximum of ____ name segments.	50	100	500	2000		1	0	0	0	TEXT	TEXT	

Carefully read the question and answer accordingly. Which command is used to search for information in a file or files?	search	None of the listed options	find	grep	grp		0	0	0	1	0	TEXT	TEXT
Carefully read the question and answer accordingly. sometimes referred to as file descriptors, are data structures that hold information about a file.	Fix Control Blocks	File Control Blocks	File Content Blocks	File Content Blocks			0	1	0	0		TEXT	TEXT
Carefully read the question and answer accordingly. State whether True or False. The command "mv" is used to move and also rename a file.	false	true	FILE_NAME_MAX	FILE_MAX	NAME_MAX		0	1				TEXT	TEXT
Carefully read the question and answer accordingly. In UNIX a file name may not exceed _____.	None of the options.						0	0	0	1		TEXT	TEXT
Carefully read the question and answer accordingly. Which command is used to find out what directory you are working in?	pwd	more	ncftp	print		Only PS & PDS	1	0	0	0		TEXT	TEXT
Carefully read the question and answer accordingly. Which of the following is a valid type of File in z/OS.	All PS,PDS & VSAM	PS	VSAM	PDS			1	0	0	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. Which of the following is the Korn shell ( ksh ) initialization script.	.kshrcs	.kshrc	.kshsr	.kshc			0	1	0	0		TEXT	TEXT
Carefully read the question and answer accordingly. State whether True or False. Partitioned Data Set is similar to File in Windows.	true	false					0	1				TEXT	TEXT
Carefully read the question and answer accordingly. State whether True or False. Files are used to store a variety of different types of information, such as programs, documents, spreadsheets, video, sounds, pictures and record-based data.	true	false					1	0				TEXT	TEXT
Carefully read the question and answer accordingly. State which statement below is true. Statement I: "r" Displays files in reverse order. Statement II: "ls -R" Displays subdirectories as well.	Statement I is FALSE & Statement II is TRUE	Both Statement I & Statement II are TRUE	Both Statement I & II are FALSE				0	0	1	0		TEXT	TEXT
Carefully read the question and answer accordingly. File ownership is an important component of an OS that provides a secure method for storing files. There are Total _____ number of permissions on a file which can be defined in UNIX.	8	10	12	9			0	0	0	1		TEXT	TEXT
Carefully read the question and answer accordingly. In mainframe system the block of data is called as a tablespace	tablespace	table	Record	cell			0	0	1	0		TEXT	TEXT
Carefully read the question and answer accordingly. What does VSAM stands for?	crt	ctr	cat	create			0	0	1	0		TEXT	TEXT
Carefully read the question and answer accordingly. Which command is used to change the permissions of a file or directory?	Virtual System Advancement Method	Virtual Storage Advancement Method	Virtual Storage Access Method	Virtual System Access Method			0	0	1	0		TEXT	TEXT
Carefully read the question and answer accordingly. Which of the following is the C shell ( csh ) initialization script.	chmod	change	cmhd	chmd			1	0	0	0		TEXT	TEXT
Carefully read the question and answer accordingly. Which command is used if you want to work on a computer different from the one you are currently working on because that remote machine might be faster.	.cshrc	.cshc	.cshrs	.cshrc			0	0	0	1		TEXT	TEXT
Carefully read the question and answer accordingly. A dataset in z/OS should have a Unique name of Max. characters.	chmd	pwd	grep	rsh			0	0	0	1		TEXT	TEXT
Carefully read the question and answer accordingly. Which option is specified to display invisible files in "ls" command?	64	40	33	44			0	0	0	1		TEXT	TEXT
Carefully read the question and answer accordingly. command lists all files in the directory that match the name if specified.	"d"	".b"	".a"	".f"			0	0	1	0		TEXT	TEXT
Carefully read the question and answer accordingly. In UNIX, command lists all files in the directory that match the name if specified.	rm	ts	top	ls		None of the options.	0	0	0	1		TEXT	TEXT
Carefully read the question and answer accordingly. Files are termed as _____ in z/OS.	Record	Datasets	table				0	1	0	0		TEXT	TEXT
Carefully read the question and answer accordingly. State whether True or False. Files can be removed by the "rm" command.	false	true					1	0				TEXT	TEXT
Carefully read the question and answer accordingly. What does FCB stands for ?	Fic Content Blocks	File Control Blocks	Fix Control Blocks	File Content Blocks			0	1	0	0		TEXT	TEXT
Carefully read the question and answer accordingly. Which of the following is the Bourne shell ( sh ) initialization script.	:rhosts	.cshrc	.kshrc	profile			0	0	0	1		TEXT	TEXT
Carefully read the question and answer accordingly. Which of the following is not hidden/invisible file in UNIX?	kshrc	profile	:rhosts	.cshrcs			0	0	0	1		TEXT	TEXT
Carefully read the question and answer accordingly. State whether True or False. Physical Sequential dataset is similar to Folder in Windows	false	true					1	0				TEXT	TEXT
Carefully read the question and answer accordingly. Which command is used to remove a directory?	Clean	rm	remove	rmkdir			0	0	0	1		TEXT	TEXT
Carefully read the question and answer accordingly. Which of the following is not valid permission of a file or folder in UNIX?	Read	Write	Read & execute	None of the Options	Full control		0	0	0	1	0	TEXT	TEXT
Carefully read the question and answer accordingly. Thread lifecycle has how many states?	3	6	2	5	4		0	0	0	1	0	TEXT	TEXT
Carefully read the question and answer accordingly. A runnable thread enters the state when it completes its task or otherwise terminates.	Timed Waiting	Waiting	New	Terminated	Runnable		0	0	0	1	0	TEXT	TEXT
Carefully read the question and answer accordingly. State whether True or False. A thread transitions back to the runnable state only when another thread signals the waiting thread to continue executing.	true	false					1	0				TEXT	TEXT
Carefully read the question and answer accordingly. A thread in _____ state is considered to be executing its task.	Runnable	Terminated	Waiting	Timed Waiting	New		1	0	0	0	0	TEXT	TEXT
Carefully read the question and answer accordingly. State whether True or False. Multithreading is an operating system to concurrently run programs that have been divided into subcomponents or threads.	false	true					0	1				TEXT	TEXT
Carefully read the question and answer accordingly. State whether True or False. Multithreading in an interactive application may allow a program to continue running even if part of it is blocked or is performing a lengthy operation, thereby increasing responsiveness to the user.	true	false					1	0				TEXT	TEXT
Carefully read the question and answer accordingly. State whether True or False. Threads in Windows 7 are represented as objects that are created, maintained, and destroyed by the Process Manager.	false	true					0	1				TEXT	TEXT
Carefully read the question and answer accordingly. State whether True or False. All thread of a single process does not share the same resources that are assigned to their corresponding processes.	true	false					0	1				TEXT	TEXT
Carefully read the question and answer accordingly. State whether True or False. Thread refers to a path through a program's instructions that can be scheduled for execution separately	false	true					0	1				TEXT	TEXT
Carefully read the question and answer accordingly. A thread in _____ state transitions back to the runnable state when that time interval expires or when the event it is waiting for occurs	Waiting	Terminated	Timed Waiting	New	Runnable		0	0	1	0	0	TEXT	TEXT

## ZeeZee Bank

Account.java

```
public class Account {  
    private long accountNumber;  
    private double balanceAmount;  
  
    public Account(long accountNumber, double balanceAmount) {  
        this.accountNumber = accountNumber;  
        this.balanceAmount = balanceAmount;  
    }  
  
    public long getAccountNumber() {  
        return accountNumber;  
    }  
  
    public void setAccountNumber(long accountNumber) {  
        this.accountNumber = accountNumber;  
    }  
  
    public double getBalanceAmount() {  
        return balanceAmount;  
    }  
  
    public void setBalanceAmount(double balanceAmount) {  
        this.balanceAmount = balanceAmount;  
    }  
  
    public void deposit(double depositAmount) {  
        balanceAmount += depositAmount;  
    }  
  
    public boolean withdraw(double withdrawAmount) {  
        if (withdrawAmount <= balanceAmount) {  
            balanceAmount -= withdrawAmount;  
            return true;  
        }  
        return false;  
    }  
}
```

Main.java

```
import java.text.DecimalFormat;
```

```

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        DecimalFormat decimalFormat = new DecimalFormat("0.00");

        System.out.println("Enter the account number:");
        long accountNumber = scanner.nextLong();

        System.out.println("Enter initial balance:");
        double balanceAmount = scanner.nextDouble();

        Account account = new Account(accountNumber, balanceAmount);

        System.out.println("Enter the amount to be deposited:");
        double depositAmount = scanner.nextDouble();
        account.deposit(depositAmount);
        double availableBalance = account.getBalanceAmount();

        System.out.println("Available balance is:" + decimalFormat.format(availableBalance));

        System.out.println("Enter the amount to be withdrawn:");
        double withdrawAmount = scanner.nextDouble();
        boolean isWithdrawn = account.withdraw(withdrawAmount);
        availableBalance = account.getBalanceAmount();

        if (!isWithdrawn) {
            System.out.println("Insufficient balance");
        }

        System.out.println("Available balance is:" + decimalFormat.format(availableBalance));
    }
}

```

### Numerology number

```

Main.java
import java.util.Scanner;

public class Main {
    private static int getSum(long num) {
        char[] chars = Long.toString(num).toCharArray();

```

```
int sum = 0;

for (char ch : chars) {
    sum += Character.digit(ch, 10);
}

return sum;
}

private static int getNumerology(long num) {
    String string = String.valueOf(num);

    while (string.length() != 1) {
        string = String.valueOf(getSum(Long.parseLong(string)));
    }

    return Integer.parseInt(string);
}

private static int getOddCount(long num) {
    int oddCount = 0;

    for (char ch : Long.toString(num).toCharArray()) {
        if (Character.digit(ch, 10) % 2 != 0) {
            ++oddCount;
        }
    }

    return oddCount;
}

private static int getEvenCount(long num) {
    int evenCount = 0;

    for (char ch : Long.toString(num).toCharArray()) {
        if (Character.digit(ch, 10) % 2 == 0) {
            ++evenCount;
        }
    }

    return evenCount;
}

public static void main(String[] args) {
```

```

Scanner scanner = new Scanner(System.in);

System.out.println("Enter the number");
long num = scanner.nextLong();

System.out.println("Sum of digits");
System.out.println(getSum(num));

System.out.println("Numerology number");
System.out.println(getNumerology(num));

System.out.println("Number of odd numbers");
System.out.println(getOddCount(num));

System.out.println("Number of even numbers");
System.out.println(getEvenCount(num));
}
}

```

### **Substitution Cipher Technique**

```

Main.java
import java.util.*;

public class Main {
    public static void main(String[] args) {
        StringBuilder stringBuilder = new StringBuilder();
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter the encrypted text:");
        String text = scanner.nextLine();
        char[] chars = text.toCharArray();
        boolean flag = false;

        for (char ch : chars) {
            if (Character.isLetter(ch)) {
                flag = true;

                if (Character.isLowerCase(ch)) {
                    int sub = (int) ch - 7;

                    if (sub < 97) {
                        ch = (char) (122 - (97 - sub) + 1);
                    } else {

```

```

        ch = (char) sub;
    }
} else if (Character.isUpperCase(ch)) {
    int sub = (int) ch - 7;

    if (sub < 65) {
        ch = (char) (90 - (65 - sub) + 1);
    } else {
        ch = (char) sub;
    }
}

stringBuilder.append(ch);
} else if (Character.isWhitespace(ch)) {
    stringBuilder.append(ch);
}
}

if (flag) {
    System.out.println("Decrypted text:");
    System.out.println(stringBuilder.toString());
} else {
    System.out.println("No hidden message");
}
}
}
}

```

### Bank Account - Interface

Account.java

```

public class Account {
    private String accountNumber;
    private String customerName;
    private double balance;

    public Account(String accountNumber, String customerName, double balance) {
        this.accountNumber = accountNumber;
        this.customerName = customerName;
        this.balance = balance;
    }

    public String getAccountNumber() {
        return accountNumber;
    }
}

```

```
public void setAccountNumber(String accountNumber) {
    this.accountNumber = accountNumber;
}

public String getCustomerName() {
    return customerName;
}

public void setCustomerName(String customerName) {
    this.customerName = customerName;
}

public double getBalance() {
    return balance;
}

public void setBalance(double balance) {
    this.balance = balance;
}
}
```

#### CurrentAccount.java

```
public class CurrentAccount extends Account implements MaintenanceCharge {
    public CurrentAccount(String accountNumber, String customerName, double balance) {
        super(accountNumber, customerName, balance);
    }

    @Override
    public float calculateMaintenanceCharge(float noOfYears) {
        return (100.0f + noOfYears) + 200.0f;
    }
}
```

#### MaintenanceCharge.java

```
public interface MaintenanceCharge {
    float calculateMaintenanceCharge(float noOfYears);
}
```

#### SavingsAccount.java

```
public class SavingsAccount extends Account implements MaintenanceCharge {
    public SavingsAccount(String accountNumber, String customerName, double balance) {
        super(accountNumber, customerName, balance);
    }
}
```

```
@Override
public float calculateMaintenanceCharge(float noOfYears) {
    return (50.0f * noOfYears) + 50.0f;
}
}

UserInterface.java
import java.text.DecimalFormat;
import java.util.Scanner;

public class UserInterface {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        DecimalFormat decimalFormat = new DecimalFormat("0.0");

        System.out.println("1. Savings Account");
        System.out.println("2. Current Account");
        System.out.println("Enter your choice:");
        int choice = scanner.nextInt();

        System.out.println("Enter the Account number");
        String accountNumber = scanner.next();

        System.out.println("Enter the Customer Name");
        String customerName = scanner.next();

        System.out.println("Enter the Balance amount");
        double balance = scanner.nextDouble();

        System.out.println("Enter the number of years");
        int noOfYears = scanner.nextInt();

        System.out.println("Customer Name " + customerName);
        System.out.println("Account Number " + accountNumber);
        System.out.println("Account Balance " + decimalFormat.format(balance));

        switch (choice) {
            case 1: {
                SavingsAccount savingsAccount = new SavingsAccount(accountNumber,
customerName, balance);
                System.out.println("Maintenance Charge for Savings Account is Rs " +
decimalFormat.format(savingsAccount.calculateMaintenanceCharge(noOfYears)));
                break;
            }
        }
    }
}
```

```

        case 2: {
            CurrentAccount currentAccount = new CurrentAccount(accountNumber,
customerName, balance);
            System.out.println("Maintenance Charge for Current Account is Rs " +
decimalFormat.format(currentAccount.calculateMaintenanceCharge(noOfYears)));
        }
    }
}
}

```

## Batting Average

```

UserInterface.java
package com.ui;

import com.utility.Player;

import java.util.ArrayList;
import java.util.Scanner;

public class UserInterface {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Player player = new Player();
        player.setScoreList(new ArrayList<>());
        boolean flag = true;

        while (flag) {
            System.out.println("1. Add Runs Scored");
            System.out.println("2. Calculate average runs scored");
            System.out.println("3. Exit");
            System.out.println("Enter your choice");
            int choice = scanner.nextInt();

            switch (choice) {
                case 1: {
                    System.out.println("Enter the runs scored");
                    int score = scanner.nextInt();
                    player.addScoreDetails(score);
                    break;
                }
                case 2: {
                    System.out.println("Average runs secured");
                }
            }
        }
    }
}

```

```
        System.out.println(player.getAverageRunScored());
        break;
    }
    case 3: {
        System.out.println("Thank you for use the application");
        flag = false;
        break;
    }
}
}
}
}
```

Player.java

```
package com.utility;
```

```
import java.util.List;
```

```
public class Player {
```

```
    private List<Integer> scoreList;
```

```
    public List<Integer> getScoreList() {
```

```
        return scoreList;
```

```
}
```

```
    public void setScoreList(List<Integer> scoreList) {
```

```
        this.scoreList = scoreList;
```

```
}
```

```
    public double getAverageRunScored() {
```

```
        if (scoreList.isEmpty()) {
```

```
            return 0.0;
```

```
}
```

```
        int size = scoreList.size();
```

```
        int totalScore = 0;
```

```
        for (int score : scoreList) {
```

```
            totalScore += score;
```

```
}
```

```
        return (double) totalScore / (double) size;
```

```
}
```

```

    public void addScoreDetails(int score) {
        scoreList.add(score);
    }
}

```

### Grade Calculation

```

Main.java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter the number of Threads:");
        int n = scanner.nextInt();

        GradeCalculator[] gradeCalculators = new GradeCalculator[n];
        Thread[] threads = new Thread[n];

        for (int i = 0; i < n; ++i) {
            System.out.println("Enter the String:");
            String string = scanner.next();
            String[] strings = string.split(":");
            int[] marks = new int[5];

            String studName = strings[0];

            for (int j = 1; j < 6; ++j) {
                marks[j - 1] = Integer.parseInt(strings[j]);
            }

            gradeCalculators[i] = new GradeCalculator(studName, marks);
            threads[i] = new Thread(gradeCalculators[i]);
            threads[i].start();
            threads[i].interrupt();
        }

        for (int i = 0; i < n; ++i) {
            System.out.println(gradeCalculators[i].getStudName() + ":" +
gradeCalculators[i].getResult());
        }
    }
}

```

```
Gradecalculator.java
public class GradeCalculator extends Thread {
    private String studName;
    private char result;
    private int[] marks;

    public GradeCalculator(String studName, int[] marks) {
        this.studName = studName;
        this.marks = marks;
    }

    public String getStudName() {
        return studName;
    }

    public void setStudName(String studName) {
        this.studName = studName;
    }

    public char getResult() {
        return result;
    }

    public void setResult(char result) {
        this.result = result;
    }

    public int[] getMarks() {
        return marks;
    }

    public void setMarks(int[] marks) {
        this.marks = marks;
    }

    @Override
    public void run() {
        int totalMarks = 0;

        for (int mark : marks) {
            totalMarks += mark;
        }
    }
}
```

```

if (totalMarks <= 500 && totalMarks >= 400) {
    result = 'A';
} else if (totalMarks < 400 && totalMarks >= 300) {
    result = 'B';
} else if (totalMarks < 300 && totalMarks >= 200) {
    result = 'C';
} else if (totalMarks < 200 && totalMarks >= 0) {
    result = 'E';
}
}
}

```

### **Employees eligible for promotionCoding exercise**

Main.java

```

import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Scanner;
import java.util.stream.Collectors;
import java.util.stream.IntStream;

class Employee implements Comparable<Employee> {
    private final String id;
    private final LocalDate joiningDate;
    private boolean isEligible;

    public Employee(String id, LocalDate joiningDate) {
        this.id = id;
        this.joiningDate = joiningDate;
    }

    public void setEligible(LocalDate now) {
        isEligible = joiningDate.until(now, ChronoUnit.YEARS) >= 5;
    }

    public boolean getEligible() {
        return isEligible;
    }

    public String getId() {
        return id;
    }
}

```

```

}

@Override
public String toString() {
    return id;
}

@Override
public int compareTo(Employee employee) {
    return this.id.compareTo(employee.getId());
}
}

public class Main {
    public static void main(String[] args) throws Exception {
        Scanner scanner = new Scanner(System.in);
        DateTimeFormatter dateTimeFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
        LocalDate now = LocalDate.parse("01/01/2019", dateTimeFormatter);
        int n = scanner.nextInt();
        ArrayList<Employee> employees = new ArrayList<>();

        IntStream.rangeClosed(1, 4).forEach(i -> {
            String id = scanner.next();
            String joiningDateStr = scanner.next();

            try {
                LocalDate joiningDate = LocalDate.parse(joiningDateStr, dateTimeFormatter);
                Employee employee = new Employee(id, joiningDate);
                employee.setIsEligible(now);
                employees.add(employee);
            } catch (Exception ignore) {
                System.out.println("Invalid date format");
                System.exit(0);
            }
        });

        List<Employee> filteredEmployees =
        employees.stream().filter(Employee::getIsEligible).collect(Collectors.toList());

        if (filteredEmployees.isEmpty()) {
            System.out.println("No one is eligible");
        } else {
            Collections.sort(filteredEmployees);
            filteredEmployees.forEach(System.out::println);
        }
    }
}

```

```
    }
}
}
```

### Check Number Type

NumberType.java

```
public interface NumberType {
    boolean checkNumber(int num);
}
```

NumberTypeUtility.java

```
import java.util.Scanner;

public class NumberTypeUtility {
    public static NumberType idOdd() {
        return (num) -> num % 2 != 0;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int num = scanner.nextInt();

        if (idOdd().checkNumber(num)) {
            System.out.println(num + " is odd");
        } else {
            System.out.println(num + " is not odd");
        }
    }
}
```

### Retrieve Flight details based on source and destination

Main.java

```
import java.util.*;
public class Main{
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the source");
        sc.next();
        String source=sc.nextLine();
        System.out.println("Enter the destination");
        String dest=sc.nextLine();
        FlightManagementSystem obj=new FlightManagementSystem();
```

```

ArrayList<Flight> res=obj.viewFlightsBySourceDestination(source,dest);
if(res!=null)
    System.out.println(res);
else
    System.out.println("No flights available for the given source and destination");
}
}

```

```

DB.java
import java.io.FileInputStream;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;

public class DB {

    private static Connection con = null;
    private static Properties props = new Properties();

    //ENSURE YOU DON'T CHANGE THE BELOW CODE WHEN YOU SUBMIT
    public static Connection getConnection() throws ClassNotFoundException,
SQLException {
        try{

            FileInputStream fis = null;
            fis = new FileInputStream("database.properties");
            props.load(fis);

            // load the Driver Class
            Class.forName(props.getProperty("DB_DRIVER_CLASS"));

            // create the connection now
            con =
DriverManager.getConnection(props.getProperty("DB_URL"),props.getProperty("DB_USERNAME"),
props.getProperty("DB_PASSWORD"));
        }
        catch(IOException e){
            e.printStackTrace();
        }
        return con;
    }
}

```

```
}

FlightManagementSystem.java
import java.util.*;
import java.sql.*;
public class FlightManagementSystem{
    public ArrayList<Flight> viewFlightsBySourceDestination(String source, String destination){
        DB db=new DB();
        ArrayList<Flight> list=new ArrayList<Flight>();
        try{
            int f=0;
            Connection con=db.getConnection();
            Statement st=con.createStatement();
            String sql= "select * from Flight where source= '"+source+"' and destination=
"+destination+"'";
            ResultSet rs=st.executeQuery(sql);
            while(rs.next()){
                f=1;
                Flight x=new Flight(rs.getInt(1), rs.getString(2),rs.getString(3), rs.getInt(4),
rs.getDouble(5));
                list.add(x);
            }
            con.close();
            if(f==1)
                return list;
            else
                return null;
        }
        catch(SQLException e){
            System.out.println("SQL Error. Contact Administrator.");
            return null;
        }
        catch(Exception e){
            System.out.println("Exception. Contact Administrator.");
            return null;
        }
    }
}
Flight.java
```

```
public class Flight {

    private int flightId;
    private String source;
```

```
private String destination;
private int noOfSeats;
private double flightFare;
public int getFlightId() {
    return flightId;
}
public void setFlightId(int flightId) {
    this.flightId = flightId;
}
public String getSource() {
    return source;
}
public void setSource(String source) {
    this.source = source;
}
public String getDestination() {
    return destination;
}
public void setDestination(String destination) {
    this.destination = destination;
}
public int getNoOfSeats() {
    return noOfSeats;
}
public void setNoOfSeats(int noOfSeats) {
    this.noOfSeats = noOfSeats;
}
public double getFlightFare() {
    return flightFare;
}
public void setFlightFare(double flightFare) {
    this.flightFare = flightFare;
}
public Flight(int flightId, String source, String destination,
             int noOfSeats, double flightFare) {
    super();
    this.flightId = flightId;
    this.source = source;
    this.destination = destination;
    this.noOfSeats = noOfSeats;
    this.flightFare = flightFare;
}
public String toString(){
```

```
        return ("Flight ID : "+getFlightId());
    }
```

```
}
```

### Perform Calculation

```
import java.util.Scanner;

public class Calculator {

    public static void main (String[] args) {

        Scanner sc=new Scanner(System.in);

        int a = sc.nextInt();

        int b= sc.nextInt();

        Calculate Perform_addition = performAddition();

        Calculate Perform_subtraction = performSubtraction();

        Calculate Perform_product = performProduct();

        Calculate Perform_division = performDivision();

        System.out.println("The sum is "+Perform_addition.performCalculation(a,b));

        System.out.println("The difference is
"+Perform_subtraction.performCalculation(a,b));

        System.out.println("The product is "+Perform_product.performCalculation(a,b));

        System.out.println("The division value is
"+Perform_division.performCalculation(a,b));

    }

    public static Calculate performAddition(){

        Calculate Perform_calculation = (int a,int b)->a+b;

        return Perform_calculation;
```

```
}

public static Calculate performSubtraction(){

    Calculate Perform_calculation = (int a,int b)->a-b;

    return Perform_calculation;

}

public static Calculate performProduct(){

    Calculate Perform_calculation = (int a,int b)->a*b;

    return Perform_calculation;

}

public static Calculate performDivision(){

    Calculate Perform_calculation = (int a,int b)->{

        float c = (float)a;

        float d = (float)b;

        return (c/d);

    };

    return Perform_calculation;

}

}

public interface Calculate {

    float performCalculation(int a,int b);

}
```

```

public class InPatient extends Patient {
    InPatient(String patientId, String patientName, long mobileNumber,
    String gender) {
        super(patientId, patientName, mobileNumber, gender);
    }
    InPatient() {
    }
    private double roomRent;
    public double getrent()
    {
        return roomRent;
    }
    public void setrent(double rent)
    {
        roomRent=rent;
    }
    public double calculateTotalBill(int no,double medi)
    {
        return (roomRent*no)+medi;
    }
}

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Scanner;

public class Main {
    public static void main(String [] args) throws IOException {
        Scanner sc=new Scanner(System.in);
        BufferedReader br=new BufferedReader(new
        InputStreamReader(System.in));
        OutPatient o1=new OutPatient();
        InPatient o2=new InPatient();
        System.out.println("1.In Patient\n2.Out Patient");
        System.out.println("Enter the choice");
        int a=sc.nextInt();
        //sc.hasNextLine();
        System.out.println("Enter the details\nPatient Id");
        String pid=br.readLine();
        System.out.println("Patient Name");
        String pname=br.readLine();
        System.out.println("Phone Number");
        long mob=sc.nextLong();
        System.out.println("Gender");
        String gen=br.readLine();
        if(a==1)
        {
            System.out.println("Room Rent");
            double rent=sc.nextDouble();
        }
    }
}

```

```

        System.out.println("Medicinal Bill");
        double med=sc.nextDouble();
        System.out.println("Number of Days of Stay");
        int no=sc.nextInt();
        c2.setrent(rent);
        System.out.println("Amount to be paid
"+c2.calculateTotalBill111(no,med));
    }
    else
    {
        System.out.println("Consultancy Fee");
        double con=sc.nextDouble();
        System.out.println("Medicinal Bill");
        double med=sc.nextDouble();
        System.out.println("Scan Pay");
        int scan=sc.nextInt();
        c1.setcon(con);
        System.out.println("Amount to be paid
"+c1.calculateTotalBill111(scan,med));
    }
}
}

public class OutPatient extends Patient {

    /*OutPatient(String patientId, String patientname, long
mobileNumber, String gender) {
    super(patientId, patientname, mobileNumber, gender);
    // TODO Auto-generated constructor stub
}*/
    OutPatient()
    {
        super();
    }
    private double consultingFee;
    public double getcon()
    {
        return consultingFee;
    }
    public void setcon(double con)
    {
        consultingFee=con;
    }
    public double calculateTotalBill111(int scan,double medi)
    {
        return (consultingFee+scan+medi);
    }
}

```

```
public class Patient {  
    private String patientId,patientname,gender;  
    private long mobileNumber;  
    Patient(String patientId, String patientname, long  
    mobileNumber, String gender)  
    {  
        this.patientId=patientId;  
        this.patientname=patientname;  
        this.gender=gender;  
        this.mobileNumber=mobileNumber;  
    }  
    Patient()  
    {  
    }  
    public String getpaid(){  
        return patientId;  
    }  
    public String getpaname(){  
        return patientname;  
    }  
    public String getpagen(){  
        return gender;  
    }  
    public long getpanob(){  
        return mobileNumber;  
    }  
  
    public void setpaid(String id){  
        patientId=id;  
    }  
    public void setpaname(String name){  
        patientname=name;  
    }  
    public void setpagen(String gen){  
        gender=gen;  
    }  
    public void setpanob(long mob){  
        mobileNumber=mob;  
    }  
}
```

### Payment Inheritance

#### **Bill.java**

```
public class Bill {  
  
    public String processPayment(Payment obj) {  
  
        String message = "Payment not done and your due amount is "+obj.getDueAmount();  
    }  
}
```

```

if(obj instanceof Cheque ) {

    Cheque cheque = (Cheque) obj;

    if(cheque.payAmount())

        message = "Payment done successfully via cheque";

}

else if(obj instanceof Cash ) {

    Cash cash = (Cash) obj;

    if(cash.payAmount())

        message = "Payment done successfully via cash";

}

else if(obj instanceof Credit ) {

    Credit card = (Credit) obj;

    if(card.payAmount())

        message = "Payment done successfully via creditcard. Remaining amount in your
"+card.getCardType()+" card is "+card.getCreditCardAmount();

}

return message;
}
}

```

### **Cash.java**

```

public class Cash extends Payment{

    private int cashAmount;

    public int getCashAmount() {

```

```
    return cashAmount;  
}  
  
public void setCashAmount(int cashAmount) {  
    this.cashAmount = cashAmount;  
}  
  
@Override  
  
public boolean payAmount() {  
    return getCashAmount() >= getDueAmount();  
}  
}
```

### Cheque.java

```
import java.text.ParseException;  
  
import java.text.SimpleDateFormat;  
  
import java.util.Calendar;  
  
import java.util.Date;  
  
import java.util.GregorianCalendar;  
  
public class Cheque extends Payment {  
  
    private String chequeNo;  
  
    private int chequeAmount;  
  
    private Date dateOfIssue;  
  
    public String getChequeNo() {  
        return chequeNo;  
    }
```

```
public void setChequeNo(String chequeNo) {
    this.chequeNo = chequeNo;
}

public int getChequeAmount() {
    return chequeAmount;
}

public void setChequeAmount(int chequeAmount) {
    this.chequeAmount = chequeAmount;
}

public Date getDateOfIssue() {
    return dateOfIssue;
}

public void setDateOfIssue(Date dateOfIssue) {

    this.dateOfIssue = dateOfIssue;
}

@Override

public boolean payAmount() {
    int months = findDifference(getDateOfIssue());
    return (getChequeAmount() >= getDueAmount() & months <= 6);
}

private int findDifference(Date date) {
    Calendar myDate = new GregorianCalendar();
    myDate.setTime(date);
    return (2020 - myDate.get(Calendar.YEAR)) * 12 + (0-myDate.get(Calendar.MONTH));
}

public void generateDate(String date) {

    try {
        Date issueDate = new SimpleDateFormat("dd-MM-yyyy").parse(date);
        setDateOfIssue(issueDate);
    }
    catch (ParseException e) {
        e.printStackTrace();
    }
}
```

```
    }  
}
```

### Credit.java

```
public class Credit extends Payment {  
  
    private int creditCardNo;  
  
    private String cardType;  
  
    private int creditCardAmount;  
  
    public int getCreditCardNo(){  
  
        return creditCardNo;  
  
    }  
  
    public void setCreditCardNo(int creditCardNo) {  
  
        this.creditCardNo = creditCardNo;  
  
    }  
  
    public String getCardType() {  
  
        return cardType;  
  
    }  
  
    public void setCardType(String cardType) {  
  
        this.cardType = cardType;  
  
    }  
  
    public int getCreditCardAmount() {  
  
        return creditCardAmount;  
  
    }  
  
    public void setCreditCardAmount(int creditCardAmount) {
```

```
this.creditCardAmount = creditCardAmount;

}

@Override

public boolean payAmount() {

int tax = 0;

boolean isDeducted = false;

switch(cardType) {

case "silver":

setCreditCardAmount(10000);

tax = (int) (0.02*getDueAmount())+getDueAmount();

if(tax <= getCreditCardAmount()) {

setCreditCardAmount(getCreditCardAmount()-tax);

isDeducted = true;

}

break;

case "gold":

setCreditCardAmount(50000);

tax = (int) (0.05*getDueAmount())+getDueAmount();

if(tax <= getCreditCardAmount()) {

setCreditCardAmount(getCreditCardAmount()-tax);

isDeducted = true;

}

}
```

```
break;

case "platinum":

setCreditCardAmount(100000);

tax = (int) (0.1*getDueAmount())+getDueAmount();

if(tax <= getCreditCardAmount()) {

setCreditCardAmount(getCreditCardAmount()-tax);

isDeducted = true;

}

break;

}

return isDeducted;

}

}
```

### **Main.java**

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Bill bill = new Bill();

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the due amount:");

        int dueAmount = sc.nextInt();

        System.out.println("Enter the mode of payment(cheque/cash/credit):");
```

```
String mode = sc.next();

switch (mode) {

    case "cash":

        System.out.println("Enter the cash amount:");

        int cashAmount = sc.nextInt();

        Cash cash = new Cash();

        cash.setCashAmount(cashAmount);

        cash.setDueAmount(dueAmount);

        System.out.println(bill.processPayment(cash));

        break;

    case "cheque":

        System.out.println("Enter the cheque number:");

        String number = sc.next();

        System.out.println("Enter the cheque amount:");

        int chequeAmount = sc.nextInt();

        System.out.println("Enter the date of issue:");

        String date = sc.next();

        Cheque cheque = new Cheque();

        cheque.setChequeAmount(chequeAmount);

        cheque.setChequeNo(number);

        cheque.generateDate(date);

        cheque.setDueAmount(dueAmount);
```

```
System.out.println(bill.processPayment(cheque));
break;

case "credit":

System.out.println("Enter the credit card number.");

int creditNumber = sc.nextInt();

System.out.println("Enter the card type(silver,gold,platinum)");

String cardType = sc.next();

Credit credit = new Credit();

credit.setCardType(cardType);

credit.setCreditCardNo(creditNumber);

credit.setDueAmount(dueAmount);

System.out.println(bill.processPayment(credit));

default:

break;

}

sc.close();

}

}
```

### **Payment.java**

```
public class Payment {

    private int dueAmount;

    public int getDueAmount() {

        return dueAmount;
    }
}
```

```

    }

    public void setDueAmount(int dueAmount) {

        this.dueAmount = dueAmount;

    }

    public boolean payAmount() {

        return false;

    }

}

```

## HUNGER EATS

```

package com.utility;
import java.util.*;
import com.bean.FoodProduct;
public class Order{
    private double discountPercentage;
    private List<FoodProduct> foodList=new ArrayList<FoodProduct>();

    public double getDiscountPercentage() {
        return discountPercentage;
    }
    public void setDiscountPercentage(double discountPercentage) {
        this.discountPercentage = discountPercentage;
    }
    public List<FoodProduct> getFoodList() {
        return foodList;
    }
    public void setFoodList(List<FoodProduct> foodList) {
        this.foodList = foodList;
    }

    public void findDiscount(String bankName)
    {
        if(bankName.equals("HDFC")) {
            discountPercentage=15.0;
        }
        else if(bankName.equals("ICICI")) {
    }
}

```

```

        discountPercentage=25.0;
    }
else if(bankName.equals("CUB")) {
    discountPercentage=30.0;
}
else if(bankName.equals("SBI")) {
    discountPercentage=50.0;
}
else if(bankName.equals("OTHERS")) {
    discountPercentage=0.0;
}

}

public void addToCart(FoodProduct foodProductObject)
{
List<FoodProduct> f=getFoodList();
f.add(foodProductObject);
setFoodList(f);

}

public double calculateTotalBill()
{
    double bill = 0;
    List<FoodProduct> f=getFoodList();
    for(int i=0;i<f.size();i++)
    {
//        System.out.println(f.get(i).getCostPerUnit());
//        System.out.println(f.get(i).getQuantity());
        bill+=f.get(i).getQuantity()*f.get(i).getCostPerUnit()*1.0;

    }
//    System.out.println(bill);
//    System.out.println(dis);
    bill=bill-((bill*discountPercentage)/100);
    return bill;
}

}

package com.ui;

import java.util.Scanner;

```

```
import com.utility.Order;
import com.bean.FoodProduct;

public class UserInterface{

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int itemno;
        String bank;

        System.out.println("Enter the number of items");
        itemno=sc.nextInt();
        System.out.println("Enter the item details");

        Order o=new Order();

        for(int i=0;i<itemno;i++)
        {
            FoodProduct fd=new FoodProduct();
            System.out.println("Enter the item id");
            fd.setFoodId(sc.nextInt());
            System.out.println("Enter the item name");
            fd.setFoodName(sc.next());
            System.out.println("Enter the cost per unit");
            fd.setCostPerUnit(sc.nextDouble());
            System.out.println("Enter the quantity");
            fd.setQuantity(sc.nextInt());
            o.addToCart(fd);

        }

        System.out.println("Enter the bank name to avail offer");
        bank=sc.next();
        o.findDiscount(bank);

        System.out.println("Calculated Bill Amount:"+o.calculateTotalBill());

    }
}
```

```
}

package com.bean;

public class FoodProduct {

    private int foodId;
    private String foodName;
    private double costPerUnit;
    private int quantity;

    public int getFoodId() {
        return foodId;
    }

    public void setFoodId(int foodId) {
        this.foodId = foodId;
    }

    public String getFoodName() {
        return foodName;
    }

    public void setFoodName(String foodName) {
        this.foodName = foodName;
    }

    public double getCostPerUnit() {
        return costPerUnit;
    }

    public void setCostPerUnit(double costPerUnit) {
        this.costPerUnit = costPerUnit;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

}
```

### Singapore

```
import java.util.*;

public class tourism {
    static String name;
    static String place;
    static int days;
    static int tickets;
```

```

static double price = 0.00;
static double total = 0.00;
public static void main(String[] args){
    Scanner in = new Scanner(System.in);
    System.out.println("Enter the passenger name");
    name = in.nextLine();
    System.out.println("Enter the place name");
    place=in.nextLine();
    if(place.equalsIgnoreCase("beach"))

||place.equalsIgnoreCase("pilgrimage")||place.equalsIgnoreCase("heritage")||place.equalsIgnoreCase("Hills")||place.equalsIgnoreCase("palls")||place.equalsIgnoreCase("adventure")){
        System.out.println("Enter the number of days");
        days = in.nextInt();
        if(days>0){
            System.out.println("Enter the number of Tickets");
            tickets = in.nextInt();
            if(tickets>0){
                if(place.equalsIgnoreCase("beach")){
                    price = tickets*270;
                    if(price>1000){
                        total = 85*price/100;
                        System.out.printf("Price:%.2f",total);
                    }
                    else {
                        System.out.printf("Price:%.2f",price);
                    }
                }
                else if(place.equalsIgnoreCase("prilgrimage")){
                    price = tickets*350;
                    if(price>1000){
                        total = 85*price/100;
                        System.out.printf("Price:%.2f",total);
                    }
                    else {
                        System.out.printf("Price:%.2f",price);
                    }
                }
                else if(place.equalsIgnoreCase("heritage")){
                    price = tickets*430;
                    if(price>1000){
                        total = 85*price/100;
                        System.out.printf("Price:%.2f",total);
                    }
                }
            }
        }
    }
}

```

```
else {
    System.out.printf("Price:%.2f",price);
}
}

else if(place.equalsIgnoreCase("hills")){
    price = tickets*780;
    if(price>1000){
        total = 85*price/100;
        System.out.printf("Price:%.2f",total);
    }
    else {
        System.out.printf("Price:%.2f",price);
    }
}

else if(place.equalsIgnoreCase("palls")){
    price = tickets*1200;
    if(price>1000){
        total = 85*price/100;
        System.out.printf("Price:%.2f",total);
    }
    else {
        System.out.printf("Price:%.2f",price);
    }
}

else {
    price = tickets*4500;
    if(price>1000){
        total = 85*price/100;
        System.out.printf("Price:%.2f",total);
    }
    else {
        System.out.printf("Price:%.2f",price);
    }
}

}

else{
    System.out.println(tickets+" is an Invalid no. of tickets");
}
}

else{
    System.out.println(days+" is an Invalid no. of days");
}
}
```

```

        else {
            System.out.println(place+" is an Invalid place");
        }
    }
}

```

### Prime no ending

```

import java.util.*;
public class Main
{
    public static void main (String[] args) {
        int flag=0, k=0, z=0;
        Scanner sc =new Scanner(System.in );
        System.out.println("Enter the first number");
        int f=sc.nextInt();
        System.out.println("Enter the last number");
        int l=sc.nextInt();
        for(int i=f; i<=l; i++)
        {
            for(int j=2; j<i; j++)// this loop increments flag if i is divisible by j
            {
                if(i%j==0)
                {
                    flag++;
                }
            }
            if(i==l && (flag!=0 || i%10!=1))//when last number is not a prime
            {
                while(z==0)
                {
                    for(int a=2; a<i; a++)
                    {
                        if(i%a==0)
                        {
                            flag++;
                        }
                    }
                    if(i%10==1 && flag==0)
                    {
                        System.out.print(", "+i);
                        z++;
                    }
                    flag=0;
                    i++;
                }
            }
        }
    }
}

```

```

        }
    }
    if(i%10==1 && flag==0)//to check for last digit 1 and prime
    {
        if(k==0)
        {
            System.out.print(i);
            k++;
        }
        else
        {
            System.out.print(", "+i);
        }
    }
    flag=0;
}
}

}
}

```

### Query Set

```

public class Query {

    private class DataSet{
        private String theatreId;
        private String theatreName;
        private String location;
        private int noOfScreen;
        private double ticketCost;
        public String getTheatreId() {
            return theatreId;
        }
        public void setTheatreId(String theatreId) {
            this.theatreId = theatreId;
        }
        public String getTheatreName() {
            return theatreName;
        }
        public void setTheatreName(String theatreName) {
            this.theatreName = theatreName;
        }
        public String getLocation() {
            return location;
        }
    }
}

```

```
public void setLocation(String location) {
this.location = location;
}
public int getNoOfScreen() {
return noOfScreen;
}
public void setNoOfScreen(int noOfScreen) {
this.noOfScreen = noOfScreen;
}
public double getTicketCost() {
return ticketCost;
}
public void setTicketCost(double ticketCost) {
this.ticketCost = ticketCost;
}
@Override
public String toString() {
return "Theatre id: " + theatreId + "\nTheatre name: " + theatreName + "\nLocation: " + location
+ "\nNo of Screen: " + noOfScreen + "\nTicket Cost: " + ticketCost+"\n";
}
}

private String queryId;
private String queryCategory;
private DataSet primaryDataset;
private DataSet secondaryDataSet;
public String getQueryId() {
return queryId;
}
public void setQueryId(String queryId) {
this.queryId = queryId;
}
public String getQueryCategory() {
return queryCategory;
}
public void setQueryCategory(String queryCategory) {
this.queryCategory = queryCategory;
}
public DataSet getPrimaryDataset() {
return primaryDataset;
}
public void setPrimaryDataset(DataSet primaryDataset) {
this.primaryDataset = primaryDataset;
}
```

```

public DataSet getSecondaryDataSet() {
    return secondaryDataSet;
}
public void setSecondaryDataSet(DataSet secondaryDataSet) {
    this.secondaryDataSet = secondaryDataSet;
}
@Override
public String toString() {
    return "Primary data set\n" + primaryDataset
        + "Secondary data set\n" + secondaryDataSet + "Query id: " + queryId + "\nQuery category=" +
        queryCategory;
}
}

}

```

```

import java.util.Scanner;
public class TestApplication {
    public static void main(String[] args) {
        Query query = new Query();
        Scanner sc = new Scanner(System.in);
        Query.DataSet primary = query.new DataSet();
        Query.DataSet secondary = query.new DataSet();
        System.out.println("Enter the Details of primary data set");
        System.out.println("Enter the theatre id");
        String theatreid = sc.nextLine();
        primary.setTheatreId(theatreid);
        sc.nextLine();
        System.out.println("Enter the theatre name");
        String theatrename = sc.nextLine();
        primary.setTheatreName(theatrename);
        sc.nextLine();
        System.out.println("Enter the location");
        String location = sc.nextLine();
        primary.setLocation(location);
        sc.nextLine();
        System.out.println("Enter the no of screens");
        int screens = sc.nextInt();
        primary.setNoOfScreen(screens);
        System.out.println("Enter the ticket cost");
        double cost = sc.nextDouble();
        primary.setTicketCost(cost);
    }
}

```

```

System.out.println("ENter the details of secondary data set");
System.out.println("Enter the theatre id");
theatreid = sc.nextInt();
secondary.setTheatreId(theatreid);
sc.nextLine();
System.out.println("Enter the theatre name");
theatrename = sc.next();
secondary.setTheatreName(theatrename);
sc.nextLine();
System.out.println("Enter the location");
location = sc.next();
secondary.setLocation(location);
sc.nextLine();
System.out.println("Enter the no of screens");
screens = sc.nextInt();
secondary.setNoOfScreen(screens);
System.out.println("Enter the ticket cost");
cost = sc.nextDouble();
secondary.setTicketCost(cost);
System.out.println("Enter the query id");
String queryid = sc.next();
query.setQueryId(queryid);
sc.nextLine();
System.out.println("Enter the query category");
String querycategory = sc.next();
query.setQueryCategory(querycategory);
sc.nextLine();
query.setPrimaryDataset(primary);
query.setSecondaryDataSet(secondary);

System.out.println(query);
}
}

```

### Extract book

```

import java.util.Scanner;

class ExtractBook {

    public static int extractDepartmentCode(String input) {
        return Integer.parseInt(input.substring(0, 3));
    }
}

```

```

public static String extractDepartmentName(int code) {

    switch (code) {
        case 101:
            return "Accounting";
        case 102:
            return "Economics";
        case 103:
            return "Engineering";
    }

    throw new Error(code + " is invalid department code");
}

public static int extractDate(String input) {
    String yearStr = input.substring(3, 7);
    try {
        int year = Integer.parseInt(yearStr);
        if (year > 2020 || year < 1900) {
            throw new NumberFormatException();
        }
        return year;
    } catch (NumberFormatException e) {
        throw new Error(yearStr + " is invalid year");
    }
}

public static int extractNumberOfPages(String input) {
    String pagesStr = input.substring(7, 12);
    try {
        int pages = Integer.parseInt(pagesStr);
        if (pages < 10) {
            throw new NumberFormatException();
        }
        return pages;
    } catch (NumberFormatException e) {
        throw new Error(pagesStr + " are invalid pages");
    }
}

public static String extractBookId(String input) {
    String id = input.substring(12, 18);
    if (!Character.isAlphabetic(id.charAt(0)))

```

```

        throw new NumberFormatException();
    try {
        Integer.parseInt(id.substring(1));
    } catch (NumberFormatException e) {
        throw new Error(id + " is invalid book id");
    }
    return id;
}

public static void parseAndPrint(String str) {
    if (str.length() != 18) {
        System.out.println(str + " is an invalid input");
        return;
    }

    try {
        int dCode = extractDepartmentCode(str);
        String dString = extractDepartmentName(dCode);
        int year = extractDate(str);
        int pages = extractNumberOfPages(str);
        String bookId = extractBookId(str);

        System.out.println("Department Code: " + dCode);
        System.out.println("Department Name: " + dString);
        System.out.println("Year of Publication: " + year);
        System.out.println("Number of Pages: " + pages);
        System.out.println("Book Id: " + bookId);

    } catch (Error e) {
        System.out.println(e.getMessage());
    }
}

public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    String input = sc.nextLine();
    parseAndPrint(input);
    sc.close();
}
}

```

```
import java.util.*;
class FDScheme {

    private int schemeNo;
    private double depositAmt;
    private int period;
    private float rate;
    public FDScheme(int schemeNo, double depositAmt, int period) {
        super();
        this.schemeNo = schemeNo;
        this.depositAmt = depositAmt;
        this.period = period;
        calculateInterestRate();
    }
    public int getSchemeNo() {
        return schemeNo;
    }
    public void setSchemeNo(int schemeNo) {
        this.schemeNo = schemeNo;
    }
    public double getDepositAmt() {
        return depositAmt;
    }
    public void setDepositAmt(double depositAmt) {
        this.depositAmt = depositAmt;
    }
    public int getPeriod() {
        return period;
    }
    public void setPeriod(int period) {
        this.period = period;
    }
    public float getRate() {
        return rate;
    }
    public void setRate(float rate) {
        this.rate = rate;
    }

    public void calculateInterestRate()
    {
        if(period>=1 && period<=90)
        {
            this.rate=(float) 5.5;
```

```

        }
        else if(period>=91 && period<=180)
        {
            this.rate=(float) 6.25;
        }
        else if(period>=181 && period<=365)
        {
            this.rate=(float) 7.5;
        }
        System.out.println("Interest rate for "+period+" days is "+this.rate);
    }
}
public class Main{

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter Scheme no");
        int no=sc.nextInt();
        sc.nextLine();
        System.out.println("Enter Deposit amount");
        double amt=sc.nextDouble();
        System.out.println("enter period of deposit");
        int prd=sc.nextInt();
        FDScheme obj=new FDScheme(no,amt,prd);
    }
}

```

### Annual Salary

```

import java.io.*;
public class Main
{
    public static void main(String[] args)throws IOException
    {
        // Scanner sc=new Scanner(System.in);
        //Fill the code
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter the Employee Name");
        String name=br.readLine();
        System.out.println("Enter percentage of salary");
        double percent=Double.parseDouble(br.readLine());
        if(percent>0&&percent<20)
        {

```

```

System.out.println("Enter the Year of Experience");
int time=Integer.parseInt(br.readLine());

if(time>0&&time<15)
{
    double permonth=12000+(2000*(time));
    double dayshift=permonth*6;
    double nightshift=((permonth*percent)/100)+permonth)*6;
    double annualIncome=dayshift+nightshift;

    String str="The annual salary of "+name+" is";
    System.out.println(str+" "+annualIncome);

}
else{
    System.out.println((int)time+" is an invalid year of experience");
}

}
else
System.out.println((int)percent+" is an invalid percentage");
}
}

```

### Amity Passenger

```

import java.util.*;
public class PassengerAmenity {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);
System.out.println("Enter the number of passengers");
int no=sc.nextInt();
sc.nextLine();
int count=0;

if(no>0)
{
String name[]=new String[no];
String seat[]=new String[no];
String arr[]=new String[no];

for(int i=0;i<no;i++)

```

```
{  
    System.out.println("Enter the name of the passenger "+(i+1));  
    String str=sc.nextLine();  
  
    name[i]=str.toUpperCase();  
  
    System.out.println("Enter the seat details of the passenger "+(i+1));  
    seat[i]=sc.nextLine();  
  
    if(seat[i].charAt(0)>='A' && seat[i].charAt(0)<='S')  
    {  
  
        int r=Integer.parseInt(seat[i].substring(1,seat[i].length()));  
  
        if(r>=10 && r<=99)  
        {  
            count++;  
        }  
  
        else  
        {  
            System.out.println(r+" is invalid seat number");  
            break;  
        }  
    }  
  
    else  
    {  
        System.out.println(seat[i].charAt(0)+" is invalid coach");  
        break;  
    }  
  
    arr[i]=name[i]+" "+seat[i];  
}  
  
if(count==seat.length)  
{  
  
    Arrays.sort(seat);  
  
    for(int i=seat.length-1;i>=0;i--)  
    {  
        for(int j=0;j<arr.length;j++)  
        {
```

```

        if(arr[j].contains(seat[i]))
        {
            System.out.println(arr[j]);
        }
    }

}
}

else
{
    System.out.println(no+" is invalid input");
}
}

}

```

### Change the Case

```

import java.util.*;

public class ChangeTheCase {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String a = sc.next();
        if(a.length() < 3) {
            System.out.println("String length of " + a + " is too short");
            return;
        }
        else if(a.length() > 10) {
            System.out.println("String length of " + a + " is too long");
            return;
        }

        char[] arr = a.toCharArray();
        char[] arr1 = new char[arr.length];
        int j = 0;
        for(int i = 0; i < a.length(); i++) {
            if((arr[i]<65 || ((arr[i]>90) && (arr[i]<97)) || arr[i]>122)) {
                arr1[j++] = arr[i];
            }
        }
    }
}
```

```
}

if(j!=0) {
    System.out.print("String should not contain ");
    for(int i = 0; i<=j; i++) {
        System.out.print(arr1[i]);
    }
    return;
}

char b = sc.next().charAt(0);
int present = 0;
for(int i = 0; i<a.length(); i++) {
    if(arr[i] == Character.toUpperCase(b)) {
        arr[i] = Character.toLowerCase(b);
        present = 1;
    }
    else if(arr[i] == Character.toLowerCase(b)) {
        arr[i] = Character.toUpperCase(b);
        present = 1;
    }
}

if(present == 0) {
    System.out.println("Character " + b + " is not found");
}
else {
    for(int i = 0; i <a.length(); i++) {
        System.out.print(arr[i]);
    }
}
}
```

## **Club Member**

```
import java.util.Scanner;  
  
public class ClubMember {  
    private int memberId;  
    private String memberName;  
    private String memberType;  
    private double membershipFees;
```

```
public ClubMember(int memberId, String memberName, String memberType) {
```

```
super();
this.memberId = memberId;
this.memberName = memberName;
this.memberType = memberType;
calculateMembershipFees();
}
public int getMemberId() {
    return memberId;
}
public void setMemberId(int memberId) {
    this.memberId = memberId;
}
public String getMemberName() {
    return memberName;
}
public void setMemberName(String memberName) {
    this.memberName = memberName;
}
public String getMemberType() {
    return memberType;
}
public void setMemberType(String memberType) {
    this.memberType = memberType;
}
public double getMembershipFees() {
    return membershipFees;
}
public void setMembershipFees(double membershipFees) {
    this.membershipFees = membershipFees;
}

public void calculateMembershipFees() {
    if(!(memberType == "Gold"))
    {
        this.membershipFees=(double) 50000.0;
    }
    else if(!(memberType=="Premium"))
    {
        this.membershipFees=(double) 75000.0;
    }
    System.out.println("Member Id is "+this.memberId);
    System.out.println("Member Name is "+this.memberName);
    System.out.println("Member Type is "+this.memberType);
```

```
System.out.println("Membership Fees is "+this.membershipFees);

}

}

public class Main {
public static void main(String[] args) {
Scanner sc=new Scanner(System.in);
System.out.println("Enter Member Id");
int id=sc.nextInt();
sc.nextLine();
System.out.println("Enter Name");
String name=sc.next();
System.out.println("Enter Member Type");
String type=sc.next();
ClubMember club=new ClubMember(id, name, type);
//club.calculateMembershipFees();
}
}
```

## **CostAndTimeEstimation**

```
package com.cts.conctes.client;

import com.cts.conctes.exception.ConstructionEstimationException;

import com.cts.conctes.service.ConstructionProjectEstimationService;

public class CostAndTimeEstimation {

    public static void main(String[] args) throws ConstructionEstimationException

    {

        ConstructionProjectEstimationService cpeService = new

        ConstructionProjectEstimationService();

        boolean isTrue=cpeService.addConstructionProjectDetails("inputfeed.txt");

        if(isTrue) {

            System.out.println("All are added successfully into the database");

        }

    }

}
```

## **CostAndTimeEstDAO**

```
package com.cts.conctes.dao;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.Date;

import com.cts.conctes.exception.ConstructionEstimationException;
```

```
import com.cts.conctes.model.ConstructionProject;
import com.cts.conctes.util.ApplicationUtil;
public class CostAndTimeEstDAO {
    public static Connection connection = null;
    public boolean insertConstructionProject(ArrayList <ConstructionProject> constProjects)
        throws ConstructionEstimationException {
        boolean recordsAdded = false;
        int index=0;
        int size=constProjects.size();
        connection=DBConnectionManager.getInstance().getConnection();
        String query="insert into constructionproject values(?, ?, ?, ?, ?, ?, ?);";
        for(ConstructionProject cp: constProjects) {
            try {
                PreparedStatement ps=connection.prepareStatement(query);
                ps.setString(1, cp.getProjectId());
                ps.setDate(2, ApplicationUtil.utilToSqlDateConverter(cp.getPlannedDOStart()));
                ps.setString(3, cp.getTypeOfProject());
                ps.setString(4, cp.getStructure());
                ps.setDouble(5, cp.getAreaInSqFt());
                ps.setDouble(6, cp.getEstimatedCostInlac());
                ps.setDouble(7, cp.getEstimatedTimeInMonths());
                int row=ps.executeUpdate();
                if(row>0)
                    index+=1;
            }
        }
    }
}
```

```
        if(index==size) {

            recordsAdded=true;

        }

    } catch (SQLException e) {

        // TODO Auto-generated catch block

        e.printStackTrace();

    }

}

return recordsAdded;

}

public ArrayList <ConstructionProject> getConstructionProjectsData() throws
ConstructionEstimationException

{

ArrayList <ConstructionProject> consApplicants = new
ArrayList<ConstructionProject>();

connection=DBConnectionManager.getInstance().getConnection();

String query="select * from constructionproject";

try {

    PreparedStatement ps=connection.prepareStatement(query);

    ResultSet rs=ps.executeQuery();

    while(rs.next()) {

        String id=rs.getString(1);

        Date d=rs.getDate(2);

        String typeProject=rs.getString(3);

        String structure=rs.getString(4);

        double areaSqFt=rs.getDouble(5);

    }

}
```

```

        double costInLac=rs.getDouble(6);

        double timeInMonths=rs.getDouble(7);

        consApplicants.add(new ConstructionProject(id, d, typeProject, structure, areaSqFt,
costInLac, timeInMonths));

    }

} catch (SQLException e) {

    // TODO Auto-generated catch block

    e.printStackTrace();

}

return consApplicants;

}
}

```

## **DBConnectionManager**

```

package com.cts.conctes.dao;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileReader;

import java.io.IOException;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.util.Properties;

import com.cts.conctes.exception.ConstructionEstimationException;

```

```
public class DBConnectionManager {  
    private static Connection con = null;  
    private static DBConnectionManager instance;  
    private DBConnectionManager() throws ConstructionEstimationException  
    {  
        //WRITE YOUR CODE HERE  
        //return con;  
    }  
  
    public static DBConnectionManager getInstance() throws  
    ConstructionEstimationException  
    {  
        if(instance==null) {  
            instance=new DBConnectionManager();  
        }  
        return instance;  
    }  
  
    public Connection getConnection()  
    {  
        try {  
            Class.forName("com.mysql.cj.jdbc.Driver");  
        } catch (ClassNotFoundException e) {  
            // TODO Auto-generated catch block  
            e.printStackTrace();  
        }  
    }  
}
```

```
Properties properties=new Properties();

FileReader f=null;

try {

    f = new FileReader("database.properties");

} catch (FileNotFoundException e2) {

    // TODO Auto-generated catch block

    e2.printStackTrace();

}

try {

    properties.load(f);

} catch (IOException e1) {

    // TODO Auto-generated catch block

    e1.printStackTrace();

}

String url=properties.getProperty("url");

String user=properties.getProperty("username");

String password=properties.getProperty("password");

try {

    con=DriverManager.getConnection(url, user, password);

} catch (SQLException e) {

    // TODO Auto-generated catch block

    e.printStackTrace();

}

return con;
```

```
}
```

```
}
```

## **ConstructionEstimationException**

```
package com.cts.conctes.exception;

public class ConstructionEstimationException extends Exception{

    String strMsg1;

    Throwable strMsg2;

    public ConstructionEstimationException() {

        super();

    }

}
```

## **ConstructionProject**

```
package com.cts.conctes.model;

import java.util.Date;

public class ConstructionProject {

    String projectId;

    Date plannedDOStart;

    String typeOfProject;

    String structure;

    double areaInSqFt;

    double estimatedCostInlac;

    double estimatedTimeInMonths;

    public ConstructionProject() {

        super();

    }
```

```
}

public ConstructionProject(String projectId, Date plannedDOStart, String typeOfProject, String
structure, double areaInSqFt, double estimatedCostInLac, double estimatedTimeInMonths) {

    super();

    this.projectId = projectId;

    this.plannedDOStart = plannedDOStart;

    this.typeOfProject = typeOfProject;

    this.structure = structure;

    this.areaInSqFt = areaInSqFt;

    this.estimatedCostInLac = estimatedCostInLac;

    this.estimatedTimeInMonths = estimatedTimeInMonths;

}

public String getProjectId() {

    return projectId;

}

public void setProjectId(String projectId) {

    this.projectId = projectId;

}

public Date getPlannedDOStart() {

    return plannedDOStart;

}

public void setPlannedDOStart(Date plannedDOStart) {

    this.plannedDOStart = plannedDOStart;

}
```

```
public String getTypeOfProject() {  
    return typeOfProject;  
}  
  
public void setTypeOfProject(String typeOfProject) {  
    this.typeOfProject = typeOfProject;  
}  
  
public String getStructure() {  
    return structure;  
}  
  
public void setStructure(String structure) {  
    this.structure = structure;  
}  
  
public double getAreaInSqFt() {  
    return areaInSqFt;  
}  
  
public void setAreaInSqFt(double areaInSqFt) {  
    this.areaInSqFt = areaInSqFt;  
}  
  
public double getEstimatedCostInLac() {  
    return estimatedCostInLac;  
}  
  
public void setEstimatedCostInLac(double estimatedCostInLac) {  
    this.estimatedCostInLac = estimatedCostInLac;  
}
```

```

public double getEstimatedTimeInMonths() {
    return estimatedTimeInMonths;
}

public void setEstimatedTimeInMonths(double estimatedTimeInMonths)
{this.estimatedTimeInMonths = estimatedTimeInMonths;
}

@Override

public String toString() {
    return "ConstructionProject [projectId=" + projectId + ", plannedDOStart=" +
plannedDOStart + ", typeOfProject="
+ typeOfProject + ", structure=" + structure + ", areaInSqFt=" +
areaInSqFt + ", estimatedCostInLac="
+ estimatedCostInLac + ", estimatedTimeInMonths=" +
estimatedTimeInMonths + "]";
}

}

```

## **ConstructionProjectEstimationService**

```

package com.cts.conctes.service;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;

import com.cts.conctes.dao.CostAndTimeEstDAO;
import com.cts.conctes.exception.ConstructionEstimationException;
import com.cts.conctes.model.ConstructionProject;

```

```

import com.cts.conctes.util.ApplicationUtil;

public class ConstructionProjectEstimationService {

    public static ArrayList <ConstructionProject> buildConstructionProjectList(List <String>
consProjectRecords) {

        final String COMMADELIMITER = ",";
        ArrayList <ConstructionProject> consProjectRecordList = new
                ArrayList<ConstructionProject>();

        for(String s: consProjectRecords) {

            String[] s1=s.split(COMMADELIMITER);

            String id=s1[0];
            Date d=ApplicationUtil.stringToDateConverter(s1[1]);
            String typeproject=s1[2];
            String structure=s1[3];
            double areaInSqFt=Double.parseDouble(s1[4]);
            double costs[]=estimateTimeAndCostForConstruction(typeproject, structure,
areaInSqFt);
            double estimatedCostInLac=costs[0];
            double estimatedTimeInMon=costs[1];
            ConstructionProject p=new ConstructionProject(id, d, typeproject, structure,
areaInSqFt, estimatedCostInLac, estimatedTimeInMon);
            consProjectRecordList.add(p);
        }
        return consProjectRecordList;
    }

    public boolean addConstructionProjectDetails(String inputFeed) throws

```

```

ConstructionEstimationException {
    ArrayList<ConstructionProject>
p=buildConstructionProjectList(ApplicationUtil.readFile(inputFeed));

    CostAndTimeEstDAO obj=new CostAndTimeEstDAO();

    if(obj.insertConstructionProject(p)) {

        ArrayList<ConstructionProject> p1=obj.getConstructionProjectsData();

        for(ConstructionProject cp: p1) {

            System.out.println(cp);

        }

        return true;

    }

    return false;

}

public static double[] estimateTimeAndCostForConstruction(String projectType,String
structure,double areaInSqFt)

{
    double costEstimateInRs=0.0,timeEstimateInMonths=0.0;

    double costs[] = {costEstimateInRs,timeEstimateInMonths};

/*
 * The Cost Estimate and
 *

```

Based on the type of the Project & the Structure , according to the required area of Construction, the cost & time have to be calculated based on the base data available in the table provided in the use case document:

For eg. If the Project Type is “Commercial” and the structure is “Shopping Complex” the cost incurred for the construction of per sq. ft is Rs.2600 and the time taken for the construction of the 1000 sq ft of the same project is 0.23 Months, calculation has to be performed on the similar basis i.e Pro rata basis depending upon the type and the area of construction.

\*/

```
if(projectType.equals("Commercial")) {  
    if(structure.equals("Shopping Complex")) {  
        costs[0]=2600*areaInSqFt;  
        costs[1]=0.23*areaInSqFt/1000;  
    }  
    else if(structure.equals("ResApartments")) {  
        costs[0]=2750*areaInSqFt;  
        costs[1]=0.24*areaInSqFt/1000;  
    }  
    else {  
        costs[0]=2600*areaInSqFt;  
        costs[1]=0.2*areaInSqFt/1000;  
    }  
}  
else if(projectType.equals("Infrastructural")) {  
    if(structure.equals("Bridge")) {  
        costs[0]=10000*areaInSqFt;
```

```
    costs[1]=0.25*areaInSqFt/1000;  
}  
  
else if(structure.equals("FlyOver")) {  
  
    costs[0]=14000*areaInSqFt;  
  
    costs[1]=0.22*areaInSqFt/1000;  
  
}  
  
else {  
  
    costs[0]=8000*areaInSqFt;  
  
    costs[1]=(0.25/1000)*areaInSqFt;  
  
}  
  
}  
  
else {  
  
    if(structure.equals("House")) {  
  
        costs[0]=2250*areaInSqFt;  
  
        costs[1]=0.26*areaInSqFt/1000;  
  
    }  
  
    else if(structure.equals("Apartments")) {  
  
        costs[0]=2500*areaInSqFt;  
  
        costs[1]=0.24*areaInSqFt/1000;  
  
    }  
  
    else {  
  
        costs[0]=2750*areaInSqFt;  
  
        costs[1]=0.23*areaInSqFt/1000;  
  
    }  
}
```

```
    }

    return costs;
}

}
```

## **ApplicationUtil**

```
package com.cts.conctes.util;

import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.StringTokenizer;
import com.cts.conctes.exception.ConstructionEstimationException;

public class ApplicationUtil {

    public static List<String> readFile(String inputfeed) throws
ConstructionEstimationException {
        List<String> constructionProjects = new ArrayList<String>();
        BufferedReader br=null;
```

```
try {  
    br = new BufferedReader(new FileReader(inputfeed));  
  
} catch (FileNotFoundException e1) {  
    // TODO Auto-generated catch block  
    e1.printStackTrace();  
}  
  
String s=null;  
try {  
    s = br.readLine();  
    while(s!=null){  
        String s1[]=s.split(",");  
        if(checkIfCurrentFinYearProject(stringToDateConverter(s1[1])))  
            constructionProjects.add(s);  
        s=br.readLine();  
    }  
} catch (IOException e) {  
    e.printStackTrace();  
}  
return constructionProjects;  
}  
  
public static java.sql.Date utilToSqlDateConverter(java.util.Date utDate) {  
    java.sql.Date sqlDate;  
    String s=new SimpleDateFormat("yyyy-MM-dd").format(utDate);
```

```
sqlDate=java.sql.Date.valueOf(s);

return sqlDate;

}

public static java.util.Date stringToDateConverter(String stringDate) {

    Date strDate = null;

    SimpleDateFormat sdf=new SimpleDateFormat("yyyy-MM-dd");

    try {

        strDate=sdf.parse(stringDate);

    } catch (ParseException e) {

        e.printStackTrace();

    }

    return strDate;

}

public static boolean checkIfCurrentFinYearProject(Date dos)

{

    boolean flag = false;

    int givenYear,givenMonth;

    givenYear = (dos.getYear()+1900);

    givenMonth = dos.getMonth();

    Date curDate = new Date();

    int curYear,curMonth;

    curYear = (curDate.getYear()+1900);

    curMonth = curDate.getMonth();

    if( curYear == givenYear)
```

```

{

    if(((curMonth >=0)&&(curMonth <= 2)) && ((givenMonth
        >=0)&&(givenMonth <= 2)))

    {

        flag = true;

    }

    else if(((curMonth >=3)&&(curMonth <= 11)) && ((givenMonth
        >=3)&&(givenMonth <= 11)))

    {

        flag = true;

    }

    else

    {

        flag = false;

    }

}

else if(curYear > givenYear)

{

    int dif = curYear - givenYear;

    if(dif == 1)

    {

        if(((curMonth >=0)&&(curMonth <= 2)) && ((givenMonth
            >=3)&&(givenMonth <= 11)))

    {

```

```
    flag = true;

}

else if(((curMonth >=3)&&(curMonth <= 11)) && ((givenMonth
>=3)&&(givenMonth <= 11)))

{

    flag = false;

}

else{

    flag = false;

}

}

else

{

    flag = false;

}

}

else if(curYear < givenYear)

{

    int dif = givenYear-curYear;

    if(dif == 1)

    {

        if(((curMonth >=3)&&(curMonth <= 11)) && ((givenMonth
>=0)&&(givenMonth <= 2)))

    {
```

```
        flag = true;

    }

else if(((curMonth >=3)&&(curMonth <= 11)) && ((givenMonth
        >=3)&&(givenMonth <= 11)))

{

    flag = false;

}

else

{

    flag = false;

}

}

else

{

    flag = false;

}

}

else

{

    flag = false;

}

}

return flag;

}

}
```

## CREDIT CARD ADMIN SYSTEM

### CreditcardDAO.java :

```
package com.cts.creditcard.dao;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import com.cts.creditcard.util.*;
import java.util.List;
import com.cts.creditcard.exception.CreditCardAdminSystemException;
import com.cts.creditcard.vo.CreditCard;
public class CreditCardDAO {

    public Boolean addCreditCardDetails(List<CreditCard> cards) throws
        CreditCardAdminSystemException {
        Connection conn =
DBConnectionManager.getInstance().getConnection();
        String query = "insert into creditcard values(?, ?, ?, ?, ?, ?, ?, ?);";
        PreparedStatement ps = null;
        boolean enterd = false;
        int index = 0;
        int size = cards.size();
        try {
            for (CreditCard cc : cards) {
                ps = conn.prepareStatement(query);
                ps.setLong(1, cc.creditCardNum);
                ps.setString(2, cc.customerName);
                ps.setString(3, cc.customerEmail);
                ps.setLong(4, cc.customerPhone);
                ps.setDouble(5, cc.billAmount);
                ps.setDate(6,
ApplicationUtil.convertUtilToSqlDate(cc.dueDate));
                if (cc.paymentDate == null) {
```

```

        ps.setDate(7, null);
    } else {
        ps.setDate(7,
ApplicationUtil.convertUtilToSqlDate(cc.paymentDate));
    }
    int row = ps.executeUpdate();
    if (row > 0) {
        index += 1;
    }
    if (index == size) {
        enterd = true;
    }
}
} catch (SQLException e) {
    e.printStackTrace();
}

return enterd;
}

}

```

---

### CreditCardAdminSystemException.java:

```

package com.cts.creditcard.exception;
public class CreditCardAdminSystemException extends Exception {
    private static final long serialVersionUID = -6349759544203601561L;
//TODO add your constructors here
}

```

---

## **MainApp.java:**

```
package com.cts.creditcard.main;

import com.cts.creditcard.exception.CreditCardAdminSystemException;
import com.cts.creditcard.service.CreditCardAdminService;
import com.cts.creditcard.util.ApplicationUtil;

public class MainApp {

    public static void main(String[] args) {
        CreditCardAdminService obj=new CreditCardAdminService();
        try {
            if(obj.addCreditCardDetails("inputfeed.txt")) {
                System.out.println("Elements added Successfully");
            }
        } catch (CreditCardAdminSystemException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

---

## **CreditCardAdminService.java:**

```
package com.cts.creditcard.service;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

import com.cts.creditcard.dao.CreditCardDAO;
```

```

import com.cts.creditcard.exception.CreditCardAdminSystemException;
import com.cts.creditcard.util.ApplicationUtil;
import com.cts.creditcard.vo.CreditCard;
public class CreditCardAdminService {

    /**
     * @param records
     * @return List<Customer>
     */
    public static List<CreditCard>
buildMasterCreditCardList(List<String> records) {
        List<CreditCard> l=new ArrayList<>();
        for(String s:records) {
            String s1[]=s.split(",");
            long id=Long.parseLong(s1[0]);
            String name=s1[1];
            String mail=s1[2];
            long phone=Long.parseLong(s1[3]);
            double amount=Double.parseDouble(s1[4]);
            Date dueDate=ApplicationUtil.convertStringToDate(s1[5]);
            Date paidDate=null;
            if(s1.length==7) {
                paidDate=ApplicationUtil.convertStringToDate(s1[6]);
                if(dueDate.compareTo(paidDate)<0)
                    amount=getBillAmountWithLatePaymentCharges(amount);
            }
            CreditCard c=new CreditCard(id, name, mail, phone, amount,
dueDate, paidDate);
            l.add(c);
            System.out.println(id+ name+ amount);
        }
        return l;
    }
    /**
     * @param billAmount
     * @return Double

```

```

*/
public static Double getBillAmountWithLatePaymentCharges(Double
billAmount) {
    return billAmount+500;
}/**
 * @param inputFeed
 * @return Boolean
 * @throws CreditCardAdminSystemException
*/
public Boolean addCreditCardDetails(String inputFeed) throws
    CreditCardAdminSystemException {
    List<CreditCard>
cd=buildMasterCreditCardList(ApplicationUtil.readFile(inputFeed));
    CreditCardDAO obj1=new CreditCardDAO();
    if(obj1.addCreditCardDetails(cd)) {
        return true;
    }
    return false; //TODO change this return value
}
}

```

---

### **ApplicationUtil.java:**

```

import java.io.BufferedReader;
import java.util.*;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;
import com.cts.creditcard.exception.CreditCardAdminSystemException;

```

```
public class ApplicationUtil {  
    /**  
     * @param fileName  
     * @return List<String>  
     * @throws CreditCardAdminSystemException  
     */  
    public static List<String> readFile(String fileName) throws  
        CreditCardAdminSystemException {  
        List<String> l=new ArrayList<>();  
        BufferedReader br=null;  
        try {  
            br=new BufferedReader(new FileReader(fileName));  
        }  
        catch (FileNotFoundException e) {  
            e.printStackTrace();  
        }  
        String s=null;  
        try {  
            s=br.readLine();  
            while(s!=null) {  
                String s1[]=s.split(",");  
                if(s1[0].startsWith("5")) {  
                    l.add(s);  
                }  
                s=br.readLine();  
            }  
        }  
        catch(IOException e) {  
        }  
        return l;  
    }  
    public static Date geDateWithoutTime(Date date) {  
  
        return null; // TODO change this return value  
    }  
}
```

```

/**
 * @param uDate* Date
 * @return sql Date
 */
public static java.sql.Date convertUtilToSqlDate(java.util.Date uDate)
{
    java.sql.Date sqlDate=null;
    String s=(new SimpleDateFormat("yyyy-MM-dd")).format(uDate);
    sqlDate=java.sql.Date.valueOf(s);
    return sqlDate;
}
/**
 * @param inDate
 * @return Date
 */
public static Date convertStringToDate(String inDate) {
    Date date=null;
    try {
        date=(new SimpleDateFormat("yyyy-MM-dd")).parse(inDate);
    } catch (ParseException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    return date; // TODO change this return value
}
}

```

---

### **DBConnectionManager.java:**

```

package com.cts.creditcard.util;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

```

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.Properties;

import com.cts.creditcard.exception.CreditCardAdminSystemException;

public class DBConnectionManager {
    private static Connection con = null;
    private static DBConnectionManager instance;
    private DBConnectionManager() throws
CreditCardAdminSystemException {

}

public static DBConnectionManager getInstance() throws
    CreditCardAdminSystemException {
    if(instance==null) {
        instance=new DBConnectionManager();
    }
    return instance; // TODO change this return value
}
public Connection getConnection()
{
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");
    } catch (ClassNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    Properties properties=new Properties();
    FileReader f=null;
    try {
        f = new FileReader("database.properties");
    } catch (FileNotFoundException e2) {
```

```

    // TODO Auto-generated catch block
    e2.printStackTrace();
}
try {
    properties.load(f);
} catch (IOException e1) {
    // TODO Auto-generated catch block
    e1.printStackTrace();
}
String url=properties.getProperty("url");
String user=properties.getProperty("username");
String password=properties.getProperty("password");
try {
    con=DriverManager.getConnection(url, user, password);
} catch (SQLException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
return con;
}

}

```

---

### CreditCard.java:

```

package com.cts.creditcard.vo;
import java.util.Date;
public class CreditCard {
    //DO NOT CHANGE THE NAMES OR DATA TYPES OR VISIBILITY
    //OF THE BELOW MEMBER VARIABLES
    public Long creditCardNum;
    public String customerName;
    public String customerEmail;

```

```
public Long customerPhone;
public Double billAmount;
public Date dueDate;
public Date paymentDate;
public CreditCard(long cnum, String cname, String cemail, long
cphone, double bamount, Date ddate, Date pdate) {
    this.creditCardNum=cnum;
    this.customerName=cname;
    this.customerEmail=cemail;
    this.customerPhone=cphone;
    this.billAmount=bamount;
    this.dueDate=ddate;
    this.paymentDate=pdate;
}
}
```

---

## TICKET RESERVATION

INVALID CARRIER-

```
public class InvalidCarrierException extends Exception{
    //FILL THE CODE HERE
    public InvalidCarrierException (String message){
        super(message);
    }
}
```

PASSENGER.JAVA-

```
//DO NOT EDIT OR ADD ANY CODE
public class Passenger {

    private String passengerName;
    private long phoneNumber;
    private String emailId;
    private String carrierName;
    private String dateOfJourney;
    private String source;
    private String destination;

    public Passenger() {
        super();
        // TODO Auto-generated constructor stub
    }

    public Passenger(String passengerName, long phoneNumber, String emailId,
String carrierName, String dateOfJourney,
                    String source, String destination) {
        super();
        this.passengerName = passengerName;
        this.phoneNumber = phoneNumber;
        this.emailId = emailId;
        this.carrierName = carrierName;
        this.dateOfJourney = dateOfJourney;
        this.source = source;
        this.destination = destination;
    }

    public String getPassengerName() {
        return passengerName;
    }
    public void setPassengerName(String passengerName) {
        this.passengerName = passengerName;
    }

    public long getPhoneNumber() {
        return phoneNumber;
    }
    public void setPhoneNumber(long phoneNumber) {
        this.phoneNumber = phoneNumber;
    }
    public String getEmailId() {
        return emailId;
    }
}
```

```

    }
    public void setEmailId(String emailId) {
        this.emailId = emailId;
    }
    public String getCarrierName() {
        return carrierName;
    }
    public void setCarrierName(String carrierName) {
        this.carrierName = carrierName;
    }
    public String getDateOfJourney() {
        return dateOfJourney;
    }
    public void setDateOfJourney(String dateOfJourney) {
        this.dateOfJourney = dateOfJourney;
    }
    public String getSource() {
        return source;
    }
    public void setSource(String source) {
        this.source = source;
    }
    public String getDestination() {
        return destination;
    }
    public void setDestination(String destination) {
        this.destination = destination;
    }
}

```

#### PASSENGER CATERGORY-

```

import java.util.List;
@FunctionalInterface
public interface PassengerCategorization {
    abstract public List<Passenger> retrievePassenger_BySource(List<Passenger>
passengerRecord, String source);
}

}

```

#### PASSENGER UTILITY-

```

import java.util.List;
import java.io.*;
import java.util.*;

public class PassengerUtility {

    public List<Passenger> fetchPassenger(String filePath) throws Exception{

        //FILL THE CODE HERE
        List<Passenger> list = new ArrayList<Passenger>();
        String line = "";
        String splitBy = ",";

```

```

        BufferedReader br = new BufferedReader(new FileReader(filePath));
        while((line=br.readLine())!=null){
            String[] p = line.split(splitBy);
            Passenger passenger = new
Passenger(p[0],Long.parseLong(p[1]),p[2],p[3],p[4],p[5],p[6]);
            if(isValidCarrierName(passenger.getCarrierName())){
                list.add(passenger);
            }
        }

        return list;
    }

    public boolean isValidCarrierName (String carrierName)
{
    //FILL THE CODE HERE
    String temp = carrierName;
    if((temp.toLowerCase()).equals("bella")){
        return true;
    }else{
        try{
            throw new InvalidCarrierException(carrierName+" is an Invalid carrier
name.");
        }
        catch(InvalidCarrierException e){
            System.out.println(e.getMessage());
        }
    }
    return false;
}

}

```

#### SKELETON VALIDATION-

```

import java.lang.reflect.Method;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.stream.Stream;

/**
 * @author TJ
 *
 * This class is used to verify if the Code Skeleton is intact and not modified by
participants thereby ensuring smooth auto evaluation
*/
public class SkeletonValidator {

    public SkeletonValidator() {

        validateClassName("PassengerCategorization");
        validateClassName("Passenger");
        validateClassName("InvalidCarrierException");
    }
}

```

```

validateClassName("PassengerUtility");

validateMethodSignature(
    "retrievePassenger_BySource:java.util.List",
    "PassengerCategorization");
validateMethodSignature(
    "fetchPassenger:java.util.List",
    "PassengerUtility");
validateMethodSignature(
    "isValidCarrierName:boolean",
    "PassengerUtility");
validateMethodSignature(
    "searchPassengerRecord:PassengerCategorization",
    "UserInterface");
}

private static final Logger LOG = Logger.getLogger("SkeletonValidator");

protected final boolean validateClassName(String className) {

    boolean iscorrect = false;
    try {
        Class.forName(className);
        iscorrect = true;
        LOG.info("Class Name " + className + " is correct");

    } catch (ClassNotFoundException e) {
        LOG.log(Level.SEVERE, "You have changed either the " + "class
name/package. Use the correct package "
                + "and class name as provided in the skeleton");

    } catch (Exception e) {
        LOG.log(Level.SEVERE,
                "There is an error in validating the " + "Class Name.
Please manually verify that the "
                + "Class name is same as skeleton before
uploading");
    }
    return iscorrect;
}

protected final void validateMethodSignature(String methodWithExcptn, String
className) {
    Class cls = null;
    try {

        String[] actualmethods = methodWithExcptn.split(",");
        boolean errorFlag = false;
        String[] methodSignature;
        String methodName = null;
        String returnType = null;

        for (String singleMethod : actualmethods) {
            boolean foundMethod = false;
            methodSignature = singleMethod.split(":");

            methodName = methodSignature[0];
            returnType = methodSignature[1];
    
```

```

        cls = Class.forName(className);
        Method[] methods = cls.getMethods();
        for (Method findMethod : methods) {
            if (methodName.equals(findMethod.getName())) {
                foundMethod = true;
                if (!
(findMethod.getReturnType().getName().equals(returnType))) {
                    errorFlag = true;
                    LOG.log(Level.SEVERE, " You have changed
the " + "return type in '" + methodName
                                         + "' method. Please stick to
the " + "skeleton provided");
                }
            } else {
                LOG.info("Method signature of " +
methodName + " is valid");
            }
        }
        if (!foundMethod) {
            errorFlag = true;
            LOG.log(Level.SEVERE, " Unable to find the given
public method " + methodName
                                         + ". Do not change the " + "given public
method name. " + "Verify it with the skeleton");
        }
    }
    if (!errorFlag) {
        LOG.info("Method signature is valid");
    }
}
} catch (Exception e) {
    LOG.log(Level.SEVERE,
            " There is an error in validating the " + "method
structure. Please manually verify that the "
                                         + "Method signature is same as the
skeleton before uploading");
}
}
}

```

#### USER INTERFACE-

```

import java.util.*;
import java.io.*;

public class UserInterface{
    public static PassengerCategorization searchPassengerRecord(){

        //FILL THE CODE HERE
        return (list,source)->{
            List<Passenger> result = new ArrayList<Passenger>();
            for(Passenger pass : list){

if((pass.getSource().toLowerCase()).equals(source.toLowerCase())){

```

```

                result.add(pass);
            }
        }
        return result;
    };
}

public static void main(String [] args)
{
    //VALIDATION STARTS
    new SkeletonValidator();
    //DO NOT DELETE THIS CODE
    //VALIDATION ENDS

    PassengerCategorization pc = searchPassengerRecord();
    //FILL THE CODE HERE
    System.out.println("Invalid Carrier Records are:");
    PassengerUtility pu = new PassengerUtility();
    List<Passenger> list = null;
    try{
        list = pu.fetchPassenger(new String("PassengerRecord.txt"));
    }
    catch(FileNotFoundException e){
        e.printStackTrace();
    }
    catch(IOException e){
        e.printStackTrace();
    }
    catch(Exception e){
        e.printStackTrace();
    }
    System.out.println("Enter the source to search");
    Scanner sc = new Scanner(System.in);
    String inp = sc.next();

    List<Passenger> result = pc.retrievePassenger_BySource(list,inp);
    if(result.size()==0){
        System.out.println("No Passenger Record");
    }
    else{
        for(Passenger passenger: result){
            System.out.println(passenger.getPassengerName()+""
"+passenger.getPhoneNumber()+" "+passenger.getDateOfJourney()+" "+
passenger.getDestination());
        }
    }
}
}

```

ZEE LAPTOP AGENCY

INVALID LAPTOP-

package com.cts.zeelaptopagency.exception;

```

public class InvalidLaptopIdException extends Exception{
    public InvalidLaptopIdException() {
    }

    public InvalidLaptopIdException(String string) {
        super(string);
    }

}

```

MAIN.JAVA-

```

package com.cts.zeelaptopagency.main;
import com.cts.zeelaptopagency.service.LaptopService;
import java.util.*;
import com.cts.zeelaptopagency.skeletonvalidator.SkeletonValidator;
import java.io.*;
import com.cts.zeelaptopagency.vo.Laptop;
import com.cts.zeelaptopagency.exception.*;

public class Main {
    public static void main(String args[]) {

        // CODE SKELETON - VALIDATION STARTS
        // DO NOT CHANGE THIS CODE
        new SkeletonValidator();
        // CODE SKELETON - VALIDATION ENDS

        //Add your code here to retreive file object from Service
class
        //Add Code here to print valid LaptopDetails returned by
Service Method
        LaptopService l=new LaptopService();
        File f=l.accessFile();
        List<Laptop> lap=l.readData(f);
        System.out.println("The Valid Laptop Details are:-");
        for(Laptop la:lap)
        {
            try{
                if(l.validate(la.getLaptopId())==true){
                    System.out.println(la.toString());
                }
            }
            catch(InvalidLaptopIdException e)
            {
                e.printStackTrace();
            }
        }
    }
}

```

LAPTOP SERVICE.JAVA-

```

package com.cts.zeelaptopagency.service;

```

```

import com.cts.zeelaptopagency.vo.Laptop;
import java.io.File;
import java.io.*;
import java.util.List;
import java.util.*;

import com.cts.zeelaptopagency.exception.InvalidLaptopIdException;
import com.cts.zeelaptopagency.vo.Laptop;

public class LaptopService {

    /**
     * Method to access file
     *
     * @return File
     */
    public File accessFile()
    {

        //Type Code to open text file here
        //File f=new File("LaptopDetails.txt");

        return new File("LaptopDetails.txt"); //TODO change this return value
    }

    /**
     * Method to validate LaptopId and, for invalid laptopId throw
     InvalidLaptopIdException with laptopId as argument
     *
     * @param laptopid
     * @return status
     */
    public boolean validate(String laptopId) throws InvalidLaptopIdException {
        if(laptopId.toUpperCase().startsWith("ZEE"))
        {

            }else{
                throw new InvalidLaptopIdException(laptopId);
            }
            return true;
        //TODO change this return value
    }

    /**
     * Method to read file ,Do necessary operations , writes validated data to
     List and prints invalid laptopID in its catch block
     *
     * @param file
     * @return List
     */
    public List<Laptop> readData(File file)
    { String s1="";

```

```

int c;
FileInputStream file1;
List<Laptop> lap=new LinkedList<>(); ;
try{
    file1=new FileInputStream(file);

    while((c=file1.read())!=-1)
    {
        s1+=(char)c;
    }
}catch(FileNotFoundException e)
{
    e.printStackTrace();
}catch(IOException e)
{
    e.printStackTrace();
}
String[] arr=s1.split("\n");
String[] laptopids=new String[4];
Laptop l;
for(String s:arr)
{
    l=new Laptop();
    laptopids=s.split(",");
    l.setLaptopId(laptopids[0]);
    l.setCustomerName(laptopids[1]);
    l.setBasicCost(Double.parseDouble(laptopids[2]));
    l.setNoOfDays(Integer.parseInt(laptopids[3]));
    this.calculateFinalAmount(l);
    l.setTotalAmount(l.getBasicCost()*l.getNoOfDays());
    lap.add(l);
}

return lap; //TODO change this return value
}

/**
 * Method to find and set totalAmount based on basicCost and noOfdays
 *
 *
 */
public void calculateFinalAmount(Laptop l)
{
    //Type code here to calculate totalAmount based on no of days and basic
cost
    double d=l.getBasicCost()*l.getNoOfDays();
    l.setTotalAmount(d);
}

```

```
    }  
}  
  
}
```

#### SKELETON VALIDATOR-

```
package com.cts.zeelaptopagency.skeletonvalidator;  
  
import java.lang.reflect.Method;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
  
public class SkeletonValidator {  
    public SkeletonValidator() {  
        validateClassName("com.cts.zeelaptopagency.service.LaptopService");  
        validateClassName("com.cts.zeelaptopagency.vo.Laptop");  
        validateMethodSignature(  
            "accessFile:java.io.File,validate:boolean,readData:java.util.List",  
            "com.cts.zeelaptopagency.service.LaptopService");  
    }  
  
    private static final Logger LOG =  
        Logger.getLogger("SkeletonValidator");  
    protected final boolean validateClassName(String className) {  
        boolean iscorrect = false;  
        try {  
            Class.forName(className);  
            iscorrect = true;  
            LOG.info("Class Name " + className + " is correct");  
        } catch (ClassNotFoundException e) {  
            LOG.log(Level.SEVERE, "You have changed either the " +  
                "class name/package. Use the correct package "  
                + "and class name as provided in the  
                skeleton");  
        } catch (Exception e) {  
            LOG.log(Level.SEVERE,  
                "There is an error in validating the " + "Class  
                Name. Please manually verify that the " + "Class name is same as skeleton  
                before uploading");  
        }  
        return iscorrect;  
    }  
  
    protected final void validateMethodSignature(String methodWithExcptn,  
        String className) {
```

```

class cls = null;
try {

    String[] actualmethods = methodWithExcptn.split(",");
    boolean errorFlag = false;
    String[] methodSignature;
    String methodName = null;
    String returnType = null;

    for (String singleMethod : actualmethods) {
        boolean foundMethod = false;
        methodSignature = singleMethod.split(":");

        methodName = methodSignature[0];
        returnType = methodSignature[1];
        cls = Class.forName(className);
        Method[] methods = cls.getMethods();
        for (Method findMethod : methods) {
            if (methodName.equals(findMethod.getName())) {
                foundMethod = true;
                if (!
                    (findMethod.getReturnType().getName().equals(returnType))) {
                    errorFlag = true;
                    LOG.log(Level.SEVERE, " You have
changed the " + "return type in " + methodName
                                         + "' method. Please
stick to the " + "skeleton provided");
                } else {
                    LOG.info("Method signature of " +
methodName + " is valid");
                }
            }
        }
        if (!foundMethod) {
            errorFlag = true;
            LOG.log(Level.SEVERE, " Unable to find the
given public method " + methodName
                                         + ". Do not change the " + "given
public method name. " + "Verify it with the skeleton");
        }
    }
    if (!errorFlag) {
        LOG.info("Method signature is valid");
    }
} catch (Exception e) {
    LOG.log(Level.SEVERE,
           " There is an error in validating the " +
"method structure. Please manually verify that the "
                                         + "Method signature is same as the
skeleton before uploading");
}
}
}

```

LAPTOP.JAVA-

```
package com.cts.zeelaptopagency.vo;
/**
 * Value Object - Laptop
 */
public class Laptop {
    private String laptopId;
    private String customerName;
    private double basicCost;
    private int noOfDays;
    private double totalAmount;

    public Laptop() {
    }

    public String toString() {
        return "Laptop [laptopId="+this.getLaptopId()+",
customerName="+this.getCustomerName()+", basicCost="+this.getBasicCost()+",
noOfDays="+this.getNoOfDays()+", totalAmount="+this.getTotalAmount()+"]";
    }
    public String getLaptopId() {
        return laptopId;
    }
    public void setLaptopId(String laptopId) {
        this.laptopId = laptopId;
    }
    public String getCustomerName() {
        return customerName;
    }
    public void setCustomerName(String customerName) {
        this.customerName = customerName;
    }
    public double getBasicCost() {
        return basicCost;
    }
    public void setBasicCost(double basicCost) {
        this.basicCost = basicCost;
    }
    public int getNoOfDays() {
        return noOfDays;
    }
    public void setNoOfDays(int noOfDays) {
        this.noOfDays = noOfDays;
    }
    public double getTotalAmount() {
        return totalAmount;
    }
    public void setTotalAmount(double totalAmount) {
```

```
        this.totalAmount = totalAmount;
    }
```

```
}
```

#### LAPTOP DETAILS-

```
Laptop Details:  
ZEE01,Jack,2000.50,4  
ZEE02,Dev,4000.00,3  
EEZ03,John,4500.00,5  
ZAE04,Milan,3500.00,4  
ZEE05,Surya,2500.50,7  
ZEE06,Milan,5000.00,6
```

#### DOLLAR CITY THEME PARK

#### USER INTERFACE-

```
package com.ui;  
  
import java.util.Scanner;  
  
import com.utility.ThemeParkBO;  
  
public class UserInterface {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
  
        // Fill the UI code  
        boolean flag = true;  
        int choice = 0;  
        ThemeParkBO park = new ThemeParkBO();  
        while(flag){  
  
            System.out.println("1.Add booking details");  
            System.out.println("2.Average customer booked");  
            System.out.println("3.Exit");  
            System.out.println("Enter your choice");  
            choice = sc.nextInt();  
  
            switch(choice){  
                case 1:  
                    System.out.println("Enter the day");  
                    String day = sc.next();  
                    System.out.println("Enter the customer count");  
                    int cc = sc.nextInt();  
                    park.addBookingDetails(cc);  
  
                    break;  
                case 2:  
                    double res = park.findAverageCustomerBooked();  
                    if(res==0){
```

```

        System.out.println("No records found");
        //break;
    }
    else{
        System.out.println(res);
        //break;
    }
    break;
case 3:
    System.out.println("Thank you for using the application");
    flag = false;
    break;
}
}

}

```

#### THEMEPARKBO.JAVA-

```

package com.utility;

import com.ui.UserInterface;
import java.util.*;
import java.util.List;

public class ThemeParkBO {

    private List<Integer> bookingList = new ArrayList<>();

    public List<Integer> getBookingList() {
        return bookingList;
    }

    public void setBookingList(List<Integer> bookingList) {
        this.bookingList = bookingList;
    }

    // This Method should add the customerCount passed as argument into the
    // bookingList

    public void addBookingDetails(int customerCount) {

        // Fill the Code here
        bookingList.add(customerCount);

    }

    /*
     * This method should return the average customer booked based on the
     * customerCount values available in the bookingList.
     */

    public double findAverageCustomerBooked() {
        double avg;

        // Fill the Code here
        double count = 0;

```

```

        double counter = 0;
        for(int i=0;i<bookingList.size();++i){
            count+=bookingList.get(i);
            counter++;
        }

        if(counter==0) return 0;
        avg = count/counter;
        return avg;
    }
}

```

## PASSENGER

### PASSENGER UTILITY-

```

import java.util.List;
import java.io.*;
import java.util.*;

public class PassengerUtility {

    public List<Passenger> fetchPassenger(String filePath) throws Exception{

        //FILL THE CODE HERE
        List<Passenger> list = new ArrayList<Passenger>();
        String line = "";
        String splitBy = ",";

        BufferedReader br = new BufferedReader(new FileReader(filePath));
        while((line=br.readLine())!=null){
            String[] p = line.split(splitBy);
            Passenger passenger = new
Passenger(p[0],Long.parseLong(p[1]),p[2],p[3],p[4],p[5],p[6]);
            if(isValidCarrierName(passenger.getCarrierName())){
                list.add(passenger);
            }
        }

        return list;
    }

    public boolean isValidCarrierName (String carrierName)
    {
        //FILL THE CODE HERE
        String temp = carrierName;
        if((temp.toLowerCase()).equals("bella")){
            return true;
        }else{
            try{
                throw new InvalidCarrierException(carrierName+" is an Invalid carrier
name.");
            }
            catch(InvalidCarrierException e){
                System.out.println(e.getMessage());
            }
        }
    }
}

```

```
        }
        return false;
    }
}
```

#### PASSENGER CATEGORIZATION-

```
//DO NOT ADD OR EDIT ANY CODE HERE
import java.util.List;
@FunctionalInterface
public interface PassengerCategorization {
    abstract public List<Passenger> retrievePassenger_BySource(List<Passenger>
passengerRecord, String source);
```

```
}
```

#### PASSENGER SKELETON-

```
import java.lang.reflect.Method;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.stream.Stream;

/**
 * @author TJ
 *
 * This class is used to verify if the Code Skeleton is intact and not modified by
participants thereby ensuring smooth auto evaluation
 */
public class SkeletonValidator {

    public SkeletonValidator() {

        validateClassName("PassengerCategorization");
        validateClassName("Passenger");
        validateClassName("InvalidCarrierException");
        validateClassName("PassengerUtility");

        validateMethodSignature(
            "retrievePassenger_BySource:java.util.List",
            "PassengerCategorization");
        validateMethodSignature(
            "fetchPassenger:java.util.List",
            "PassengerUtility");
        validateMethodSignature(
            "isValidCarrierName:boolean",
            "PassengerUtility");
        validateMethodSignature(
            "searchPassengerRecord:PassengerCategorization",
            "UserInterface");
    }

    private static final Logger LOG = Logger.getLogger("SkeletonValidator");
```

```

protected final boolean validateClassName(String className) {

    boolean iscorrect = false;
    try {
        Class.forName(className);
        iscorrect = true;
        LOG.info("Class Name " + className + " is correct");

    } catch (ClassNotFoundException e) {
        LOG.log(Level.SEVERE, "You have changed either the " + "class
name/package. Use the correct package "
                + "and class name as provided in the skeleton");

    } catch (Exception e) {
        LOG.log(Level.SEVERE,
                "There is an error in validating the " + "Class Name.
Please manually verify that the "
                + "Class name is same as skeleton before
uploading");
    }
    return iscorrect;
}

protected final void validateMethodSignature(String methodWithExcptn, String
className) {
    Class cls = null;
    try {

        String[] actualmethods = methodWithExcptn.split(",");
        boolean errorFlag = false;
        String[] methodSignature;
        String methodName = null;
        String returnType = null;

        for (String singleMethod : actualmethods) {
            boolean foundMethod = false;
            methodSignature = singleMethod.split(":");

            methodName = methodSignature[0];
            returnType = methodSignature[1];
            cls = Class.forName(className);
            Method[] methods = cls.getMethods();
            for (Method findMethod : methods) {
                if (methodName.equals(findMethod.getName())) {
                    foundMethod = true;
                    if (!
(findMethod.getReturnType().getName().equals(returnType))) {
                        errorFlag = true;
                        LOG.log(Level.SEVERE, " You have changed
the " + "return type in '" + methodName
                                + "' method. Please stick to
the " + "skeleton provided");

                } else {
                    LOG.info("Method signature of " +
methodName + " is valid");
                }
            }
        }
    }
}

```

```

        }
    }
    if (!foundMethod) {
        errorFlag = true;
        LOG.log(Level.SEVERE, " Unable to find the given
public method " + methodName
                    + ". Do not change the " + "given public
method name. " + "Verify it with the skeleton");
    }

}
if (!errorFlag) {
    LOG.info("Method signature is valid");
}

} catch (Exception e) {
    LOG.log(Level.SEVERE,
            " There is an error in validating the " + "method
structure. Please manually verify that the "
            + "Method signature is same as the
skeleton before uploading");
}
}
}

```

#### PASSENGER USER INTERFACE-

```

import java.util.*;
import java.io.*;

public class UserInterface{
    public static PassengerCategorization searchPassengerRecord(){

        //FILL THE CODE HERE
        return (list,source)->{
            List<Passenger> result = new ArrayList<Passenger>();
            for(Passenger pass : list){

if((pass.getSource().toLowerCase()).equals(source.toLowerCase())){
                result.add(pass);
            }
        }
        return result;
    };
}

public static void main(String [] args)
{
    //VALIDATION STARTS
    new SkeletonValidator();
    //DO NOT DELETE THIS CODE
    //VALIDATION ENDS

    PassengerCategorization pc = searchPassengerRecord();
    //FILL THE CODE HERE
    System.out.println("Invalid Carrier Records are:");
    PassengerUtility pu = new PassengerUtility();
}

```

```

List<Passenger> list = null;
try{
    list = pu.fetchPassenger(new String("PassengerRecord.txt"));
}
catch(FileNotFoundException e){
    e.printStackTrace();
}
catch(IOException e){
    e.printStackTrace();
}
catch(Exception e){
    e.printStackTrace();
}
System.out.println("Enter the source to search");
Scanner sc = new Scanner(System.in);
String inp = sc.next();

List<Passenger> result = pc.retrievePassenger_BySource(list,inp);
if(result.size()==0){
    System.out.println("No Passenger Record");
}
else{
    for(Passenger passenger: result){
        System.out.println(passenger.getPassengerName()+""
"+passenger.getPhoneNumber()+" "+passenger.getDateOfJourney()+" "+
passenger.getDestination());
    }
}
}
}

```

#### INVALID CARRIER EXEMPTION -

```

public class InvalidCarrierException extends Exception{
    //FILL THE CODE HERE
    public InvalidCarrierException (String message){
        super(message);
    }
}

```

#### EMPLOYEE SALARY

##### EMPLOYEE-

```

public class Employee {
3
4 // Fill the code
5 private String employeeName;
6 private int employeeId;
7 private int incrementPercentage;
8 private double salary;
9
10 public void setEmployeeId(int employeeId){
11     this.employeeId=employeeId;
12 }

```

```

13 public int getEmployeeId(){
14 return employeeId;
15 }
16 public void setEmployeeName(String employeeName){
17 this.employeeName=employeeName;
18 }
19 public String getEmployeeName(){
20 return employeeName;
21 }
22 public void setSalary(double salary){
23 this.salary=salary;
24 }
25 public double getSalary(){
26 return salary;
27 }
28 public void setIncrementPercentage(int incrementPercentage){
29 this.incrementPercentage=incrementPercentage;
30 }
31 public int getIncrementPercentage(){
32 return incrementPercentage;
33 }
34 public Employee(int employeeId, String employeeName, double salary){
35 this.employeeId=employeeId;
36 this.employeeName=employeeName;
37 this.salary=salary;
38 }
39 public void findIncrementPercentage(int yearsOfExperience){
40 //Calculate the incremented salay of the employee
41 if(yearsOfExperience>=1&&yearsOfExperience<=5){
42 incrementPercentage=15;
43 }
44 else if(yearsOfExperience>=6&&yearsOfExperience<=10){
45 incrementPercentage=30;
46 }
47 else if(yearsOfExperience>=11&&yearsOfExperience<=15){
48 incrementPercentage=45;
49 }
50 }
51 public double calculateIncrementSalary(){
52 double incrementedSalary=salary+((salary*(double)incrementPercentage)/100);
53 return incrementedSalary;
54 }

```

#### MAIN .JAVA-

```

1 import java.util.*;
2 public class Main {
3
4 public static void main(String[] args)
5 {
6 Scanner read=new Scanner(System.in);
7
8 //Fill the code
9 try
10 {
11 System.out.println("Enter the Employee Id");
12 int id=Integer.parseInt(read.nextLine());
13 System.out.println("Enter the Employee Name");

```

```

14 String name=read.nextLine();
15 System.out.println("Enter the salary");
16 double salary=Double.parseDouble(read.nextLine());
17 System.out.println("Enter the Number of Years in Experience");
18 int exp_year=Integer.parseInt(read.nextLine());
19 Employee e=new Employee(id,name,salary);
20 e.findIncrementPercentage(exp_year);
21
22 double incrementedSalary=e.calculateIncrementSalary();
23 System.out.printf("Incremented Salary %.2f", incrementedSalary);
24 }
25 catch(Exception e)
26 {
27 System.out.println(e);
28 }
29 }
30
31 }

```

## HOME APPLIANCES

### USER INTERFACE-

```

import java.util.*;
public class HomeAppliances {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter Product Id");
        String id = sc.nextLine();
        System.out.println("Enter Product Name");
        String name = sc.nextLine();
        switch (name)
        {
        case "AirConditioner":
        {
            System.out.println("Enter Batch Id");
            String batch = sc.next();
            System.out.println("Enter Dispatch
date");
            String date = sc.next();
            System.out.println("Enter Warranty
Years");
            int years = sc.nextInt();
            System.out.println("Enter type of Air
Conditioner");
            String type = sc.nextLine();
            System.out.println("Enter quantity");
            double capac = sc.nextDouble();
            AirConditioner ob1 = new
AirConditioner(id, name, batch, date, years, type,
capac);
            ob1.calculateProductPrice();
            double price =
            System.out.printf("Price of the Product
is %.2f ", price);
        }
    }
}

```

```

        case "LEDTV":
        {
            System.out.println("Enter Batch Id");
            String batch = sc.nextLine();
            System.out.println("Enter Dispatch
date");
            String date = sc.nextLine();
            System.out.println("Enter Warranty
Years");
            int years = sc.nextInt();
            System.out.println(name);
            System.out.println("Enter size in
inches");
            int size = sc.nextInt();
            System.out.println("Enter quality");
            String quality = sc.nextLine();
            LEDTV ob2 = new LEDTV(id, name, batch,
date, years, size, quality);
            ob2.calculateProductPrice();
            System.out.printf("Price of the Product
is %.2f ", price);
        }
        case "MicrowaveOven":
        {
            System.out.println("Enter Batch Id");
            String batch = sc.nextLine();
            System.out.println("Enter Dispatch
date");
            String date = sc.nextLine();
            System.out.println("Enter Warranty
Years");
            int years = sc.nextInt();
            System.out.println("Enter quantity");
            int quantity = sc.nextInt();
            System.out.println("Enter quality");
            String quality = sc.nextLine();
            MicrowaveOven ob3 = new
MicrowaveOven(id, name, batch, date, years,
quantity, quality);
            ob3.calculateProductPrice();
            System.out.printf("Price of the Product
is %.2f ", price);
        }
        default:
        {
            System.out.println("Provide a valid
Product name");
        }
    }
}

```

#### MICROWAVE.JAVA

```

public class MicrowaveOven extends ElectronicProducts {
    private int quantity;

```

```

private String quality;
public int getQuantity() {
    return quantity;
}
public void setQuantity(int quantity) {
    this.quantity = quantity;
}
public String getQuality() {
    return quality;
}
public void setQuality(String quality) {
    this.quality = quality;
}
// Include Constructor
public MicrowaveOven(String productId, String productName, String
batchId, String
dispatchDate, int warrantyYears,
int quantity, String quality) {
super(productId, productName, batchId, dispatchDate,
warrantyYears);
this.quantity = quantity;
this.quality = quality;
}
public double calculateProductPrice() {
    // Fill Code
    double price = 0;
    if (quality == "Low") {
        price = quantity * 1250;
    } else if (quality == "Medium") {
        price = quantity * 1750;
    } else if (quality == "High") {
        price = quantity * 2000;
    }
    return price;
}
}

```

#### ELECTRONIC PRODUCT.JAVA-

```

public class ElectronicProducts {
    protected String productId;
    protected String productname;
    protected String batchId;
    protected String dispatchDate;
    protected int warrantyYears;
    public String getProductId() {
        return productId;
    }
    public void setProductId(String productId) {
        this.productId = productId;
    }
    public String getProductname() {
        return productname;
    }
    public void setProductname(String productname) {
        this.productname = productname;
    }
    public String getBatchId() {

```

```

                    return batchId;
    }
    public void setBatchId(String batchId) {
        this.batchId = batchId;
    }
    public String getDispatchDate() {
        return dispatchDate;
    }
    public void setDispatchDate(String dispatchDate) {
        this.dispatchDate = dispatchDate;
    }
    public int getWarrantyYears() {
        return warrantyYears;
    }
    public void setWarrantyYears(int warrantyYears) {
        this.warrantyYears = warrantyYears;
    }
    public ElectronicProducts(String productId, String productName,
String batchId, String
dispatchDate,
                                int warrantyYears) {
        this.productId = productId;
        this.productName = productName;
        this.batchId = batchId;
        this.dispatchDate = dispatchDate;
        this.warrantyYears = warrantyYears;
    }
}

```

#### AIR CONDITIONER .JAVA-

```

public class AirConditioner extends ElectronicProducts {
    private String airConditionerType;
    private double capacity;
    public String getAirConditionerType() {
        return airConditionerType;
    }
    public void setAirConditionerType(String airConditionerType) {
        this.airConditionerType = airConditionerType;
    }
    public double getCapacity() {
        return capacity;
    }
    public void setCapacity(double capacity) {
        this.capacity = capacity;
    }
    // Include Constructor
    public AirConditioner(String productId, String productName, String
batchId, String
dispatchDate, int warrantyYears,
                                String airConditionerType, double
capacity) {
        super(productId, productName, batchId, dispatchDate,
warrantyYears);
        this.airConditionerType = airConditionerType;
        this.capacity = capacity;
    }
    public double calculateProductPrice() {

```

```

        // Fill Code
        double cost = 0;
        if (airConditionerType == "Residential") {
            if (capacity == 2.5) {
                cost = 32000;
            } else if (capacity == 4) {
                cost = 40000;
            } else if (capacity == 5.5) {
                cost = 47000;
            }
        } else if (airConditionerType == "Commercial") {
            if (capacity == 2.5) {
                cost = 40000;
            } else if (capacity == 4) {
                cost = 55000;
            } else if (capacity == 5.5) {
                cost = 67000;
            }
        } else if (airConditionerType == "Industrial") {
            if (capacity == 2.5) {
                cost = 47000;
            } else if (capacity == 4) {
                cost = 60000;
            } else if (capacity == 5.5) {
                cost = 70000;
            }
        }
        return cost;
    }
}

```

#### LED TV.JAVA

```

public class LEDTV extends ElectronicProducts {
    private int size;
    private String quality;
    public int getSize() {
        return size;
    }
    public void setSize(int size) {
        this.size = size;
    }
    public String getQuality() {
        return quality;
    }
    public void setQuality(String quality) {
        this.quality = quality;
    }
    // Include Constructor
    public LEDTV(String productId, String productName, String batchId,
String dispatchDate, int
warrantyYears, int size,
                           String quality) {
        super(productId, productName, batchId, dispatchDate,
warrantyYears);
        this.size = size;
        this.quality = quality;
    }
}

```

```

        public double calculateProductPrice() {
            // Fill Code
            double price = 0;
            if (quality == "Low") {
                price = size * 850;
            } else if (quality == "Medium") {
                price = size * 1250;
            } else if (quality == "High") {
                price = size * 1550;
            }
            return price;
        }
    }
}

```

## CINEMA

### BOOK A MOVIE TICKET-

```

public class BookAMovieTicket {
    protected String ticketId;
    protected String customerName;
    protected long mobileNumber;
    protected String emailId;
    protected String movieName;
    public void setticketId( String ticketId){
        this.ticketId=ticketId;
    }
    public void setcustomerName( String customerName){
        this.customerName=customerName;
    }
    public void setmobileNumber( long mobileNumber){
        this.mobileNumber=mobileNumber;
    }
    public void setemailId( String emailId){
        this.emailId=emailId;
    }
    public void setmovieName( String movieName){
        this.movieName=movieName;
    }
    public String getticketId(){
        return ticketId;
    }
    public String getcustomerName(){
        return customerName;
    }
    public String getemailId(){
        return emailId;
    }
    public String getmovieName(){
        return movieName;
    }
    public long getmobileNumber(){
        return mobileNumber;
    }
    public BookAMovieTicket(String ticketId,String customerName,long
mobileNumber,String emailId,String movieName){
}
}

```

```
this.ticketId=ticketId;
this.customerName=customerName;
this.mobileNumber=mobileNumber;
this.emailId=emailId;
this.movieName=movieName;
}
}
```

#### PLATINUM TICKET-

```
public class PlatinumTicket extends BookAMovieTicket {
public PlatinumTicket(String ticketId, String customerName, long mobileNumber,
String emailId, String movieName) {
super(ticketId, customerName, mobileNumber, emailId, movieName);
}
public boolean validateTicketId(){
int count=0;
if(ticketId.contains("PLATINUM")){
count++;
char[] cha=ticketId.toCharArray();
for(int i=8;i<11;i++){
if(cha[i]>='1'&& cha[i]<='9')
count++;
}
if(count==4)
return true;
else
return false;
}
public double caculateTicketCost(int numberoftickets,String ACFacility){
double amount;
if(ACFacility.equals("yes")){
amount=750*numberoftickets;
}
else{
amount=600*numberoftickets;
}
return amount;
}
}
```

#### GOLD TICKET-

```
public class GoldTicket extends BookAMovieTicket {
public GoldTicket(String ticketId, String customerName, long mobileNumber,
String emailId, String movieName) {
super(ticketId, customerName, mobileNumber, emailId, movieName);
}
public boolean validateTicketId(){
int count=0;
if(ticketId.contains("GOLD")){
count++;
char[] cha=ticketId.toCharArray();
for(int i=4;i<7;i++){
if(cha[i]>='1'&& cha[i]<='9')
count++;
}
}
```

```

if(count==4)
return true;
else
return false;
}
public double caculateTicketCost(int numberOfTickets,String ACFacility){
double amount;
if(ACFacility.equals("yes")){
amount=500*numberOfTickets;
}
else{
amount=350*numberOfTickets;
}
return amount;
}
}

```

#### SILVER TICKET-

```

public class SilverTicket extends BookAMovieTicket{
public SilverTicket(String ticketId, String customerName, long mobileNumber,
String emailId, String movieName) {
super(ticketId, customerName, mobileNumber, emailId, movieName);
}
public boolean validateTicketId(){
int count=0;
if(ticketId.contains("SILVER"));
count++;
char[] cha=ticketId.toCharArray();
for(int i=6;i<9;i++){
if(cha[i]>='1'&& cha[i]<='9')
count++;
}
if(count==4)
return true;
else
return false;
}
public double caculateTicketCost(int numberOfTickets,String ACFacility){
double amount;
if(ACFacility.equals("yes")){
amount=250*numberOfTickets;
}
else{
amount=100*numberOfTickets;
}
return amount;
}
}

```

#### USER INTERFACE-

```

import java.util.*;
public class UserInterface {
public static void main(String[] args) {
Scanner sc=new Scanner(System.in);
System.out.println("Enter Ticket Id");

```



## 1. Check Sum of Odd Digits

Write a program to read a number , calculate the sum of odd digits (values) present in the given number.

Include a class **UserMainCode** with a static method **checkSum** which accepts a positive integer . The return type should be 1 if the sum is odd . In case the sum is even return -1 as output.

Create a class **Main** which would get the input as a positive integer and call the static method **checkSum** present in the UserMainCode.

### Input and Output Format:

Input consists of a positive integer n.

Refer sample output for formatting specifications.

### Sample Input 1:

56895

### Sample Output 1:

Sum of odd digits is odd.

### Sample Input 2:

84228

### Sample Output 2:

Sum of odd digits is even.

### MAIN:

```
import java.util.*;
public class Main {
    public static void main(String[] args)
    {
        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        int r=UserMainCode.checkSum(n);
        if(r==1)
        {
            System.out.println("The sum of odd digits are odd");
        }
        else
        {
            System.out.println("The sum of odd digits are even");
        }
        s.close();
    }
}
```

### USERMAINCODE:

```
public class UserMainCode {
    public static int checkSum(int n)
    {
        int n1;
        int sum=0;
        int r;
        while(n!=0)
        {
            n1=n%10;
            if(n1%2!=0)
            {
                sum=sum+n1;
```

```

        }
        n=n/10;
    }
    if(sum%2==0)
    {
        r=-1;
    }
    else
    {
        r=1;
    }
    return r;
}
}

```

## 2. Number Validation

Write a program to read a string of 10 digit number , check whether the string contains a 10 digit number in the format XXX-XXX-XXXX where 'X' is a digit.

Include a class **UserMainCode** with a static method **validateNumber** which accepts a string as input .

The return type of the output should be 1 if the string meets the above specified format . In case the number does not meet the specified format then return -1 as output.

Create a class **Main** which would get the input as a String of numbers and call the static method**validateNumber** present in the UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output is a string specifying the given string is valid or not .

Refer sample output for formatting specifications.

#### **Sample Input 1:**

123-456-7895

#### **Sample Output 1:**

Valid number format

#### **Sample Input 2:**

-123-12344322

#### **Sample Output 2:**

Invalid number format

### **MAIN:**

```

import java.util.*;
public class Main {
    public static void main(String[] args)
    {
        Scanner s=new Scanner(System.in);
        String number=s.next();
        int r=UserMainCode.validateNumber(number);
        if(r==1)
        {
            System.out.println("Valid number format");
        }
    }
}

```

```

        else
        {
            System.out.println("Invalid number format");
        }
        s.close();
    }

}

```

### USERMAINCODE:

```

import java.util.*;
public class UserMainCode {
    public static int validateNumber(String number)
    {
        int b;
        if(number.matches("[0-9]{3}[-]{1}[0-9]{3}[-]{1}[0-9]{4}"))
        {
            b=1;
        }
        else
        {
            b=0;
        }
        return b;
    }
}

```

### **3. Sum of Squares of Even Digits**

Write a program to read a number , calculate the sum of squares of even digits (values) present in the given number.

Include a class **UserMainCode** with a static method **sumOfSquaresOfEvenDigits**

which

accepts a positive integer . The return type (integer) should be the sum of squares of the even digits.

Create a class **Main** which would get the input as a positive integer and call the static method sumOfSquaresOfEvenDigits present in the UserMainCode.

#### **Input and Output Format:**

Input consists of a positive integer n.

Output is a single integer .

Refer sample output for formatting specifications.

#### **Sample Input 1:**

56895

#### **Sample Output 1:**

100

### MAIN:

```

import java.util.*;
public class Main {
    public static void main(String[] args)
    {

```

```

        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        System.out.println(UserMainCode.sumOfSquaresOfEvenDigits(n));
        s.close();
    }

}

```

### **USERMAINCODE:**

```

public class UserMainCode {
    public static int sumOfSquaresOfEvenDigits(int n)
    {
        int n1=0;
        int sum=0;
        while(n!=0)
        {
            n1=n%10;
            if(n1%2==0)
            {
                sum+=n1*n1;
            }
            n=n/10;
        }
        return sum;
    }
}

```

## **4. Fetching Middle Characters from String**

Write a program to read a string of even length and to fetch two middle most characters from the input string and return it as string output.

Include a class **UserMainCode** with a static method **getMiddleChars** which accepts a string

of even length as input . The return type is a string which should be the middle characters of the string.

Create a class **Main** which would get the input as a string and call the static method **getMiddleChars**present in the UserMainCode.

### **Input and Output Format:**

Input consists of a string of even length.

Output is a string .

Refer sample output for formatting specifications.

### **Sample Input 1:**

this

### **Sample Output 1:**

hi

### **Sample Input 1:**

Hell

### **Sample Output 1:**

el

### **MAIN:**

```
import java.util.*;
public class Main {
    public static void main(String[] args)
    {
        Scanner s=new Scanner(System.in);
        String str=s.nextLine();
        System.out.println(UserMainCode.getMiddleChars(str));
        s.close();
    }
}
```

### **USERMAINCODE:**

```
import java.util.*;
public class UserMainCode {
    public static String getMiddleChars(String str)
    {
        StringBuffer sb=new StringBuffer();
        if(str.length()%2==0)
        {
            sb.append(str.substring((str.length()/2)-1,(str.length()/2)+1));
        }
        return sb.toString();
    }
}
```

## **5. Check Characters in a String**

Write a program to read a string and to test whether first and last character are same. The string is said to be valid if the 1st and last character are the same. Else the string is said to be invalid.

Include a class **UserMainCode** with a static method **checkCharacters** which accepts a string as input .

The return type of this method is an int. Output should be 1 if the first character and last character are same . If they are different then return -1 as output.

Create a class **Main** which would get the input as a string and call the static method **checkCharacters** present in the UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output is a string saying characters are same or not .

Refer sample output for formatting specifications.

### **Sample Input 1:**

the picture was great

### **Sample Output 1:**

Valid

### **Sample Input 1:**

this

### **Sample Output 1:**

Invalid

#### **MAIN:**

```

import java.util.*;
public class main {
    public static void main(String[] args)
    {
        Scanner s=new Scanner(System.in);
        String input=s.nextLine();
        int r=UserMainCode.checkCharacters(input);
        if(r==1)
        {
            System.out.println("Valid");
        }
        else
        {
            System.out.println("Invalid");
        }
        s.close();
    }
}

```

#### **USERMAINCODE:**

```

import java.util.*;
public class UserMainCode {
    public static int checkCharacters(String input)
    {
        int r;
        StringTokenizer t = new StringTokenizer(input, " ");
        String s = t.nextToken();
        String s1 =s ;
        while(t.hasMoreTokens())
        {
            s1 = t.nextToken();
        }
        if(s.charAt(0) == s1.charAt(s1.length()-1))
            r=1;
        else
            r=0;
        return r;
    }
}

```

## **6. Forming New Word from a String**

Write a program to read a string and a positive integer n as input and construct a string with first n and last n characters in the given string.

Include a class **UserMainCode** with a static method **formNewWord** which accepts a string

and positive integer .

The return type of the output should be a string (value) of first n character and last n character.

Create a class **Main** which would get the input as a string and integer n and call the static method **formNewWord** present in the UserMainCode.

### **Input and Output Format:**

Input consists of a string of even length.

Output is a string .

Note: The given string length must be  $\geq 2n$ .

Refer sample output for formatting specifications.

### **Sample Input 1:**

California

3

### **Sample Output 1:**

Calnia

### **Sample Input 2:**

this

1

### **Sample Output 2:**

Ts

### **MAIN:**

```
import java.util.*;
public class Main {
    public static void main(String[] args)
    {
        Scanner s=new Scanner(System.in);
        String s1=s.nextLine();
        int n1=s.nextInt();
        System.out.println(UserMainCode.formNewWord(s1,n1));
        s.close();
    }
}
```

### **USERMAINCODE:**

```
import java.util.*;
public class UserMainCode {
    public static String formNewWord(String s1,int n)
    {
        String s = new String();
        if(s1.length()>n)
        {
            s = s1.substring(0,n) + s1.substring(s1.length()-n, s1.length());
            return s;
        }
        else
            return null;
    }
}
```

## 7. Reversing a Number

Write a program to read a positive number as input and to get the reverse of the given number and return it as output.

Include a class **UserMainCode** with a static method **reverseNumber** which accepts a positive integer .

The return type is an integer value which is the reverse of the given number.

Create a **Main** class which gets the input as a integer and call the static method **reverseNumber** present in the **UserMainCode**

### Input and Output Format:

Input consists of a positive integer.

Output is an integer .

Refer sample output for formatting specifications.

### Sample Input 1:

543

### Sample Output 1:

345

### Sample Input 1:

1111

### Sample Output 1:

1111

### MAIN:

```
import java.util.*;
public class Main {
    public static void main(String[] args)
    {
        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        System.out.println(UserMainCode.reverseNumber(n));
        s.close();
    }
}
```

### USERMAINCODE:

```
public class UserMainCode {
    public static int reverseNumber(int n)
    {
        int a,r=0;
        while(n!=0)
        {
            a=n%10;
            r=r*10+a;
            n=n/10;
        }
        return r;
    }
}
```

## 8. Array List Sorting and Merging

Write a code to read two int array lists of size 5 each as input and to merge the two arrayLists, sort the merged arrayList in ascending order and fetch the elements at 2nd, 6th and 8th index into a new arrayList and return the final ArrayList.

Include a class **UserMainCode** with a static method **sortMergedArrayList** which accepts 2 ArrayLists.

The return type is an ArrayList with elements from 2,6 and 8th index position .Array index starts from position 0.

Create a **Main** class which gets two array list of size 5 as input and call the static method **sortMergedArrayList** present in the **UserMainCode**.

**Input and Output Format:**

Input consists of two array lists of size 5.

Output is an array list .

Note - The first element is at index 0.

Refer sample output for formatting specifications.

**Sample Input 1:**

```
3  
1  
17  
11  
19  
5  
2  
7  
6  
20
```

**Sample Output 1:**

```
3  
11  
19
```

**Sample Input 2:**

```
1  
2  
3  
4  
5  
6  
7  
8  
9  
10
```

**Sample Output 2:**

```
3  
7  
9
```

**Main:**

```
import java.util.*;  
public class Main {  
    public static void main(String[] args)
```

```

{
    Scanner s=new Scanner(System.in);
    ArrayList<Integer> list1=new ArrayList<Integer>();
    ArrayList<Integer> list2=new ArrayList<Integer>();
    ArrayList<Integer> newlist=new ArrayList<Integer>();
    for (int i = 0; i < 5; i++)
    {
        list1.add(s.nextInt());
    }
    for (int i = 0; i < 5; i++)
    {
        list2.add(s.nextInt());
    }
    newlist=UserMainCode.sortMergedArrayList(list1,list2);
    for (int i = 0; i < 3; i++)
    {
        System.out.println(newlist.get(i));
    }
    s.close();
}
}

```

#### UserMainCode:

```

import java.util.*;
public class UserMainCode {
    public static ArrayList<Integer> sortMergedArrayList(ArrayList<Integer>
list1,ArrayList<Integer> list2)
    {
        list1.addAll(list2);
        Collections.sort(list1);
        ArrayList<Integer> ans=new ArrayList<Integer>();
        ans.add(list1.get(2));
        ans.add(list1.get(6));
        ans.add(list1.get(8));
        return ans;
    }
}

```

## 9. Validating Date Format

Obtain a date string in the format dd/mm/yyyy. Write code to validate the given date against the given format.

Include a class **UserMainCode** with a static method **validateDate** which accepts a string . The return type of the validateDate method is 1 if the given date format matches the specified format , If the validation fails return the output as -1.

Create a **Main** class which gets date string as an input and call the static method **validateDate** present in the **UserMainCode**.

#### **Input and Output Format:**

Input is a string .

Refer sample output for formatting specifications

#### **Sample Input 1:**

12/06/1987

### **Sample Output 1:**

Valid date format

### **Sample Input 2:**

03/1/1987

### **Sample Output 2:**

Invalid date format

#### **Main:**

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        String s1=sc.nextLine();
        int b=UserMainCode.ValidateDate(s1);
        if(b==1){
            System.out.println("Valid date format");
        }
        else{
            System.out.println("Invalid date format");
        }
        sc.close();
    }
}
```

#### **UserMainCode:**

```
import java.util.*;
import java.text.*;
public class UserMainCode{
    public static int ValidateDate(String s1) {
        if(s1.matches("[0-9]{2}[/]{1}[0-9]{2}[/]{1}[0-9]{4}"))
        {
            SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
            sdf.setLenient(false);
            try {
                Date d1=sdf.parse(s1);
                return 1;
            } catch (ParseException e) {
                return -1;
            }
        }
        else{
            return -1;}}}
```

## **10. Validate Time**

Obtain a time string as input in the following format 'hh:mm am' or 'hh:mm pm'. Write code to validate it using the following rules:

- It should be a valid time in 12 hrs format
- It should have case insensitive AM or PM

Include a class **UserMainCode** with a static method **validateTime** which accepts a string. If the given time is as per the given rules then return 1 else return -1. If the value returned is 1 then print as valid time else print as Invalid time.

Create a **Main** class which gets time(string value) as an input and call the static method **validateTime** present in the **UserMainCode**.

**Input and Output Format:**

Input is a string .

Output is a string .

**Sample Input 1:**

09:59 pm

**Sample Output 1:**

Valid time

**Sample Input 2:**

10:70 AM

**Sample Output 2:**

Invalid time

**Main:**

```
import java.util.*;
public class Main{
public static void main(String []args){
Scanner sc=new Scanner(System.in);
String str=sc.nextLine();
int b=UserMainCode.ValidateTime(str);
if(b==1){
    System.out.println("Valid time");
}
else{
    System.out.println("Invalid time");
}
sc.close();
}}
```

**UserMainCode:**

```
import java.text.*;
import java.util.*;
public class UserMainCode{
public static int ValidateTime(String str){
StringTokenizer st=new StringTokenizer(str,":");
if(st.countTokens()==3)
{
SimpleDateFormat sdf1 = new SimpleDateFormat("h:mm:ss a");
sdf1.setLenient(false);
try
{
Date d2=sdf1.parse(str);
return 1;
}
catch(Exception e)
{
    return -1;
}}
else
{
SimpleDateFormat sdf = new SimpleDateFormat("h:mm a");
sdf.setLenient(false);
try
{
Date d1=sdf.parse(str);
return 1;
}
}}
```

```
        catch(Exception e){  
            return -1;  
        } } } }
```

## 11. String Encryption

Given an input as string and write code to encrypt the given string using following rules and return the encrypted string:

1. Replace the characters at odd positions by next character in alphabet.
  2. Leave the characters at even positions unchanged.

### Note:

- If an odd position character is 'z' replace it by 'a'.
  - Assume the first character in the string is at position 1.

Include a class **UserMainCode** with a static method **encrypt** which accepts a string.

The return type of the output is the encrypted string.

Create a **Main** class which gets string as an input and call the static method **encrypt** present

in the **UserMainCode**.

## **Input and Output Format:**

Input is a string .

Output is a

## Sample Input 1:

## curiosity

## Sample Output

dusipsjtz

San

zzzz

Sam

```
import java.util.*;
public class Main {
public static void main(String[] args) {
    Scanner s=new Scanner(System.in);
    String s1=s.next();
    System.out.println(UserMainCode.encrypt(s1));
    s.close();
}
}
```

## UserMainCode:

```
public class UserMainCode{  
    public static String encrypt(String s1) {  
        StringBuffer sb=new StringBuffer();  
        for(int i=0;i<s1.length();i++){  
            char c=s1.charAt(i);  
            if(i%2==0){  
                if(c==122)  
                    if((c==122)&&(i==0)){  
                        c='A';}  
                else  
                    c=(char) (c-25);  
            }  
            else{  
                if(c==122)  
                    c='Z';  
                else  
                    c=(char) (c+25);  
            }  
            sb.append(c);  
        }  
        return sb.toString();  
    }  
}
```

```

c=(char) (c+1);
sb.append(c);
else
sb.append(c);
return sb.toString();
}}

```

## 12. Password Validation

Given a method with a password in string format as input. Write code to validate the password using following rules:

- Must contain at least one digit
- Must contain at least one of the following special characters @, #, \$
- # Length should be between 6 to 20 characters.

Include a class **UserMainCode** with a static method **validatePassword** which accepts a password string as input.

If the password is as per the given rules return 1 else return -1. If the return value is 1 then print valid password else print as invalid password.

Create a **Main** class which gets string as an input and call the static method **validatePassword** present in the **UserMainCode**.

### Input and Output Format:

Input is a string .

Output is a string .

### Sample Input 1:

%Dhoom%

### Sample Output 1:

Invalid password

### Sample Input 2:

#@6Don

### Sample Output 2:

Valid password

### Main:

```

import java.util.*;
public class Main {
public static void main(String[] args){
    Scanner s=new Scanner(System.in);
    String password=s.next();
    int b=UserMainCode.ValidatePassword(password);
    if(b==1){
        System.out.println("Valid Password");
    }
    else{
        System.out.println("Invalid Password");
    }
    s.close();
}}

```

### UserMainCode:

```

public class UserMainCode{
public static int ValidatePassword(String password){
if(password.matches("[0-9]{1,}.*") && password.matches(".*[@$]{1,}.*"))
&& password.length()>=6 && password.length()<=20)
{
}

```

```

    return 1;
}
else
{
return -1;
}}}

```

### 13. Removing vowels from String

Given a method with string input. Write code to remove vowels from even position in the string.

Include a class **UserMainCode** with a static method **removeEvenVowels** which accepts a string as input.

The return type of the output is string after removing all the vowels.

Create a **Main** class which gets string as an input and call the static method **removeEvenVowels** present in the **UserMainCode**.

#### **Input and Output Format:**

Input is a string .

Output is a string .

Assume the first character is at position 1 in the given string.

#### **Sample Input 1:**

commitment

#### **Sample Output 1:**

cmmitmmt

#### **Sample Input 2:**

capacity

#### **Sample Output 2:**

Cpcty

#### **Main:**

```

import java.util.*;
public class Main {
public static void main(String[] args) {
Scanner s=new Scanner(System.in);
String s1=s.nextLine();
System.out.println(UserMainCode.removeEvenVowels(s1));
s.close();
}}

```

#### **UserMainCode:**

```

public class UserMainCode{
public static String removeEvenVowels(String s1) {
StringBuffer sb1=new StringBuffer();
for(int i=0;i<s1.length();i++)
if((i%2)==0)
sb1.append(s1.charAt(i));
else if((i%2)!=0)
if(s1.charAt(i)!='a' && s1.charAt(i)!='e' &&
s1.charAt(i)!='i' && s1.charAt(i)!='o' && s1.charAt(i)!='u')
if(s1.charAt(i)!='A' && s1.charAt(i)!='E' &&
s1.charAt(i)!='I' && s1.charAt(i)!='O' && s1.charAt(i)!='U')
sb1.append(s1.charAt(i));
return sb1.toString();
}}

```

## **14. Sum of Powers of elements in an array**

Given a method with an int array. Write code to find the power of each individual element according to its position index, add them up and return as output.

Include a class **UserMainCode** with a static method **getSumOfPower** which accepts an integer array as input.

The return type of the output is an integer which is the sum powers of each element in the array.

Create a **Main** class which gets integer array as an input and call the static method **getSumOfPower** present in the **UserMainCode**.

### **Input and Output Format:**

Input is an integer array. First element corresponds to the number(n) of elements in an array. The next inputs corresponds to each element in an array.

Output is an integer .

### **Sample Input 1:**

```
4  
3  
6  
2  
1
```

### **Sample Output 1:**

```
12
```

### **Sample Input 2:**

```
4  
5  
3  
7  
2
```

### **Sample Output 2:**

```
61
```

#### **Main:**

```
import java.util.Scanner;  
public class Main{  
    public static void main(String args[]){  
        Scanner sc=new Scanner(System.in);  
        int n=sc.nextInt();  
        int a[]={};  
        for(int i=0;i<n;i++)  
        {  
            a[i]=sc.nextInt();  
        }  
        System.out.println(UserMainCode.getSumOfPower(n,a));  
        sc.close();  
    }}
```

#### **UserMainCode:**

```
public class UserMainCode{  
    public static int getSumOfPower(int n,int[]a)  
    {{  
        int sum=0;  
        for(int i=0;i<n;i++)
```

```

sum=(int)(sum+Math.pow(a[i], i));
return sum;
}}}

```

## 15.Difference between largest and smallest elements in an array

Given a method taking an int array having size more than or equal to 1 as input. Write code to return the difference between the largest and smallest elements in the array. If there is only one element in the array return the same element as output.

Include a class **UserMainCode** with a static method **getBigDiff** which accepts a integer array as input.

The return type of the output is an integer which is the difference between the largest and smallest elements in the array.

Create a **Main** class which gets integer array as an input and call the static method **getBigDiff** present in the **UserMainCode**.

### **Input and Output Format:**

Input is an integer array. First element in the input represents the number of elements in an array.

Size of the array must be  $\geq 1$

Output is an integer which is the difference between the largest and smallest element in an array.

### **Sample Input 1:**

```

4
3
6
2
1

```

### **Sample Output 1:**

```
5
```

### **Sample Input 2:**

```

4
5
3
7
2

```

### **Sample Output 2:**

```
5
```

### **Main:**

```

import java.util.*;
public class Main {
    public static void main(String args[]){
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        int a[]={};
        for(int i=0;i<n;i++)
        {
            a[i]=sc.nextInt();
        }
        System.out.println(UserMainCode.getBigDiff(a,n));
        sc.close();
    }
}

```

```

        }}
UserMainCode:
import java.util.*;
public class UserMainCode{
public static int getBigDiff(int [] a,int n)
{
    Arrays.sort(a);
    int n1=a[a.length-1]-a[0];
    return n1;
}
}

```

## **16.Find the element position in a reversed string array**

Given a method with an array of strings and one string variable as input. Write code to sort the given array in reverse alphabetical order and return the postion of the given string in the array.

Include a class **UserMainCode** with a static method **getElementPosition** which accepts an

array of strings and a string variable as input.

The return type of the output is an integer which is the position of given string value from the array.

Create a **Main** class which gets string array and a string variable as an input and call the static method **getElementPosition** present in the **UserMainCode**.

### **Input and Output Format:**

Input is an string array. First element in the input represents the size the array

Assume the position of first element is 1.

Output is an integer which is the position of the string variable

### **Sample Input 1:**

```

4
red
green
blue
ivory
ivory

```

### **Sample Output 1:**

```

2

```

### **Sample Input 2:**

```

3
grape
mango
apple
apple

```

### **Sample Output 2:**

```

3

```

### **Main:**

```

import java.util.*;
public class Main {
public static void main(String[] args) {
Scanner sc=new Scanner(System.in);
int fr=sc.nextInt();

```

```

String a[] = new String[fr];
for(int i=0;i<fr;i++)
{
a[i]=sc.next();
}
String ba=sc.next();
UserMainCode.getElementPosition(a,ba);
sc.close();
}

UserMainCode:

import java.util.*;
public class UserMainCode{
public static void getElementPosition(String[] a, String b) {
ArrayList<String>al=new ArrayList<String>();
for(int i=0;i<a.length;i++)
{
al.add(a[i]);
}
Collections.sort(al);
Collections.reverse(al);
for(int i=0;i<al.size();i++)
{
if(b.equals(al.get(i)))
{
System.out.println(i+1);
}}}
}

```

## 17.generate the series

Given a method taking an odd positive Integer number as input. Write code to evaluate the following series:

1+3-5+7-9...+/-n.

Include a class **UserMainCode** with a static method **addSeries** which accepts a positive integer .

The return type of the output should be an integer .

Create a class **Main** which would get the input as a positive integer and call the static method **addSeries** present in the UserMainCode.

### **Input and Output Format:**

Input consists of a positive integer n.

Output is a single integer .

Refer sample output for formatting specifications.

**Sample Input 1:**

9

**Sample Output 1:**

-3

**Sample Input 2:**

11

**Sample Output 2:**

8

**Main**

```
import java.util.*;

public class Main {

    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        System.out.println(UserMainCode.addSeries(n));
        s.close();
    }
}
```

**UserMainCode**

```
import java.util.ArrayList;
import java.util.List;

public class UserMainCode {
```

```

public static int addSeries(int n){

    List<Integer> l1=new ArrayList<Integer>();

    for(int i=1;i<=n;i++)

        if(i%2!=0)

            l1.add(i);

        int n1=l1.get(0);

        for(int i=1;i<l1.size();i++)

            if(i%2!=0)

                n1=n1+l1.get(i);

            else

                n1=n1-l1.get(i);

        return n1;

    }

}

```

## **18.Calculate Electricity Bill**

Given a method calculateElectricityBill() with three inputs. Write code to calculate the current bill.

Include a class **UserMainCode** with a static method **calculateElectricityBill** which accepts 3

inputs .The return type of the output should be an integer .

Create a class **Main** which would get the inputs and call the static method **calculateElectricityBill** present in the UserMainCode.

### **Input and Output Format:**

Input consist of 3 integers.

First input is previous reading, second input is current reading and last input is per unit charge.

Reading Format - XXXXXAAAAA where XXXXX is consumer number and AAAAA is meter

reading.

Output is a single integer corresponding to the current bill.

Refer sample output for formatting specifications.

**Sample Input 1:**

ABC2012345

ABC2012660

4

**Sample Output 1:**

**1260**

**Sample Input 2:**

ABCDE11111

ABCDE11222

3

**Sample Output 2:**

333

**Main**

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);

        String input1=s.next();
        String input2=s.next();
        int input3=s.nextInt();
```

```

        System.out.println(UserMainCode.calculateElectricityBill(input1,input2,input3));

        s.close();

    }

}

```

### UserMainCode

```

public class UserMainCode {

    public static int calculateElectricityBill(String input1, String input2,
int input3)

    {

        int n1=Integer.parseInt(input1.substring(5, input1.length()));

        int n2=Integer.parseInt(input2.substring(5, input2.length()));

        int n=Math.abs((n2-n1)*input3);

        return n;

    }

}

```

## 19.Sum of Digits in a String

Write code to get the sum of all the digits present in the given string.

Include a class **UserMainCode** with a static method **sumOfDigits** which accepts string input.

Return the sum as output. If there is no digit in the given string return -1 as output.

Create a class **Main** which would get the input and call the static

method **sumOfDigits** present in the UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output is a single integer which is the sum of digits in a given string.

Refer sample output for formatting specifications.

**Sample Input 1:**

good23bad4

**Sample Output 1:**

9

**Sample Input 2:**

good

**Sample Output 2:**

-1

**Main**

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.next();
        UserMainCode.sumOfDigits(s1);
        s.close();
    }
}
```

**UserMainCode**

```
public class UserMainCode {
    public static void sumOfDigits(String s1) {
        int sum=0;
        for(int i=0;i<s1.length();i++)
        {
```

```

        char a=s1.charAt(i);

        if(Character.isDigit(a))

        {

            int b=Integer.parseInt(String.valueOf(a));

            sum=sum+b;

        }

    }

    if(sum==0)

    {

        System.out.println(-1);

    }

    else

        System.out.println(sum);

}

}

```

## 20.String Concatenation

Write code to get two strings as input and If strings are of same length simply append them together and return the final string. If given strings are of different length, remove starting characters from the longer string so that both strings are of same length then append them together and return the final string.

Include a class **UserMainCode** with a static method **concatstring** which accepts two string input.

The return type of the output is a string which is the concatenated string.

Create a class **Main** which would get the input and call the static

method **concatstring** present in the UserMainCode.

### **Input and Output Format:**

Input consists of two strings.

Output is a string.

Refer sample output for formatting specifications.

### **Sample Input 1:**

Hello

hi

### **Sample Output 1:**

lohi

### **Sample Input 2:**

Hello

Delhi

### **Sample Output 2:**

HelloDelhi

## **Main**

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);

        String s1=s.next();
        String s2=s.next();

        UserMainCode.concatstring(s1,s2);

        s.close();
    }
}
```

```
}
```

### **} UserMainCode**

```
public class UserMainCode {  
  
    public static void concatstring(String s1, String s2) {  
  
        StringBuffer sb=new StringBuffer();  
  
        int l1=s1.length();  
  
        int l2=s2.length();  
  
        if(l1==l2)  
        {  
            sb.append(s1).append(s2);  
        }  
  
        else if(l1>l2)  
        {  
            sb.append(s1.substring(s1.length()-  
s2.length(),s1.length())).append(s2);  
        }  
  
        else if(l1<l2)  
        {  
            sb.append(s1).append(s2.substring(s2.length()-  
s1.length(),s2.length()));  
        }  
  
        System.out.println(sb);  
    }  
  
}
```

---

## **21. Color Code**

Write a program to read a string and validate whether the given string is a valid color code based on the following rules:

- Must start with "#" symbol
- Must contain six characters after #
- It may contain alphabets from A-F or digits from 0-9

Include a class **UserMainCode** with a static method **validateColorCode** which accepts a string. The return type (integer) should return 1 if the color is as per the rules else return -1.

Create a Class Main which would be used to accept a String and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of a string (Valid or Invalid).

Refer sample output for formatting specifications.

### **Sample Input 1:**

#FF9922

### **Sample Output 1:**

Valid

### **Sample Input 2:**

#FF9(22

### **Sample Output 2:**

Invalid

### **Main**

```
import java.util.*;  
  
public class Main {
```

```

public static void main(String[] args) {
    Scanner s=new Scanner(System.in);
    String s1=s.next();
    int b=UserMainCode.validateColorCode(s1);
    if(b==1)
        System.out.println("Valid");
    else
        System.out.println("Invalid");
    s.close();
}
}

```

## UserMainCode

```

public class UserMainCode {
    public static int validateColorCode(String s1) {
        int b=0,b1=0;
        String s2=s1.substring(1,s1.length());
        if(s1.length()==7)
            if(s1.charAt(0)=='#')
                b1=1;
        if(b1==1){
            /*for(int i=0;i<s2.length();i++){
                char c=s2.charAt(i);
                if(c!='#')
                    }*/
            if(s2.matches("[A-F0-9]{1,}"))
                b=1;
        }
    }
}

```

```
    else  
        b=-1;  
        //break;  
    }  
    return b;  
}  
}
```

---

## 22.Three Digits

Write a program to read a string and check if the given string is in the format "CTS-XXX" where XXX is a three digit number.

Include a class **UserMainCode** with a static method **validatestrings** which accepts a string.

The return type (integer) should return 1 if the string format is correct else return -1.

Create a Class Main which would be used to accept a String and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of a string (Valid or Invalid).

Refer sample output for formatting specifications.

### **Sample Input 1:**

CTS-215

### **Sample Output 1:**

Valid

### **Sample Input 2:**

CTS-2L5

## Sample Output 2:

Invalid

### Main

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner s=new Scanner(System.in);

        String s1=s.next();

        int b=UserMainCode.validatestrings(s1);

        if(b==1){

            System.out.println("Valid");

        }

        else

            System.out.println("Invalid");

        s.close();

    }

}
```

### UserMainCode

```
public class UserMainCode {

    public static int validatestrings(String s1) {

        int res=0;

        if(s1.matches("(CTS)[-]{1}[0-9]{3}"))

        {

            res=1;

        }

        else
```

```
    res=-1;

    return res;

}

}
```

## 23.Removing Keys from HashMap

Given a method with a `HashMap<Integer,string>` as input. Write code to remove all the entries having keys multiple of 4 and return the size of the final hashmap.

Include a class **UserMainCode** with a static method **sizeOfResultandHashMap** which accepts hashmap as input.

The return type of the output is an integer which is the size of the resultant hashmap.

Create a class **Main** which would get the input and call the static method **sizeOfResultandHashMap** present in the UserMainCode.

### **Input and Output Format:**

First input corresponds to the size of the hashmap.

Input consists of a `hashmap<integer,string>`.

Output is an integer which is the size of the hashmap.

Refer sample output for formatting specifications.

### **Sample Input 1:**

3

2

hi

4

hello

12

hello world

**Sample Output 1:**

1

**Sample Input 2:**

3

2

hi

4

sdfsdf

3

asdf

**Sample Output 2:**

2

**Main**

```
import java.util.*;

public class Main {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        int s=sc.nextInt();

        HashMap<Integer, String>hm=new HashMap<Integer, String>();

        for(int i=0;i<s;i++){

            hm.put((sc.nextInt()),(sc.next()));

        }

        System.out.println(UserMainCode.sizeOfResultandHashMap(hm));
    }
}
```

```
sc.close();
```

} }

## UserMainCode

```
import java.util.HashMap;  
  
import java.util.Iterator;
```

```
public class UserMainCode {  
  
    public static int sizeOfResultandHashMap(HashMap<Integer, String> hm) {  
  
        int count=0;  
  
        Iterator<Integer>itr=hm.keySet().iterator();  
  
        while(itr.hasNext())  
  
        {  
  
            int n=itr.next();  
  
            if(n%4!=0)  
  
            {  
  
                count++;  
  
            }  
  
        }  
  
        return count;  
  
    }  
  
}
```

## 24. Largest Element

Write a program to read an int array of odd length, compare the first, middle and the last

elements in the array and return the largest. If there is only one element in the array return the same element.

Include a class **UserMainCode** with a static method **checkLargestAmongCorner** which

accepts an int arrayThe return type (integer) should return the largest element among the first, middle and the last elements.

Create a Class Main which would be used to accept Input array and call the static method present in UserMainCode.

Assume maximum length of array is 20.

### **Input and Output Format:**

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

Output consists of a single Integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

5  
2  
3  
8  
4  
5

### **Sample Output 1:**

8

### **Main**

```
import java.util.*;  
  
public class Main {
```

```

public static void main(String[] args) {

    Scanner s=new Scanner(System.in);

    int n=s.nextInt();

    int a[]={new int[n];

    for(int i=0;i<n;i++){

        a[i]=s.nextInt();

    }

    System.out.println(UserMainCode.checkLargestAmongCorner(a));

    s.close();

}

}

```

## UserMainCode

```

public class UserMainCode {

    public static int checkLargestAmongCorner(int []a)

    {

        int max=0;

        int x,y,z;

        x=a[0];

        y=a[a.length/2];

        z=a[a.length-1];

        if(x>y && x>z)

            max=x;

        else if(y>x && y>z)

            max=y;

```

```

else if(z>x && z>y)
    max=z;
return max;
}
}

```

## 25. nCr

Write a program to calculate the ways in which r elements can be selected from n population, using nCr formula  $nCr = n!/r!(n-r)!$  where first input being n and second input being r.

**Note1 :** n! factorial can be achieved using given formula  $n!=nx(n-1)x(n-2)x..3x2x1$ .

**Note2 :**  $0! = 1$ .

Example  $5!=5x4x3x2x1=120$

Include a class **UserMainCode** with a static method **calculateNcr** which accepts two integers. The return type (integer) should return the value of nCr.

Create a Class Main which would be used to accept Input elements and call the static method present in UserMainCode.

### Input and Output Format:

Input consists of 2 integers. The first integer corresponds to n, the second integer corresponds to r.

Output consists of a single Integer.

Refer sample output for formatting specifications.

### Sample Input 1:

4

3

## **Sample Output 1:**

4

### **Main**

```
import java.util.*;

public class Main {

    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);

        int n=s.nextInt();
        int r=s.nextInt();

        System.out.println(UserMainCode.calculateNcr(n,r));
    }
}
```

### **UserMainCode**

```
public class UserMainCode {

    public static int calculateNcr(int n, int r) {
        int fact=1,fact1=1,fact2=1;

        for(int i=1;i<=n;i++)
        {
            fact=fact*i;
        }

        //System.out.println(fact);

        for(int i=1;i<=r;i++)
        {
            fact1=fact1*i;
        }
    }
}
```

```

        }

        //System.out.println(fact1);

        for(int i=1;i<=(n-r);i++)

        {

            fact2=fact2*i;

        }

        //System.out.println(fact2);

        int res=fact/(fact1*fact2);

        return res;

    }

}

```

## 26.Sum of Common Elements

Write a program to find out sum of common elements in given two arrays. If no common elements are found print - “No common elements”.

Include a class **UserMainCode** with a static method **getSumOfIntersection** which accepts

two integer arrays and their sizes. The return type (integer) should return the sum of common elements.

Create a Class Main which would be used to accept 2 Input arrays and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of  $2+m+n$  integers. The first integer corresponds to  $m$  (Size of the 1st array), the second integer corresponds to  $n$  (Size of the 2nd array), followed by  $m+n$  integers corresponding to the array elements.

Output consists of a single Integer corresponds to the sum of common elements or a string

“No common elements”.

Refer sample output for formatting specifications.

Assume the common element appears only once in each array.

**Sample Input 1:**

4

3

2

3

5

1

1

3

9

**Sample Output 1:**

4

**Sample Input 2:**

4

3

2

3

5

1

12

31

9

## **Sample Output 2:**

No common elements

-

### **Main**

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        int n=sc.nextInt();

        int m=sc.nextInt();

        int[] a=new int[n];

        int[] b=new int[m];

        for(int i=0;i<n;i++){
            a[i]=sc.nextInt();
        }

        for(int i=0;i<m;i++){
            b[i]=sc.nextInt();
        }

        int u=UserMainCode.getSumOfIntersection (a,b,n,m);

        if(u== -1)

            System.out.println("No common elements");

        else

            System.out.println(u);

        sc.close();
    }
}
```

### **UserMainCode**

```
public class UserMainCode {
```

```

public static int getSumOfIntersection(int a[],int b[],int n,int m)

{
    int sum=0;

    for(int i=0;i<a.length;i++)

    {
        for(int j=0;j<b.length;j++)

        {if(a[i]==b[j])

        sum=sum+a[i];

        }
        if(sum==0)

        return -1;

        else

        return sum;

    }

}

```

## 27.Validating Input Password

102. Write a code get a password as string input and validate using the rules specified below.

Apply following validations:

1. Minimum length should be 8 characters
2. Must contain any one of these three special characters @ or \_ or #
3. May contain numbers or alphabets.
4. Should not start with special character or number
5. Should not end with special character

Include a class **UserMainCode** with a static method **validatePassword** which accepts password string as input and returns an integer. The method returns 1 if the password is

valid. Else it returns -1.

Create a class **Main** which would get the input and call the static method **validatePassword** present in the UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output is a string Valid or Invalid.

Refer sample output for formatting specifications.

### **Sample Input 1:**

ashok\_23

### **Sample Output 1:**

Valid

### **Sample Input 2:**

1980\_200

### **Sample Output 2:**

Invalid

## **Main**

```
import java.util.*;

public class Main{

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

String a=sc.next();

int e=UserMainCode.validatePassword(a);

if(e==1){

System.out.println("Valid");

}
```

```

else
{
    System.out.println("Invalid");

}
sc.close();
}}

```

## UserMainCode

```

public class UserMainCode {

    public static int validatePassword(String a){

        int d=0;

        if(a.length()>=8){

            if(a.contains("#") || a.contains("@") || a.contains("_"))

            {

                char c= a.charAt(0);

                //System.out.println(c);

                if(Character.isAlphabetic(c))

                {

                    char dd=a.charAt(a.length()-1);

                    //System.out.println(dd);

                    if((Character.isAlphabetic(dd))||(Character.isDigit(dd)))

                    {

                        if(a.matches(".*[0-9]{1,}.*")||a.matches(".*[a-zA-Z]{1,}.*")){

                            d=1;

                        }

                    }

                }

            }

        }

    }

}

```

```
    }  
}  
  
else  
  
d=-1;  
  
return d;  
  
}}}
```

## 28.iD Validation

Write a program to get two string inputs and validate the ID as per the specified format.

Include a class **UserMainCode** with a static method **validateIDLocations** which accepts two

strings as input.

The return type of the output is a string Valid Id or Invalid Id.

Create a class **Main** which would get the input and call the static  
method **validateIDLocations** present in the UserMainCode.

### **Input and Output Format:**

Input consists of two strings.

First string is ID and second string is location. ID is in the format CTS-LLL-XXXX where  
LLL is

the first three letters of given location and XXXX is a four digit number.

Output is a string Valid id or Invalid id.

Refer sample output for formatting specifications.

### **Sample Input 1:**

CTS-hyd-1234

hyderabad

### **Sample Output 1:**

Valid id

### **Sample Input 2:**

CTS-hyd-123

hyderabad

### **Sample Output 2:**

Invalid id

## **Main**

```
import java.util.*;

public class Main3 {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        String s1=sc.next();

        String s2=sc.next();

        boolean b=UserMainCode3.validateIDLocations(s1,s2);

        if(b==true)

            System.out.println("Valid id");

        else

            System.out.println("Invalid id");

        sc.close();
    }
}
```

```
}
```

```
}
```

---

### UserMainCode

```
import java.util.StringTokenizer;

public class UserMainCode3 {

    public static boolean validateIDLocations(String s1, String s2) {

        String s3=s2.substring(0, 3);

        boolean b=false;

        StringTokenizer t=new StringTokenizer(s1,"-");

        String s4=t.nextToken();

        String s5=t.nextToken();

        String s6=t.nextToken();

        if(s4.equals("CTS") && s5.equals(s3) && s6.matches("[0-9]{4}"))

            b=true;

        else{

            b=false;
        }

        return b;
    }
}
```

### **29.Remove Elements**

Write a program to remove all the elements of the given length and return the size of the final array as output. If there is no element of the given length, return the size of the same

array as output.

Include a class **UserMainCode** with a static method **removeElements** which accepts a string

array, the number of elements in the array and an integer. The return type (integer) should return the size of the final array as output.

Create a Class Main which would be used to accept Input String array and a number and call the static method present in UserMainCode.

Assume maximum length of array is 20.

### **Input and Output Format:**

Input consists of a integers that corresponds to n, followed by n strings and finally m which corresponds to the length value.

Output consists of a single Integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

5

a

bb

b

ccc

ddd

2

### **Sample Output 1:**

4

## **Main**

```
import java.util.*;
```

```

public class Main
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);

        int n=Integer.parseInt(sc.nextLine());

        String[] a=new String[n];

        for(int i=0;i<n;i++)
        {
            a[i]=sc.nextLine();

            int m=Integer.parseInt(sc.nextLine());

            System.out.println(UserMainCode.removeElements(a,m));

            sc.close();
        }
    }
}

```

### UserMainCode

```

public class UserMainCode {

    public static int removeElements(String[] a,int m){

        int u=a.length;

        for(int i=0;i<a.length;i++)
        {
            if(a[i].length()==m)

                u--;
        }

        return u;
    }
}

```

### **30.Find the difference between Dates in months**

Given a method with two date strings in yyyy-mm-dd format as input. Write code to find the difference between two dates in months.

Include a class **UserMainCode** with a static method **getMonthDifference** which accepts two date strings as input.

The return type of the output is an integer which returns the difference between two dates in months.

Create a class **Main** which would get the input and call the static method **getMonthDifference** present in the UserMainCode.

#### **Input and Output Format:**

Input consists of two date strings.

Format of date : yyyy-mm-dd.

Output is an integer.

Refer sample output for formatting specifications.

#### **Sample Input 1:**

2012-03-01

2012-04-16

#### **Sample Output 1:**

1

#### **Sample Input 2:**

2011-03-01

2012-04-16

#### **Sample Output 2:**

13

## Main

```
import java.text.*;
import java.util.*;

public class Main {

    public static void main(String[] args) throws ParseException {
        Scanner sc=new Scanner(System.in);
        String s1=sc.next();
        String s2=sc.next();
        System.out.println(UserMainCode.getMonthDifference(s1,s2));
        sc.close();
    }
}
```

## UserMainCode

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

public class UserMainCode {
    public static int getMonthDifference(String s1, String s2) throws ParseException {
        SimpleDateFormat sdf=new SimpleDateFormat("yyyy-MM-dd");
        Date d1=sdf.parse(s1);
        Date d2=sdf.parse(s2);
        Calendar cal=Calendar.getInstance();
    }
}
```

```

    cal.setTime(d1);

    int months1=cal.get(Calendar.MONTH);

    int year1=cal.get(Calendar.YEAR);

    cal.setTime(d2);

    int months2=cal.get(Calendar.MONTH);

    int year2=cal.get(Calendar.YEAR);

    int n=((year2-year1)*12)+(months2-months1);

    return n;

}

}

```

### **31.Sum of cubes and squares of elements in an array**

Write a program to get an int array as input and identify even and odd numbers. If number is odd get cube of it, if number is even get square of it. Finally add all cubes and squares together and return it as output.

Include a class **UserMainCode** with a static method **addEvenOdd** which accepts integer array as input.

The return type of the output is an integer which is the sum of cubes and squares of elements in the array.

Create a class **Main** which would get the input and call the static method **addEvenOdd** present in the UserMainCode.

#### **Input and Output Format:**

Input consists of integer array.

Output is an integer sum.

Refer sample output for formatting specifications.

### **Sample Input 1:**

5

2

6

3

4

5

### **Sample Output 1:**

208

### **Main**

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        int a[]={};
        for(int i=0;i<n;i++){
            a[i]=sc.nextInt();
        }
        System.out.println(UserMainCode.addEvenOdd(a));
        sc.close();
    }
}
```

```
}
```

### **UserMainCode**

```
public class UserMainCode6 {  
  
    public static int addEvenOdd(int[] a) {  
  
        int n1=0,n2=0;  
  
        for(int i=0;i<a.length;i++)  
  
            if(a[i]%2==0)  
  
                n1+=(a[i]*a[i]);  
  
            else  
  
                n2+=(a[i]*a[i]*a[i]);  
  
        return n1+n2;  
  
    }  
  
}
```

### **32.IP Validator**

Write a program to read a string and validate the IP address. Print “Valid” if the IP address is valid, else print “Invalid”.

Include a class **UserMainCode** with a static method **ipValidator** which accepts a string. The

return type (integer) should return 1 if it is a valid IP address else return 2.

Create a Class Main which would be used to accept Input String and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of a string that corresponds to an IP.

Output consists of a string (“Valid” or “Invalid”).

Refer sample output for formatting specifications.

**Note:** An IP address has the format a.b.c.d where a,b,c,d are numbers between 0-255.

### **Sample Input 1:**

132.145.184.210

### **Sample Output 1:**

Valid

### **Sample Input 2:**

132.145.184.290

### **Sample Output 2:**

Invalid

## **Main**

```
import java.util.*;

public class Main7 {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        String ipAddress=sc.next();

        boolean b=UserMainCode7.validateIpAddress(ipAddress);

        if(b==true)

            System.out.println("Valid");

        else

            System.out.println("Invalid");

        sc.close();

    }
}
```

```
}
```

### **UserMainCode**

```
import java.util.StringTokenizer;

public class UserMainCode7 {

    public static boolean validateIpAddress(String ipAddress) {

        boolean b1=false;

        StringTokenizer t=new StringTokenizer(ipAddress,".");

        int a=Integer.parseInt(t.nextToken());

        int b=Integer.parseInt(t.nextToken());

        int c=Integer.parseInt(t.nextToken());

        int d=Integer.parseInt(t.nextToken());

        if((a>=0 && a<=255)&&(b>=0 && b<=255)&&(c>=0 && c<=255)&&(d>=0
        && d<=255))

            b1=true;

        return b1;
    }
}
```

### **33.Difference between two dates in days**

Get two date strings as input and write code to find difference between two dates in days.

Include a class **UserMainCode** with a static method **getDateDifference** which accepts two

date strings as input.

The return type of the output is an integer which returns the difference between two dates in days.

Create a class **Main** which would get the input and call the static method **getDateDifference** present in the UserMainCode.

### **Input and Output Format:**

Input consists of two date strings.

Format of date : yyyy-mm-dd.

Output is an integer.

Refer sample output for formatting specifications.

#### **Sample Input 1:**

2012-03-12

2012-03-14

#### **Sample Output 1:**

2

#### **Sample Input 2:**

2012-04-25

2012-04-28

#### **Sample Output 2:**

3

## **Main**

```
import java.text.ParseException;
import java.util.*;
public class Main {
    public static void main(String[] args) throws ParseException
```

```
{  
  
Scanner s=new Scanner(System.in);  
  
String s1=s.nextLine();  
  
String s2=s.nextLine();  
  
int output=UserMainCode.getDateDifference(s1,s2);  
  
System.out.println(output);  
  
s.close();  
}  
  
}
```

### UserMainCode

```
import java.text.ParseException;  
  
import java.text.SimpleDateFormat;  
  
import java.util.*;  
  
public class UserMainCode {  
  
public static int getDateDifference(String s1,String s2) throws ParseException  
{  
  
SimpleDateFormat sd=new SimpleDateFormat("yyyy-MM-dd");  
  
Date d=sd.parse(s1);  
  
Calendar c=Calendar.getInstance();  
  
c.setTime(d);  
  
long d1=c.getTimeInMillis();  
  
d=sd.parse(s2);  
  
c.setTime(d);  
  
long d2=c.getTimeInMillis();  
  
int n=Math.abs((int) ((d1-d2)/(1000*3600*24)));  
  
return n;  
}
```

```
}
```

### 34.File Extension

Write a program to read a file name as a string and find out the file extension and return it as output. For example, the file sun.gif has the extension gif.

Include a class **UserMainCode** with a static method **fileIdentifier** which accepts a string. The return type (string) should return the extension of the input string (filename).

Create a Class Main which would be used to accept Input String and call the static method present in UserMainCode.

#### **Input and Output Format:**

Input consists of a string that corresponds to a file name.

Output consists of a string(extension of the input string (filename)).

Refer sample output for formatting specifications.

#### **Sample Input 1:**

sun.gif

#### **Sample Output 1:**

Gif

#### **Main**

```
import java.util.*;  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        Scanner s=new Scanner(System.in);  
  
        System.out.println("enter the string");  
  
        String s1=s.nextLine();  
  
        String output=UserMainCode.fileIdentifier(s1);  
  
        System.out.println(output);  
    }  
}
```

```

s.close();

}

}

UserMainCode

import java.util.*;

public class UserMainCode {

public static String fileIdentifier(String s1)

{

StringTokenizer t=new StringTokenizer(s1,".");

t.nextToken();

String s2=t.nextToken();

return s2;

}

}

```

### **35.Find common characters and unique characters in string**

Given a method with two strings as input. Write code to count the common and unique letters in the two strings.

Note:

- Space should not be counted as a letter.
- Consider letters to be case sensitive. ie, "a" is not equal to "A".

Include a class **UserMainCode** with a static method **commonChars** which accepts two strings as input.

The return type of the output is the count of all common and unique characters in the two strings.

Create a class **Main** which would get the inputs and call the static method **commonChars** present in the UserMainCode.

## **Input and Output Format:**

Input consists of two strings.

Output is an integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

a black cow

battle ship

### **Sample Output 1:**

2

**[Explanation :** b, l and a are the common letters between the 2 input strings. But 'a' appears

more than once in the 1st string. So 'a' should not be considered while computing the count value.]

### **Sample Input 2:**

australia

sri lanka

### **Sample Output 2:**

4

## **Main**

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);

        String s1=sc.nextLine();

        String s2=sc.nextLine();

        StringBuffer sb1=new StringBuffer(s1.replace(" ",""));
    }
}
```

```
StringBuffer sb2=new StringBuffer(s2.replace(" ",""));

int output=UserMainCode.commonChars(s1,s2,sb1,sb2);

System.out.println(output);

sc.close();

}

}
```

### UserMainCode

```
import java.util.*;

public class UserMainCode {

public static int commonChars(String s1,String s2,StringBuffer sb1,StringBuffer
sb2) {

for(int i=0;i<sb1.length();i++){

int c=0;

for(int j=i+1;j<sb1.length();j++){

if(sb1.charAt(i)==sb1.charAt(j)){

sb1.deleteCharAt(j);

c++;

}

}

if(c>=1){

sb1.deleteCharAt(i);

}

}

for(int i=0;i<sb2.length();i++){

int c=0;

for(int j=i+1;j<sb2.length();j++){

if(sb2.charAt(i)==sb2.charAt(j)){

sb2.deleteCharAt(j);

}

}

}

}
```

```

c++;
}

}

if(c>=1){

sb2.deleteCharAt(i);

}

}

int count=0;

for(int i=0;i<sb1.length();i++){

for(int j=0;j<sb2.length();j++){

if(sb1.charAt(i)==sb2.charAt(j)){

count++;

}

}

}

return (count);

}
}

```

### **36.Initial Format**

Write a program to input a person's name in the format "FirstName LastName" and return the person name in the following format - "LastName, InitialOfFirstName".

Include a class **UserMainCode** with a static method **nameFormatter** which accepts a string.

The return type (string) should return the expected format.

Create a Class Main which would be used to accept Input String and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string that corresponds to a Person's name.

Output consists of a string(person's name in expected format).

Refer sample output for formatting specifications.

**Sample Input :**

Jessica Miller

**Sample Output:**

Miller, J

**Main**

```
import java.util.*;

public class Main {

    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);

        String s1=s.nextLine();

        String output=UserMainCode.nameFormatter(s1);

        System.out.println(output);

        s.close();
    }
}
```

**UserMainCode**

```
import java.util.*;

public class UserMainCode {

    public static String nameFormatter(String s1) {
        StringBuffer sb=new StringBuffer();

        StringTokenizer st=new StringTokenizer(s1, " ");

        String s2=st.nextToken();

        String s3=st.nextToken();
```

```
sb.append(s3).append(",");
sb.append(s2.substring(0,1).toUpperCase());
return sb.toString();
}
}
```

### 37.Character cleaning

Write a program to input a String and a character, and remove that character from the given String. Print the final string.

Include a class **UserMainCode** with a static method **removeCharacter** which accepts a string

and a character. The return type (string) should return the character cleaned string.

Create a Class Main which would be used to accept Input String and call the static method present in UserMainCode.

#### **Input and Output Format:**

Input consists of a string and a character.

Output consists of a string(the character cleaned string).

Refer sample output for formatting specifications.

#### **Sample Input :**

elephant

e

#### **Sample Output:**

Lphant

#### **Main**

```
import java.util.*;
public class Main {
```

```
public static void main(String[] args) {  
  
    Scanner s=new Scanner(System.in);  
  
    String s1=s.nextLine();  
  
    String c=s.nextLine();  
  
    String output=UserMainCode.removeCharacter(s1,c);  
  
    System.out.println(output);  
  
}  
  
}
```

### **UserMainCode**

```
import java.util.*;  
  
public class UserMainCode {  
  
    public static String removeCharacter(String s1,String c)  
  
    {  
  
        String d=s1.replace(c,"");  
  
        return d;  
  
    }  
  
}
```

### **38.Vowel Check**

Write a program to read a String and check if that String contains all the vowels. Print "yes" if the string contains all vowels else print "no".

Include a class **UserMainCode** with a static method **getVowels** which accepts a string. The return type (integer) should return 1 if the String contains all vowels else return -1.

Create a Class Main which would be used to accept Input String and call the static method present in UserMainCode.

#### **Input and Output Format:**

Input consists of a string.

Output consists of a string("yes" or "no").

Refer sample output for formatting specifications.

**Sample Input 1:**

abceiduosp

**Sample Output 1:**

yes

**Sample Input 2:**

bceiduosp

**Sample Output 2:**

No

**Main**

```
import java.util.*;  
  
public class User {  
  
    public static void main(String[] args) {  
  
        Scanner s=new Scanner(System.in);  
  
        String s1=s.nextLine();  
  
        String s2=s1.toLowerCase();  
  
        int output=UserMainCode.getVowels(s2);  
  
        //System.out.println(output);  
  
        if(output==1)  
        {  
            System.out.println("yes");  
        }  
        else  
            System.out.println("no");  
    }  
}
```

```
s.close();
```

```
}
```

```
}
```

### **UserMainCode**

```
import java.util.*;  
  
public class UserMainCode {  
  
    public static int getVowels(String s2) {  
  
        if(s2.contains("a") && s2.contains("e") && s2.contains("i") && s2.contains("o") &&  
        s2.contains("u"))  
        {  
  
            return 1;  
  
        }  
  
        else  
  
            return -1;  
    }  
}
```

### **39.Swap Characters**

Write a program to input a String and swap the every 2 characters in the string. If size is an odd number then keep the last letter as it is. Print the final swapped string.

Include a class **UserMainCode** with a static method **swapCharacter** which accepts a string.

The return type (String) should return the character swapped string.

Create a Class Main which would be used to accept Input String and call the static method present in UserMainCode.

#### **Input and Output Format:**

Input consists of a string.

Output consists of a string.

Refer sample output for formatting specifications.

**Sample Input 1:**

TRAINER

**Sample Output 1:**

RTIAENR

**Sample Input 2:**

TOM ANDJERRY

**Sample output 2:**

OT MNAJDREYR

## Main

```
import java.util.*;

public class Main {

    public static void main(String[] args) {

        Scanner s=new Scanner(System.in);

        String s1=s.nextLine();

        String output=UserMainCode.swapCharacter(s1);

        System.out.println(output);

        s.close();
    }
}
```

## UserMainCode

```
import java.util.*;

public class UserMainCode {

    public static String swapCharacter(String s1)

    {
```

```

StringBuffer sb=new StringBuffer();

int l=s1.length();

if(l%2==0)

{

for(int i=0;i<s1.length()-1;i=i+2)

{

char a=s1.charAt(i);

char b=s1.charAt(i+1);

sb.append(b).append(a);

}

return sb.toString();

}

else

{

for(int i = 0;i<s1.length()-1;i=i+2)

{

char a=s1.charAt(i);

char b=s1.charAt(i+1);

sb.append(b).append(a);

}

sb.append(s1.charAt(l-1));

return sb.toString();

}
}
}
}

```

#### **40.Average of Elements in Hashmap**

Given a method with a `HashMap<int, float>` as input. Write code to find out avg of all

values whose keys are even numbers. Round the average to two decimal places and return as output.

[Hint : If the average is 5.901, the rounded average value is 5.9 . If the average is 6.333, the rounded average value is 6.33 . ]

Include a class **UserMainCode** with a static method **avgOfEven** which accepts a `HashMap<int, float>` as input.

The return type of the output is a floating point value which is the average of all values whose key elements are even numbers.

Create a class **Main** which would get the input and call the static method **avgOfEven** present in the UserMainCode.

### **Input and Output Format:**

Input consists of the number of elements in the `HashMap` and the `HashMap<int, float>`.

Output is a floating point value that corresponds to the average.

Refer sample output for formatting specifications.

### **Sample Input 1:**

3

1

2.3

2

4.1

6

6.2

### **Sample Output 1:**

5.15

**Sample Input 2:**

3

9

3.1

4

6.3

1

2.6

**Sample Output 2:**

6.3

**Main**

```
import java.util.HashMap;
import java.util.Scanner;
public class Main {
    public static void main(String []args){
        Scanner sc=new Scanner(System.in);
        int s=sc.nextInt();
        HashMap<Integer,Float>hm=new HashMap<Integer,Float>();
        for(int i=0;i<s;i++)
        {
            int r=sc.nextInt();
            float j=sc.nextFloat();
            hm.put(r,j);
        }
        System.out.println(UserMainCode.display(hm));
        sc.close();
    }
}
```

```
}

}

UserMainCode

import java.text.DecimalFormat;

import java.util.*;

public class UserMainCode

{

public static String display(HashMap<Integer,Float>hm)

{

float sum=0;

int count=0;

DecimalFormat df=new DecimalFormat("#.00");

Iterator<Integer> it=hm.keySet().iterator();

while(it.hasNext())

{

int y=it.next();

if(y%2==0)

{

sum=(float) (sum+hm.get(y));

count++;

}}
```

**float** d=sum/count;

**return** df.format(d);

}

}

## 41.Calculate Average – Hash Map

Write a method that accepts the input data as a hash map and finds out the avg of all values whose keys are odd numbers.

Include a class **UserMainCode** with a static method **calculateAverage** which accepts aHashMap<Integer,Double> and the size of the HashMap. The return type (Double) should return the calculated average. Round the average to two decimal places and return it.

Create a Class Main which would be used to accept Input values and store it as a hash map, and call the static method present in UserMainCode.

### Input and Output Format:

Input consists of an integer n corresponds to number of hash map values, followed by 2n values. (index followed by value).

Output consists of a Double.

Refer sample input and output for formatting specifications.

### Sample Input :

```
4
1
3.41
2
4.1
3
1.61
4
2.5
```

### Sample Output :

```
2.51
```

## Main

```
import java.util.*;
import java.text.*;
public class Main {
public static void main(String[] args)
{
HashMap<Integer,Double> hm=new HashMap<Integer,Double>();
Scanner sc=new Scanner(System.in);
int n=sc.nextInt();
for(int i=0;i<n;i++)
{
int a=sc.nextInt();
double s=sc.nextDouble();
hm.put(a,s);
}
System.out.println(UserMaincode.dis(hm));}}
```

## UserMainCode

```
class UserMaincode
{
public static double dis(HashMap<Integer,Double> h1)
```

```

{
double avg=0.0,sum=0.0;
int k=0;
for(Map.Entry m:h1.entrySet())
{
int a=(Integer)m.getKey();
if(a%2!=0)
{
Double d=(Double) m.getValue();
sum=sum+d;
k++;
}
avg = (double)sum/k;
DecimalFormat df = new DecimalFormat(".##");
String b1 = df.format(avg);
double b = Double.parseDouble(b1);
return b;
}

```

## 42.Count Sequential Characters

109.Get a string as input and write code to count the number of characters which gets repeated 3 times consecutively and return that count (ignore case). If no character gets repeated 3 times consecutively return -1.

Include a class **UserMainCode** with a static method **countSequentialChars** which accepts a string as input.

The return type of the output is the repeat count.

Create a class **Main** which would get the input and call the static method **countSequentialChars** present in the UserMainCode.

### **Input and Output Format:**

Input consists a string.

Output is an integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

abcXXXabc

### **Sample Output 1:**

1

### **Sample Input 2:**

aaxxyzAAAx

### **Sample Output 2:**

2

## **Main**

```

import java.util.*;
import java.text.*;

public class Main {
public static void main(String[] args) {
Scanner sc=new Scanner (System.in);
String input1=sc.next();

System.out.println(UserMainCode.consecutiveRepeatingChar(input1));
}

```

```
}
```

### UserMainCode

```
class UserMainCode
{
    public static int consecutiveRepeatingChar(String input1)
    {
        int c=0;
        int n=0;
        for(int i=0;i<input1.length()-1;i++){
            if(input1.charAt(i)==input1.charAt(i+1))
                n++;
            else
                n=0;
            if(n==2)
                c++;
        }
        return c;
    }
}
```

### 43.Length of the Largest Chunk

Write a program to read a string and find the length of the largest chunk in the string. If there are no chunks print “No chunks” else print the length.

NOTE: chunk is the letter which is repeating 2 or more than 2 times.

Include a class **UserMainCode** with a static method **largestChunk** which accepts a string.

The return type (Integer) should return the length of the largest chunk if the chunk is present, else return -1.

Create a Class Main which would be used to accept Input String and call the static method present in UserMainCode.

#### Input and Output Format:

Input consists of a string.

Refer sample output for formatting specifications.

#### Sample Input 1:

You are toooo good

#### Sample Output 1:

4

(Because the largest chunk is letter 'o' which is repeating 4 times)

#### Sample Input 2:

who are u

#### Sample Output 2:

No chunks

#### Main

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        String s1=sc.nextLine();
        System.out.println(UserMainCode.largestChunk(s1));
```

```

}

}

UserMaincode
class UserMainCode
{
public static int largestChunk(String s1) {
int max=1;
int b=0;
StringTokenizer t=new StringTokenizer(s1, " ");
while(t.hasMoreTokens()){
String s2=t.nextToken();
int n=0;
for(int i=0;i<s2.length()-1;i++)
if(s2.charAt(i)==s2.charAt(i+1))
n++;
if(n>max)
{
max=n;
b=max+1;
}
}
return b;
}
}

```

#### **44.Unique Characters in a string**

Write a program that takes a string and returns the number of unique characters in the string. If the given string does not contain any unique characters return -1

Include a class **UserMainCode** with a static method **uniqueCounter** which accepts a string as input.

The return type of the output is the count of all unique characters in the strings.

Create a class **Main** which would get the input and call the static method **uniqueCounter** present in the UserMainCode.

#### **Input and Output Format:**

Input consists a string.

Output is an integer.

Refer sample output for formatting specifications.

#### **Sample Input 1:**

HelloWorld

#### **Sample Output 1:**

5

#### **Sample Input 2:**

coco

#### **Sample Output 2:**

-1

#### **Main**

```

import java.util.*;
import java.text.*;

```

```

public class Main {

```

```

public static void main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);
String s1 = sc.nextLine();
System.out.println(UserMaincode.uniqueCounter(s1));
}}

```

#### UserMainCode

```

class UserMaincode
{
    public static int uniqueCounter(String s1)
    {

StringBuffer sb = new StringBuffer(s1);
for (int i = 0; i < sb.length(); i++) {
    int count = 0;
    for (int j = i + 1; j < sb.length(); j++) {
        if (sb.charAt(i) == sb.charAt(j)) {
            sb.deleteCharAt(j);
            j--;
            count++;
        }
    }
    if (count >= 1) {
        sb.deleteCharAt(i);
        i--;
    }
}
return sb.length();
}
}

```

## 45.Name Shrinking

Write a program that accepts a string as input and converts the first two names into dotseparated initials and printa the output.

Input string format is 'fn mn ln'. Output string format is 'ln [mn's 1st character].[fn's 1st character]'

Include a class **UserMainCode** with a static method **getFormattedString** which accepts a string. The return type (String) should return the shrinked name.

Create a Class Main which would be used to accept Input String and call the static method present in UserMainCode.

#### **Input and Output Format:**

Input consists of a string.

Output consists of a String.

Refer sample output for formatting specifications.

#### **Sample Input:**

Sachin Ramesh Tendulkar

#### **Sample Output:**

Tendulkar R.S

```

Main
import java.text.*;
import java.util.*;

public class Main {
public static void main(String[] args) {
Scanner sc=new Scanner(System.in);
String s1=sc.nextLine();
System.out.println(UserMainCode.getFormatedString(s1));
}
}

UserMainCode
class UserMainCode
{

public static String getFormatedString(String s1) {
StringBuffer sb=new StringBuffer();
StringTokenizer st=new StringTokenizer(s1, " ");
String s2=st.nextToken();
String s3=st.nextToken();
String s4=st.nextToken();
sb.append(s4).append(" ");
sb.append(s3.substring(0,1));
sb.append(".");
sb.append(s2.substring(0,1));
return sb.toString();
}
}

```

## 46.Odd Digit Sum

Write a program to input a String array. The input may contain digits and alphabets ("de5g4G7R"). Extract odd digits from each string and find the sum and print the output. For example, if the string is "AKj375A" then take 3+7+5=15 and not as 375 as digit. Include a class **UserMainCode** with a static method **oddDigitSum** which accepts a string array and the size of the array. The return type (Integer) should return the sum. Create a Class Main which would be used to accept Input Strings and call the static method present in UserMainCode.

Assume maximum length of array is 20.

### **Input and Output Format:**

Input consists of an integer n, corresponds to the number of strings, followed by n Strings.  
Output consists of an Integer.

Refer sample output for formatting specifications.

### **Sample Input :**

```

3
cog2nizant1
al33k
d2t4H3r5

```

### **Sample Output :**

```

15
(1+3+3+3+5)

```

```

Main
import java.util.Scanner;
public class Main {
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
int s1=sc.nextInt();
String[] s2 = new String[s1];
for (int i = 0; i < s1; i++) {
s2[i] = sc.next();
}
System.out.println(UserMainCode.oddDigitSum(s2));
}}

```

### UserMainCode

```

public class UserMainCode {
public static int oddDigitSum (String[] s1) {
int sum=0;
for(int i=0;i<s1.length;i++)
for(int j=0;j<s1[i].length();j++){
char c=s1[i].charAt(j);
if(Character.isDigit(c)){
if(c%2!=0)
{
String t=String.valueOf(c);
int n=Integer.parseInt(t);
sum=sum+n; } }
return sum;
}
}

```

## 47.Unique Number

Write a program that accepts an Integer as input and finds whether the number is Unique or not. Print Unique if the number is “Unique”, else print “Not Unique”.

**Note:** A Unique number is a positive integer (without leading zeros) with no duplicate digits. For example 7, 135, 214 are all unique numbers whereas 33, 3121, 300 are not.

Include a class **UserMainCode** with a static method **getUnique** which accepts an integer. The return type (Integer) should return 1 if the number is unique else return -1.

Create a Class Main which would be used to accept Input Integer and call the static method present in UserMainCode.

### Input and Output Format:

Input consists of an integer .

Output consists of a String (“Unique” or “Not Unique”).

Refer sample output for formatting specifications.

### Sample Input 1:

123

### Sample Output 1:

Unique

### Sample Input 2:

33

### Sample Output 2:

Not Unique

```

import java.util.*;
import java.text.*;

public class Main{
public static void main(String[] args)
{int j=0;
Scanner sc=new Scanner(System.in);
int n=sc.nextInt();
j=UserMainCode.getUnique(n);
if(j>0)
{
System.out.println("Not Unique");
}
else if(j==0)
{
System.out.println("Unique");
}
}}

```

```

class UserMainCode
{
public static int getUnique(int n)
{
int []a=new int[100];
int i=0,count=0;
while(n!=0)
{
int num=n%10;
a[i]=num;
i++;
n=n/10;
}
for(int j=0;j<=i-1;j++)
{
for(int k=j+1;k<=i-1;k++)
{
if(a[j]==a[k]){
count++;
}
}
}
return count;
}
}

```

## 48.Sum of Lowest marks

Given input as HashMap, value consists of marks and rollno as key.Find the sum of the lowest three subject marks from the HashMap.

Include a class **UserMainCode** with a static method **getLowest** which accepts a Hashmap

with marks and rollno.

The return type of the output is the sum of lowest three subject marks.

Create a class **Main** which would get the input and call the static method **getLowest** present

in the UserMainCode.

### **Input and Output Format:**

First line of the input corresponds to the HashMap size.

Input consists a HashMap with marks and rollno.

Output is an integer which is the sum of lowest three subject marks.

Refer sample output for formatting specifications.

### **Sample Input 1:**

```
5  
1  
54  
2  
85  
3  
74  
4  
59  
5  
57
```

### **Sample Output 1:**

```
170
```

### **Sample Input 2:**

```
4  
10  
56  
20  
58  
30  
87  
40  
54
```

### **Sample Output 2:**

```
168
```

## **Main**

```
import java.util.*;  
  
public class Main {  
    public static void main(String args[]){  
        Scanner sc = new Scanner(System.in);  
        int n=Integer.parseInt(sc.nextLine());  
        HashMap<Integer, Integer> h1=new HashMap<Integer, Integer>();  
        for(int i=0;i<n;i++)  
        {  
            h1.put(sc.nextInt(), sc.nextInt());  
        }  
        System.out.println(UserMainCode.getLowest(h1));  
    }  
}
```

```

}
}

UserMainCode
class UserMainCode {
public static int getLowest(HashMap<Integer, Integer> h1)
{
ArrayList<Integer> a1=new ArrayList<Integer>();
int m=0;
Iterator<Integer> it=h1.keySet().iterator();
while(it.hasNext())
{
int x=it.next();
a1.add(h1.get(x));
}
Collections.sort(a1);
m=a1.get(0)+a1.get(1)+a1.get(2);
return m;
}}

```

## 49. Color Code Validation same as 21

Give a String as colour code as input and write code to validate whether the given string is a valid color code or not.

Validation Rule:

String should start with the Character '#'.

Length of String is 7.

It should contain 6 Characters after '#' Symbol.

It should contain Characters between 'A-F' and Digits '0-9'.

If String acceptable the return true otherwise false.

Include a class **UserMainCode** with a static method **validateColourCode** which accepts a string as input.

The return type of the output is a boolean which returns true if its is a valid color code else it returns false.

Create a class **Main** which would get the input and call the static method **validateColourCode** present in the UserMainCode.

**Input and Output Format:**

Input consists a string corresponding to the color code.

Output is a boolean which returns true or false

Refer sample output for formatting specifications.

**Sample Input 1:**

#99FF33

**Sample Output 1:**

true

**Sample Input 2:**

#CCCC99#

**Sample Output 2:**

False

**Main**

```

import java.util.*;
public class Add {
public static void main(String[] args) {
Scanner s=new Scanner(System.in);
String s1=s.next();
boolean b=userMainCode. validateColourCode (s1);
if(b==true)

```

```

System.out.println("valid color code");
else
System.out.println("invalid color code");
}

UserMainCode

static class userMainCode{
public static boolean validateColourCode (String s1)
boolean b=false;
    if(s1.length()==7&&s1.matches("#[A-F0-9]{1,}")){
        b=true;
    }
    return b;
}}}

```

## 50.Repeating set of characters in a string

Get a string and a positive integer n as input .The last n characters should repeat the number of times given as second input.Write code to repeat the set of character from the given string.

Include a class **UserMainCode** with a static method **getString** which accepts a string and an

integer n as input.

The return type of the output is a string with repeated n characters.

Create a class **Main** which would get the input and call the static method **getString** present

in the UserMainCode.

### **Input and Output Format:**

Input consists a string and a positive integer n.

Output is a string with repeated characters.

Refer sample output for formatting specifications.

### **Sample Input 1:**

Cognizant

3

### **Sample Output 1:**

Cognizantantant

### **Sample Input 2:**

myacademy

2

### **Sample Output 2:**

Myacademymymy

### **Main**

```

import java.util.*;
public class Main {

    public static void main(String[] args) {
        Scanner s= new Scanner(System.in);
        String input= s.next();
    }
}

```

```

        int n=s.nextInt();
        System.out.println(userMainCode.getString(input,n));
    }

}

UserMainCode
class userMainCode {

    public static String getString(String input, int n){
        StringBuffer sb=new StringBuffer();
        sb.append(input);
        for (int i=0;i<n;i++){
            sb.append(input.substring(input.length()-n,input.length()));
        }
        return sb.toString();
    }
}

```

### 51.Finding the day of birth

Given an input as date of birth of person, write a program to calculate on which day (MONDAY,TUESDAY....) he was born store and print the day in Upper Case letters.

Include a class **UserMainCode** with a static method **calculateBornDay** which accepts a string as input.

The return type of the output is a string which should be the day in which the person was born.

Create a class **Main** which would get the input and call the static method **calculateBornDay** present in the UserMainCode.

#### **Input and Output Format:**

NOTE: date format should be(dd-MM-yyyy)

Input consists a date string.

Output is a string which the day in which the person was born.

Refer sample output for formatting specifications.

#### **Sample Input 1:**

29-07-2013

#### **Sample Output 1:**

MONDAY

#### **Sample Input 2:**

14-12-1992

#### **Sample Output 2:**

MONDAY

#### **Main**

```

import java.util.*;
import java.text.*;
public class Main {

    public static void main(String[] args) throws ParseException {
        Scanner s= new Scanner(System.in);
        String input= s.next();
        System.out.println(userMainCode.calculateBornDay(input));
    }
}

```

```

    }
}

UserMainCode
class userMainCode{
    public static String calculateBornDay(String input) throws ParseException{
        SimpleDateFormat sdf= new SimpleDateFormat("dd-MM-yyyy");
        SimpleDateFormat sdf1= new SimpleDateFormat("EEEE");
        Date d= sdf.parse(input);
        String s1= sdf1.format(d);
        return s1;
    }
}

```

## 52.Removing elements from HashMap

Given a HashMap as input, write a program to perform the following operation : If the keys are divisible by 3 then remove that key and its values and print the number of remaining keys in the hashmap.

Include a class **UserMainCode** with a static method **afterDelete** which accepts a HashMap  
as input.

The return type of the output is an integer which represents the count of remaining elements in the hashmap.

Create a class **Main** which would get the input and call the static method **afterDelete** present in the UserMainCode.

### **Input and Output Format:**

First input corresponds to the size of hashmap

Input consists a HashMap

Output is an integer which is the count of remaining elements in the hashmap.

Refer sample output for formatting specifications.

### **Sample Input 1:**

```

4
339
RON
1010
JONS
3366
SMITH
2020
TIM

```

### **Sample Output 1:**

```

2

```

### **Sample Input 2:**

```

5
1010
C2WE
6252
XY4E
1212

```

M2ED  
7070  
S2M41ITH  
8585  
J410N

**Sample Output 2:**

3

**Main**

```
import java.util.*;
import java.text.*;
public class Main {

    public static void main(String[] args) {
        Scanner s= new Scanner(System.in);
        HashMap<Integer, String>hm=new HashMap<Integer, String>();
        int n= s.nextInt();
        for(int i=0;i<n;i++){
            hm.put(s.nextInt(),s.next());
        }

        System.out.println(UserMainCode.afterDelete(hm));
        s.close();
    }
}
```

**UserMainCode**

```
class UserMainCode{
    public static int afterDelete(HashMap<Integer, String> hm){
        int count=0;
        Iterator<Integer>itr=hm.keySet().iterator();
        while(itr.hasNext())
        {
            int n=itr.next();
            if(n%3!=0)
            {
                count++;
            }
        }
        return count;
    }
}
```

**53.Experience Calculator**

Write a program to read Date of Joining and current date as Strings and Experience as integer and validate whether the given experience and calculated experience are the same. Print “true” if same, else “false”.

Include a class **UserMainCode** with a static method **calculateExperience** which accepts 2

strings and an integer. The return type is boolean.

Create a Class Main which would be used to accept 2 string (dates) and an integer and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of 2 strings and an integer, where the 2 strings corresponds to the date of joining and current date, and the integer is the experience.

Output is either “true” or “false”.

Refer sample output for formatting specifications.

### **Sample Input 1:**

11/01/2010

01/09/2014

4

### **Sample Output 1:**

true

### **Sample Input 2:**

11/06/2009

01/09/2014

4

### **Sample Output 2:**

False

### **Main**

```
import java.util.*;
import java.text.*;
public class Main {

    public static void main(String[] args) throws ParseException {
        Scanner sc=new Scanner(System.in);
        String a=sc.next();
        String b=sc.next();
        int c=sc.nextInt();
        long res=(userMainCode.calculateExperience(a,b,c));
        if(res==c)
        {
            System.out.println("true");
        }
        else
            System.out.println("false");
    }
}
```

### **UserMainCode**

```
class userMainCode{
    public static long calculateExperience(String a, String b, int c) throws
ParseException{
    SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
    Date d=new Date();
    Date d1=new Date();
    d=sdf.parse(a);
    d1=sdf.parse(b);
    long t=d.getTime();
    long t1=d1.getTime();
    long t3=t1-t;
    long l1=(24 * 60 * 60 * 1000);
```

```

long l=l1*365;
long res=t3/l;
return res;
}
}

```

## 54.Flush Characters

Write a program to read a string from the user and remove all the alphabets and spaces from the String, and **only store special characters and digit** in the output String. Print

the output string.

Include a class **UserMainCode** with a static method **getSpecialChar** which accepts a string.

The return type (String) should return the character removed string.

Create a Class Main which would be used to accept a string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a strings.

Output consists of an String (character removed string).

Refer sample output for formatting specifications.

### **Sample Input :**

cogniz\$#45Ant

### **Sample Output :**

\$#45

#### **Main:**

```

import java.util.*;
import java.text.*;
public class Main {

    public static void main(String[] args) {
        Scanner s= new Scanner(System.in);
        String input=s.next();
        System.out.println(UserMainCode.getSpecialChar(input));

    }
}

```

#### **UserMainCode:**

```

class UserMainCode{
    public static String getSpecialChar(String input){
        int i;
        StringBuffer sb= new StringBuffer();
        for(i=0;i<input.length();i++)
        {
            char a=input.charAt(i);
            if (!Character.isAlphabetic(a))
                sb.append(a);
        }
        return sb.toString();
    }
}

```

```
}
```

## 55.String Repetition

Write a program to read a string and an integer and return a string based on the below rules.

If input2 is equal or greater than 3 then repeat the first three character of the String by given input2 times, separated by a space.

If input2 is 2 then repeat the first two character of String two times separated by a space,  
If input2 is 1 then return the first character of the String.

Include a class UserMainCode with a static method **repeatString** which takes a string & integer and returns a string based on the above rules.

Create a Class Main which would be used to accept Input string and call the static method present in UserMainCode.

### Input and Output Format:

Input consists of a string and integer.

Output consists of a string.

Refer sample output for formatting specifications.

### Sample Input 1:

```
COGNIZANT
```

```
4
```

### Sample Output 1:

```
COG COG COG COG
```

### Sample Input 2:

```
COGNIZANT
```

```
2
```

### Sample Output 2:

```
CO CO
```

```
class Main{
```

```
    public static void main(String[] args) {  
        Scanner s= new Scanner(System.in);  
        System.out.println("enter a string");  
        String input= s.next();  
        int n= s.nextInt();  
        System.out.println(UserMainCode.repeatString(input, n));  
    }
```

```
}
```

```
class UserMainCode{
```

```
    public static String repeatString(String input, int n){  
        StringBuffer sb= new StringBuffer();  
        String s1= new String();  
        if (n==1){  
            s1=input.substring(0,1);  
            sb.append(s1).append(" ");  
        }  
        if(n==2){  
            s1=input.substring(0,2);  
            for(int i=0;i<n;i++)  
                sb.append(s1).append(" ");  
        }  
        if(n>=3){
```

```

        s1=input.substring(0,3);
        for(int i=0;i<n;i++)
            sb.append(s1).append(" ");
    }
    return sb.toString();
}
}

```

## 56.Average of Prime Locations

Write a program to read an integer array and find the average of the numbers located on the Prime location(indexes).

Round the avarage to two decimal places.

Assume that the array starts with index 0.

Include a class UserMainCode with a static method **averageElements** which accepts a single

integer array. The return type (double) should be the average.

Create a Class Main which would be used to accept Input array and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of  $n+1$  integers. The first integer corresponds to  $n$ , the number of elements in the array. The next ' $n$ ' integers correspond to the elements in the array.

Output consists of a single Double value.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20.

### **Sample Input 1:**

```

8
4
1
7
6
5
8
6
9

```

### **Sample Output 1:**

```
7.5
```

#### **Main:**

```

import java.util.*;
import java.text.*;
public class Main {

    public static void main(String[] args) {
        Scanner s= new Scanner(System.in);
        int n,i;
        System.out.println("enter the array size");
        n=s.nextInt();
        int array[]=new int[n];
        for(i=0;i<n;i++){
            array[i]=s.nextInt();
        }
    }
}

```

```

        }
        System.out.println(UserMainCode.AverageElements(array));
        s.close();
    }

}

UserMainCode:
class UserMainCode{
    public static double AverageElements(int array[]){
        int n, sum=0, count=0, count1=0;
        double average;
        n=array.length;
        for(int i=0; i<=n; i++){
            for(int j=1; j<n; j++){
                if(i%j==0)
                    count++;
                if(count==2){
                    sum= sum+array[i];
                    count1++;
                }
            }
        }
        average= sum/count1;
        DecimalFormat df=new DecimalFormat("#.00");
        double ddd=Double.parseDouble(df.format(average));

        return ddd;
    }
}

```

## 57.Common Elements

Write a program to read two integer arrays and find the sum of common elements in both the arrays. If there are no common elements return -1 as output. Include a class UserMainCode with a static method sumCommonElements which accepts two single integer array. The return type (integer) should be the sum of common elements.

Create a Class Main which would be used to accept Input array and call the static method present in UserMainCode. Assume that all the elements will be distinct. Input and Output Format: Input consists of  $2n+1$  integers. The first integer corresponds to  $n$ , the number of elements in the array. The next ' $n$ ' integers correspond to the elements in the array, The last  $n$  elements correspond to the elements of the second array. Output consists of a single Integer value. Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20.

### Sample Input 1:

```

4
1
2
3
4
2

```

3  
6  
7

**Sample Output 1:**

5

**Main:**

```
import java.util.*;
public class Main {
private static Scanner s ;
;
public static void main(String[] args) {
s = new Scanner (System.in);
int n = s.nextInt();
int a[] = new int[n];
int b[] = new int[n];
for(int i=0;i<n;i++)
{
a[i] = s.nextInt();
}
for(int i=0;i<n;i++)
{
b[i] = s.nextInt();
}
System.out.println(UserMainCode.sumCommonElements(a, b));
}
}
```

**UserMainCode:**

```
public class UserMainCode {
public static int sumCommonElements(int a[],int b[]){
int sum = 0 ;
for(int i=0;i<a.length;i++)
{
for(int j=0;j<b.length;j++){
if(a[i]==b[j])
sum = sum + a[i];}
}
if(sum==0)
return -1;
else return sum;
}
}
```

**58. Middle of Array**

Write a program to read an integer array and return the middle element in the array. The size of the array would always be odd. Include a class UserMainCode with a static method getMiddleElement which accepts a single integer array. The return type (integer) should be the middle element in the array. Create a Class Main which would be used to accept Input

array and call the static method present in UserMainCode. Input and Output Format: Input consists of  $n+1$  integers. The first integer corresponds to  $n$ , the number of elements in the array. The next ' $n$ ' integers correspond to the elements in the array. Output consists of a single Integer value. Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 19.

Sample Input 1:

```
5  
1  
5  
23  
64  
9  
Sample Output 1:  
23
```

**Main:**

```
import java.util.*;  
  
public class Main {  
  
    private static Scanner s;  
  
    public static void main(String[] args) {  
  
        s = new Scanner(System.in);  
  
        int n = s.nextInt();  
  
        int[] a = new int[n];  
  
        for(int i=0;i<n;i++){  
  
            a[i] = s.nextInt();  
        }  
  
        System.out.println(UserMainCode.getMiddleElement(a));  
    }  
}
```

**UserMainCode:**

```
public class UserMainCode {  
  
    public static int getMiddleElement(int a[]){  
  
        int n = a.length;  
  
        return a[n/2];  
    }  
}
```

```
}
```

## 59. Simple String Manipulation

Write a program to read a string and return a modified string based on the following rules.

Return the String without the first 2 chars except when

1. keep the first char if it is 'j'
2. keep the second char if it is 'b'.

Include a class UserMainCode with a static method getString which accepts a string. The return type (string) should be the modified string based on the above rules. Consider all letters in the input to be small case. Create a Class Main which would be used to accept Input string and call the static method present in UserMainCode. Input and Output Format: Input consists of a string with maximum size of 100 characters. Output consists of a string. Refer sample output for formatting specifications.

Sample Input 1:

hello

Sample Output 1:

llo

Sample Input 2:

java

Sample Output 2:

Jva

### Main:

```
import java.util.*;  
  
public class Main {  
  
    private static Scanner s;  
  
    public static void main(String[] args) {  
  
        s = new Scanner(System.in);  
  
        String s1 = s.next();  
  
        System.out.println(UserMainCode.getString(s1));  
  
    }  
  
}
```

### UserMainCode:

```
public class UserMainCode {  
  
    public static String getString(String s1){  
  
        StringBuffer sb=new StringBuffer();  
  
        char a=s1.charAt(0);  
  
        char b=s1.charAt(1);  
  
        if(a=='j')  
            sb.append(a);  
        else if(b=='b')  
            sb.append(b);  
        else  
            sb.append(s1.substring(2));  
  
        return sb.toString();  
    }  
}
```

```

        if(a!='j' && b!='b')
            sb.append(s1.substring(2));

        else if(a=='j' && b!='b')
            sb.append("j").append(s1.substring(2));

        else if(a!='j' && b=='b')
            sb.append(s1.substring(1));

        else
            sb.append(s1.substring(0));

        return sb.toString();
    }

}

```

## 60. Date Validation

Write a program to read a string representing a date. The date can be in any of the three formats 1:dd-MM-yyyy 2: dd/MM/yyyy 3: dd.MM.yyyy If the date is valid, print valid else print invalid. Include a class UserMainCode with a static method getValidDate which accepts a string. The return type (integer) should be based on the validity of the date. Create a Class Main which would be used to accept Input string and call the static method present in UserMainCode. Input and Output Format: Input consists of a string. Output consists of a string. Refer sample output for formatting specifications.

Sample Input 1:

03.12.2013

Sample Output 1:

valid

Sample Input 2:

03\$12\$2013

Sample Output 3:

Invalid

### Main:

```

import java.util.*;

public class Main {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        String s= sc.next();

        int b = UserMainCode.getvalues(s);

        if(b==1)
            System.out.println("Valid");
    }
}

```

```

    else
        System.out.println("Invalid");
    }
}

```

### UserMainCode:

```

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class UserMainCode {

    public static int getvalues(String s) {
        if(s.matches("[0-9]{2}[.]{1}[0-9]{2}[.]{1}[0-9]{4}"))
        {
            SimpleDateFormat sdf=new SimpleDateFormat("dd.MM.yyyy");
            sdf.setLenient(false);
            try
            {
                Date d1=sdf.parse(s);
                return 1;
            } catch (ParseException e) {
                return -1;
            }
        }
        else if(s.matches("[0-9]{2}[/]{1}[0-9]{2}[/][0-9]{4}"))
        {
            SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
            sdf.setLenient(false);
            try
            {
                Date d1=sdf.parse(s);

```

```

    return 1;

} catch (ParseException e) {

    return -1;
}

}

else if(s.matches("[0-9]{2}[-]{1}[0-9]{2}[-][0-9]{4})) {

SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");
sdf.setLenient(false);

try

{

Date d1=sdf.parse(s);

return 1;

} catch (ParseException e) {

    return -1;
}

}

else

return -1;
}
}

```

## 61. Boundary Average

Given an int array as input, write a program to compute the average of the maximum and minimum element in the array. Include a class UserMainCode with a static method “getBoundaryAverage” that accepts an integer array as argument and returns a float that corresponds to the average of the maximum and minimum element in the array. Create a class Main which would get the input array and call the static method getBoundaryAverage present in the UserMainCode. Input and Output Format: The first line of the input consists of an integer n, that corresponds to the size of the array. The next n lines consist of integers that correspond to the elements in the array. Assume that the maximum number of elements in the array is 10. Output consists of a single float value that corresponds to the average of the max and min element in the array.

Sample Input :

```
3  
6  
9  
4  
2  
5
```

Sample Output:

```
5.5
```

**Main:**

```
import java.util.*;  
  
import java.util.Arrays;  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        Scanner sc = new Scanner(System.in);  
  
        int s = sc.nextInt();  
  
        int a[] = new int[s];  
  
        for (int i = 0; i < s; i++)  
  
            a[i] = sc.nextInt();  
  
        System.out.println(UserMainCode.getBoundaryAverage(a));  
  
    }  
  
}
```

**UserMainCode**

```
import java.util.Arrays;  
  
public class UserMainCode {  
  
    public static float getBoundaryAverage(int a[] ){  
  
        Arrays.sort(a);  
  
        int sum = a[0] + a[a.length - 1];  
  
        float avg = (float) sum / 2;  
  
        return avg;  
    }  
  
}
```

## 62. Count Vowels

Given a string input, write a program to find the total number of vowels in the given string. Include a class UserMainCode with a static method “countVowels” that accepts a String argument and returns an int that corresponds to the total number of vowels in the given string. Create a class Main which would get the String as input and call the static method countVowels present in the UserMainCode. Input and Output Format: Input consists of a string. Output consists of an integer..

Sample Input:

avinash

Sample Output:

3

### **Main:**

```
import java.util.*;
public class Main {
private static Scanner s;

public static void main(String[] args) {
s = new Scanner(System.in);
String s1= s.next();
System.out.println(countVowels(s1));
}
```

### **UserMainCode**

```
public class UserMainCode{
public static int countVowels(String s1)
{
String s2=s1.toLowerCase();
String s3="aeiou";
int count=0;
for(int i=0;i<s2.length();i++)
{
    for(int j=0;j<s3.length();j++)
    {
        if(s2.charAt(i)==s3.charAt(j))
        {
            count++;
        }
    }
}
return count;
}
```

## 63. Month Name

Given a date as a string input in the format dd-mm-yy, write a program to extract the month and to print the month name in upper case. Include a class UserMainCode with a static method “getMonthName” that accepts a String argument and returns a String that corresponds to the month name. Create a class Main which would get the String as input and

call the static method getMonthName present in the UserMainCode. The month names are {JANUARY, FEBRUARY, MARCH, APRIL, MAY, JUNE, JULY, AUGUST, SEPTEMBER, OCTOBER, NOVEMBER, DECEMBER} Input and Output Format: Input consists of a String. Output consists of a String.

Sample Input:

01-06-82

Sample Output:

JUNE

**Main:**

```
import java.text.ParseException;
import java.util.Scanner;
public class Main {
    public static void main(String[] args) throws ParseException {

        Scanner sc=new Scanner(System.in);
        String s1=sc.nextLine();
        System.out.println(UserMainCode.calculateBornDay(s1));
        sc.close();
    }
}
```

**UserMainCode:**

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
public class UserMainCode {
    public static String calculateBornDay(String s1) throws ParseException
    {
        SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yy");
        SimpleDateFormat sdf1=new SimpleDateFormat("MMMM");
        Date d=sdf.parse(s1);
        String s=sdf1.format(d);
        return s.toUpperCase();
    }
}
```

#### 64. Reverse SubString

Given a string, startIndex and length, write a program to extract the substring from right to left. Assume the last character has index 0. Include a class UserMainCode with a static method “reverseSubstring” that accepts 3 arguments and returns a string. The 1st argument corresponds to the string, the second argument corresponds to the startIndex and the third argument corresponds to the length. Create a class Main which would get a String and 2 integers as input and call the static method reverseSubstring present in the UserMainCode. Input and Output Format: The first line of the input consists of a string. The second line of the input consists of an integer that corresponds to the startIndex. The third line of the input consists of an integer that corresponds to the length of the substring.

Sample Input:

rajasthan

2

3

Sample Output:

hts

**Main:**

```
import java.util.*;
public class Main {
private static Scanner s;
public static void main(String[] args) {
s=new Scanner(System.in);
String input1= s.next();
int input2=s.nextInt();int input3=s.nextInt();
System.out.println(UserMainCode.retrieveString(input1,input2,input3));
}
}
```

**UserMainCode**

```
class UserMainCode {
public static String retrieveString(String input1, int input2, int input3) {
StringBuffer sb=new StringBuffer(input1);
sb.reverse();
String output=sb.substring(input2, input2+input3);
return output;
}
}
```

## 65. String Finder

Given three strings say Searchstring, Str1 and Str2 as input, write a program to find out if Str2 comes after Str1 in the Searchstring.

Include a class **UserMainCode** with a static method “**stringFinder**” that accepts 3 String arguments and returns an integer. The 3 arguments correspond to SearchString, Str1 and Str2. The function returns 1 if Str2 appears after Str1 in the Searchstring. Else it returns 2.

Create a class **Main** which would get 3 Strings as input and call the static method **stringFinder** present in the UserMainCode.

**Input and Output Format:**

Input consists of 3 strings.

The first input corresponds to the SearchString.

The second input corresponds to Str1.

The third input corresponds to Str2.

Output consists of a string that is either “yes” or “no”

**Sample Input 1:**

geniousRajKumarDev

Raj

Dev

**Sample Output 1:**

Yes

Sample Input 2:  
geniousRajKumarDev  
Dev  
Raj

Sample Output 2:  
No

**USERMAINCODE:**

```
public class UserMainCode {  
    public static int stringFinder(String s1, String s2, String s3)  
    {  
        String a1=s1.toLowerCase();  
        String a2=s2.toLowerCase();  
        String a3=s3.toLowerCase();  
        if(a1.contains(a2)&&a1.contains(a3))  
        {  
            if(a1.indexOf(a2)<a1.indexOf(a3))  
            {  
                return 1;  
            }  
            else  
                return 2;  
        }  
        return 0;  
    }  
}
```

**MAIN:**

```
import java.util.*;  
public class Main {  
    public static void main(String[] args) {  
        Scanner s=new Scanner(System.in);  
        String s1=s.next();  
        String s2=s.next();  
        String s3=s.next();  
        int b=UserMainCode.stringFinder(s1, s2, s3);  
        if(b==1)  
        {  
            System.out.println("yes");  
        }  
        else  
            System.out.println("No");  
        s.close();  
    }  
}
```

**66. Phone Number Validator**

Given a phone number as a string input, write a program to verify whether the phone number is valid using the following business rules:

- It should contain only numbers or dashes (-)
- Dashes may appear at any position
- Should have exactly 10 digits

Include a class **UserMainCode** with a static method “**validatePhoneNumber**” that accepts a String input and returns an integer. The method returns 1 if the phone number is valid. Else it returns 2.

Create a class **Main** which would get a String as input and call the static method **validatePhoneNumber** present in the UserMainCode.

#### **Input and Output Format:**

Input consists of a string.

Output consists of a string that is either 'Valid' or 'Invalid'

#### **Sample Input 1:**

265-265-7777

#### **Sample Output 1:**

Valid

#### **Sample Input 2:**

265-65-7777

#### **Sample Output 2:**

Invalid

#### **USERMAINCODE:**

```
public class UserMainCode {  
    public static int validatePhoneNumber(String s1)  
  
    {  
        String s2 = s1.replaceAll("-", "");  
        if (s2.matches("[0-9]{10}"))  
        {  
            return 1;  
        }  
        else  
            return 2;  
    }  
}
```

### MAIN:

```
import java.util.*;  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        Scanner s=new Scanner(System.in);  
  
        String s1=s.nextLine();  
  
        int b=UserMainCode.validatePhoneNumber(s1);  
  
        if(b==1)  
        {  
            System.out.println("Valid");  
        }  
  
        else  
        {  
            System.out.println("Invalid");  
            s.close();  
        }  
  
    }  
}
```

## 67. Month : Number of Days

Given two inputs year and month (Month is coded as: Jan=0, Feb=1 ,Mar=2 ...), write a program to find out total number of days in the given month for the given year. Include a class **UserMainCode** with a static method “**getNumberOfDays**” that accepts 2 integers as arguments and returns an integer. The first argument corresponds to the year and the second argument corresponds to the month code. The method returns an integer corresponding to the number of days in the month.  
Create a class **Main** which would get 2 integers as input and call the static method **getNumberOfDays** present in the **UserMainCode**.

### Input and Output Format:

Input consists of 2 integers that correspond to the year and month code. Output consists of an integer that corresponds to the number of days in the month in the given year.

Sample Input:

2000

1

Sample Output:

29

USERMAINCODE:

```
import java.util.Calendar;
public class UserMainCode {
    public static int getNumberOfDays(int y,int c)
    {
        Calendar cal=Calendar.getInstance();
        cal.set(Calendar.YEAR, y);
        cal.set(Calendar.MONTH, c);
        int day=cal.getActualMaximum(Calendar.DAY_OF_MONTH);
        return day;
    }
}
```

MAIN:

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        int y=s.nextInt();
        int c=s.nextInt();
        System.out.println(UserMainCode.getNumberOfDays(y, c));
        s.close();
    }
}
```

## 68. Negative String

Given a string input, write a program to replace every appearance of the word "is" by "is not".

If the word "is" is immediately preceded or followed by a letter no change should be made to the string .

Include a class **UserMainCode** with a static method "**negativeString**" that accepts a String arguement and returns a String.

Create a class **Main** which would get a String as input and call the static method **negativeString** present in the UserMainCode.

Input and Output Format:

Input consists of a String.

Output consists of a String.

Sample Input 1:

This is just a misconception

Sample Output 1:

This is not just a misconception

**Sample Input 2:**

Today is misty

**Sample Output 2:**

Today is not misty

**USERMAINCODE:**

```
public class UserMainCode {
    public static String negativeString(String s1)
    {
        String str=s1.replace(" is ", " is not ");
        return str;
    }
}
```

**MAIN:**

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.nextLine();
        System.out.println(UserMainCode.negativeString(s1));
        s.close();
    }
}
```

## 69. Validate Number

Given a negative number as string input, write a program to validate the number and to print the corresponding positive number.

Include a class **UserMainCode** with a static method “**validateNumber**” that accepts a string argument and returns a string. If the argument string contains a valid negative number, the method returns the corresponding positive number as a string. Else the method returns -1. Create a class **Main** which would get a String as input and call the static method **validateNumber** present in the UserMainCode.

**Input and Output Format:**

Input consists of a String.

Output consists of a String.

**Sample Input 1:**

-94923

**Sample Output 1:**

94923

**Sample Input 2:**

-6t

**Sample Output 2:**

-1

**USERMAINCODE:**

```
public class UserMainCode {
    public static String validateNumber(String s1)
```

```

{
    String ss="-1";
    if (s1.matches("[ -]{1}[0-9]{1,}"))
    {
        String st=s1.replace(" -","");
        return st;
    }
    else
        return ss;
    }
}

```

### MAIN:

```

import java.util.*;
public class Main {

    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.next();
        System.out.println(UserMainCode.validateNumber(s1));
        s.close();
    }
}

```

## 70. Digits

Write a program to read a non-negative integer n that returns the count of the occurrences of 7 as digit.

Include a class UserMainCode with a static method **countSeven** which accepts the integer value. The return type is integer which is the count value.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

#### Input and Output Format:

Input consists of a integer.

Output consists of integer.

Refer sample output for formatting specifications.

#### Sample Input 1:

717

#### Sample Output 1:

2

#### Sample Input 2:

4534

#### Sample Output 2:

0

#### USERMAINCODE:

```

public class UserMainCode {
    public static int countSeven(int n)
    {
        int rem,sum=0;

```

```

while(n>0)
{
rem=n%10;
if(rem==7)
{
    sum++;
}
n=n/10;
}
return sum;
}
}

```

### MAIN:

```

import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        System.out.println(UserMainCode.countSeven(n));
        s.close();
    }
}

```

## 71. String Processing – III

Write a program to read a string where all the lowercase 'x' chars have been moved to the end of the string.

Include a class UserMainCode with a static method **moveX** which accepts the string. The return type is the modified string.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### Input and Output Format:

Input consists of a string.

Output consists of a string.

Refer sample output for formatting specifications.

#### Sample Input 1:

xxhixx

#### Sample Input 2:

XXxtest

#### Sample Output 1:

hixxxx

#### Sample Output 2:

XXtestxx

### USERMAINCODE:

```

public class UserMainCode {
    public static String moveX(String s1)
{
}

```

```

String s2="";
String s3="";
for(int i=0;i<s1.length();i++)
{
    char c=s1.charAt(i);
    if(c=='x')
    {
        s2=s2+s1.charAt(i);
    }
    else
    {
        s3=s3+s1.charAt(i)
    }
}
String st=s3.concat(s2);
return st;
}
}

```

### MAIN:

```

import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.next();
        System.out.println(UserMainCode.moveX(s1));
        s.close();
    }
}

```

## 72. String Processing – IV

Write a program to read a string and also a number N. Form a new string starting with 1st character and with every Nth character of the given string. Ex - if N is 3, use chars 1, 3, 6,... and so on to form the new String. Assume  $N \geq 1$ .

Include a class UserMainCode with a static method `getStringUsingNthCharacter` which accepts the string and the number n. The return type is the string as per the problem statement.

Create a Class Main which would be used to accept the string and integer and call the static method present in UserMainCode.

### Input and Output Format:

Input consists of a string and integer.

Output consists of a string.

Refer sample output for formatting specifications.

### Sample Input 1:

### Sample Output 1:

HelloWorld

HelWrd

#### **USERMAINCODE:**

```
public class UserMainCode {  
    public static String getStringUsingNthCharacter(String s1,int n)  
    {  
        StringBuffer sb=new StringBuffer();  
        sb.append(s1.charAt(0));  
        for(int i=n-1;i<s1.length();i+=n)  
        {  
            sb.append(s1.charAt(i));  
        }  
        return sb.toString();  
    }  
}
```

#### **MAIN:**

```
import java.util.*;  
public class Main {  
    public static void main(String[] args) {  
        Scanner s=new Scanner(System.in);  
        String s1=s.next();  
        int n=s.nextInt();  
        System.out.println(UserMainCode.getStringUsingNthCharacter(s1, n));  
        s.close();  
    }  
}
```

### **73. Digit Comparison**

Write a program to read two integers and return true if they have the same last digit.  
Include a class UserMainCode with a static method **compareLastDigit** which accepts two integers and returns boolean. (true / false)

Create a Class Main which would be used to accept two integers and call the static method present in UserMainCode.

#### **Input and Output Format:**

Input consists of two integer.

Output consists TRUE / FALSE.

Refer sample output for formatting specifications.

#### **Sample Input 1:**

59

29

### Sample Output 1:

TRUE

#### UserMainCode

```
public class UserMainCode {  
    public static boolean compareLastDigit(int c,int d)  
    {  
        int c1=c%10;  
        int d1=d%10;  
        boolean b=false;  
        if(c1==d1)  
        {  
            b=true;  
        }  
        return b;  
    }  
}
```

#### Main

```
import java.util.*;  
public class Main {  
  
    public static void main(String[] args) {  
        Scanner s=new Scanner(System.in);  
        int c=s.nextInt();  
        int d=s.nextInt();  
        boolean res=UserMainCode.compareLastDigit(c,d);  
        if(res==true)  
        {  
            System.out.println("TRUE");  
        }  
  
        else  
        {  
            System.out.println("FALSE");  
        }  
  
        s.close();  
    }  
}
```

### 74. Duplicates

Given three integers (a,b,c) find the sum. However, if one of the values is the same as another, both the numbers do not count towards the sum and the third number is returned as the sum.

Include a class UserMainCode with a static method **getDistinctSum** which accepts three integers and returns integer.

Create a Class Main which would be used to accept three integers and call the static method

present in UserMainCode.

### **Input and Output Format:**

Input consists of three integers.

Output consists of a integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

```
1  
2  
1
```

### **Sample Output 1:**

```
2
```

### **Sample Input 2:**

```
1  
2  
3
```

### **Sample Output 2:**

```
6
```

### **UserMainCode:**

```
public class UserMainCode {  
    public static int getDistinctSum(int a,int b,int c)  
    {  
        int sum;  
        if(a==b)  
        {  
            sum=c;  
        }  
  
        else if(b==c)  
        {  
            sum=a;  
        }  
        else if(c==a)  
        {  
            sum=b;  
        }  
        else  
        {  
            sum=a+b+c;  
        }  
        return sum;  
    }  
}
```

### **Main:**

```
import java.util.*;  
public class Main {  
  
    public static void main(String[] args) {  
        Scanner s=new Scanner(System.in);  
        int a=s.nextInt();
```

```

        int b=s.nextInt();
        int c=s.nextInt();
        int sum=UserMainCode.getDistinctSum(a, b, c);
        System.out.println(sum);
        s.close();
    }

}

```

## 75. String Processing - MixMania

Write a program to read a string and check if it starts with '\_ix' where '\_' is any one char(a-z, A-Z, 0-9).

If specified pattern is found return true else false.

Include a class UserMainCode with a static method **checkPattern** which accepts the string.

The return type is TRUE / FALSE.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of TRUE / FALSE.

Refer sample output for formatting specifications.

### **Sample Input 1:**

Mix Mania

### **Sample Output 1:**

TRUE

### **UserMainCode**

```

public class UserMainCode {
    public static boolean checkPattern(String str)
    {
        String str1=str.substring(0,3);
        int a=0,b=0,c=0;
        char c1=str1.charAt(0);
        char c2=str1.charAt(1);
        char c3=str1.charAt(2);
        boolean b1=false;
        if(Character.isDigit(c1)||Character.isLetter(c1))
        {
            a=1;
        }
        if(c2=='i')
        {
            b=1;
        }
        if(c3=='x')
        {
            c=1;
        }
        if(a==1&&b==1&&c==1)
        {
            b1=true;
        }
    }
}

```

```

        }
        return b1;
    }

}

Main:
import java.util.*;
public class Main {

    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String str=s.nextLine();
        boolean b2=UserMainCode.checkPattern(str);
        if(b2==true)
        {
            System.out.println("TRUE");
        }
        else
        {
            System.out.println("FALSE");
        }
    s.close();
    }
}

```

## 76. String Processing

Write a program to read a string and return a new string where the first and last chars have been interchanged.

Include a class UserMainCode with a static method **exchangeCharacters** which accepts the string. The return type is the modified string.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of string.

Refer sample output for formatting specifications.

### **Sample Input 1:**

HelloWorld

### **Sample Output 1:**

delloWorlH

### **UserMainCode**

```

public class UserMainCode {
public static String exchangeCharacters(String s1)
{
    String s2=s1.substring(1,s1.length()-1);
    StringBuffer sb=new StringBuffer();
    char c1=s1.charAt(0);
    char c2=s1.charAt(s1.length()-1);
    sb.append(c2).append(s2).append(c1);
    return sb.toString();
}

```

```
}
```

**Main:**

```
import java.util.*;
public class Main {

    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.next();

        System.out.println(UserMainCode.exchangeCharacters(s1));
        s.close();
    }
}
```

## 77. Regular Expression - II

Given a string (s) apply the following rules.

1. String consists of three characters only.
2. The characters should be alphabets only.

If all the conditions are satisfied then print TRUE else print FALSE.

Include a class UserMainCode with a static method **validateString** which accepts the string.

The return type is the boolean formed based on rules.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of a string.

Output consists of TRUE or FALSE .

Refer sample output for formatting specifications.

**Sample Input 1:**

AcB

**Sample Output 1:**

TRUE

**Sample Input 2:**

A2B

**Sample Output 2:**

FALSE

**UserMainCode:**

```
public class UserMainCode {
    public static boolean validateString(String s1)
    {
        boolean b=false;
        if(s1.length()==3)
        {
            if(s1.matches("[a-zA-Z]{3}"))
            {
                b=true;
            }
        }
    }
}
```

```

        }
    }
    return b;
}
}

Main:

import java.util.*;
public class Main {

    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.next();
        boolean b1=userMainCode.validateString(s1);
        if(b1==true)
        {
            System.out.println("TRUE");
        }
        else
        {
            System.out.println("FALSE");
        }
    }

    s.close();
}
}

```

## 78. Strings Processing - Replication

Write a program to read a string and also a number N. Return the replica of original string for n given time.

Include a class UserMainCode with a static method **repeatString** which accepts the the string and the number n. The return type is the string based on the problem statement.

Create a Class Main which would be used to accept the string and integer and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string and integer.

Output consists of a string.

Refer sample output for formatting specifications.

### **Sample Input 1:**

Lily

2

### **Sample Output 1:**

LilyLily

### **UserMainCode:**

```

public class UserMainCode {
    public static String repeatString(String s1,int n)
    {
        StringBuffer sb=new StringBuffer();
        for(int i=0;i<n;i++)

```

```

        {
            sb.append(s1);
        }
    return sb.toString();
}

}

```

### **Main:**

```

import java.util.*;
public class Main {

    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.next();
        int n=s.nextInt();
        System.out.println(UserMainCode.repeatString(s1, n));
    s.close();
    }

}

```

### **79. SumOdd**

Write a program to read an integer and find the sum of all odd numbers from 1 to the given number. [inclusive of the given number]

if N = 9 [ 1,3,5,7,9]. Sum = 25

Include a class UserMainCode with a static method **addOddNumbers** which accepts the number n. The return type is the integer based on the problem statement.

Create a Class Main which would be used to accept the integer and call the static method present in UserMainCode.

#### **Input and Output Format:**

Input consists of a integer.

Output consists of a integer.

Refer sample output for formatting specifications.

#### **Sample Input 1:**

6

#### **Sample Output 1:**

9

#### **UserMainCode:**

```

public class UserMainCode {
    public static int addOddNumbers(int n)
    {
        int sum=0;
        for(int i=1;i<=n;i+=2)
        {
            sum=sum+i;
        }
    return sum;
}

```

```
}
```

### **Main:**

```
import java.util.*;
public class Main {

    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        System.out.println(UserMainCode.addOddNumbers(n));
        s.close();
    }
}
```

## **80. String Processing - V**

Write a program to read a string array, concatenate the array elements one by one separated by comma and return the final string as output.

Include a class UserMainCode with a static method **concatString** which accepts the string array. The return type is the string.

Create a Class Main which would be used to accept the string array and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of an integer n which is the number of elements followed by n string values.

Output consists of the string.

Refer sample output for formatting specifications.

### **Sample Input 1:**

```
3
```

```
AAA
```

```
BBB
```

```
CCC
```

### **Sample Output 1:**

```
AAA,BBB,CCC
```

### **UserMainCode:**

```
public class UserMainCode {
    public static String concatString(int n, String[] s1)
    {
        StringBuffer sb=new StringBuffer();
        for(int i=0;i<s1.length;i++)
        {
            sb.append(s1[i]).append(",");
        }
        String s2=sb.toString();
        String s3=s2.substring(0,s2.length()-1);

        return s3;
    }
}
```

**Main:**

```
import java.util.*;
public class Main {

    public static void main(String[] args) {
Scanner s=new Scanner(System.in);
int n=s.nextInt();
String s1[]={};
for(int i=0;i<n;i++)
{
    s1[i]=s.next();
}
System.out.println(UserMainCode.concatString(n, s1));
s.close();
    }

}
```

**81.Unique Number**

Given three integers (a,b,c) , Write a program that returns the number of unique integers among the three.

Include a class UserMainCode with a static method **calculateUnique** which accepts three integers and returns the count as integer.

Create a Class Main which would be used to accept three integers and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of three integers.

Output consists of a integer.

Refer sample output for formatting specifications.

**Sample Input 1:**

```
12
4
3
```

**Sample Output 1:**

```
3
```

**Sample Input 2:**

```
4
-4
4
```

**Sample Output 2:**

```
2
```

**Main:**

```
import java.util.*;
public class Main {
public static void main(String[]args){
Scanner s=new Scanner(System.in);
int a=s.nextInt();
int b=s.nextInt();
int c=s.nextInt();
System.out.println(UserMainCode.calculateUnique(a, b, c));
```

```

        s.close();
    }
}

```

### UserMainCode:

```

public class UserMainCode {
    public static int calculateUnique(int a,int b,int c)
    {
        int d=0;
        if(a!=b&&a!=c&&b!=c)
        {
            d=3;
        }
        else if(a==b&&a==c&&b==c)
        {
            d=1;
        }
        else if((a!=b&&a==c&&b==c) || (a!=b&&a!=c&&b==c))
        {
            d=2;
        }
        else if((a==b&&a!=c&&b==c) || (a==b&&a!=c&&b!=c))
        {
            d=2;
        }
        else if((a==b&&a==c&&b!=c) || (a!=b&&a==c&&b!=c))
        {
            d=2;
        }

        return d;
    }
}

```

## 82. Math Calculator

Write a program that accepts three inputs, first two inputs are operands in int form and third one being one of the following five operators: +, -, \*, /, %. Implement calculator logic and return the result of the given inputs as per the operator provided. In case of division, Assume the result would be integer.

Include a class UserMainCode with a static method **calculator** which accepts two integers, one operand and returns the integer.

Create a Class Main which would be used to accept three integers and call the static method present in UserMainCode.

### Input and Output Format:

Input consists of two integers and a character.

Output consists of a integer.

Refer sample output for formatting specifications.

### Sample Input 1:

23

2

\*

## Sample Output 1:

46

### Main:

```
import java.util.*;
public class Main {
    public static void main(String[] args){
        Scanner s=new Scanner(System.in);
        int a=s.nextInt();
        int b=s.nextInt();
        char c = s.next().trim().charAt(0);
        System.out.println(UserMainCode.calculator(a, b, c));
        s.close();
    }
}
```

### UserMainCode:

```
public class UserMainCode {
    public static int calculator(int a,int b,char c)
    {
        int a1=0;

        if(c=='*')
        {
            a1=a*b;
        }
        else if(c=='+')
        {
            a1=a+b;
        }
        else if(c=='-')
        {
            a1=a-b;
        }
        else if(c=='/')
        {
            a1=a/b;
        }
        else if(c=='%')
        {
            a1=a%b;
        }
        return a1;
    }
}
```

## 83. Scores

Write a program to read a integer array of scores, if 100 appears at two consecutive locations return true else return false.

Include a class UserMainCode with a static method **checkScores** which accepts the integer

array. The return type is boolean.

Create a Class Main which would be used to accept the integer array and call the static

method present in UserMainCode.

### **Input and Output Format:**

Input consists of an integer n which is the number of elements followed by n integer values.

Output consists of a string that is either 'TRUE' or 'FALSE'.

Refer sample output for formatting specifications.

### **Sample Input 1:**

```
3  
1  
100
```

```
100
```

### **Sample Output 1:**

```
TRUE
```

### **Sample Input 2:**

```
3  
100  
1  
100
```

### **Sample Output 2:**

```
FALSE
```

### **Main:**

```
import java.util.*;  
public class Main {  
    public static void main (String[] args)  
    {  
        Scanner sc = new Scanner(System.in);  
        int n = sc.nextInt();  
        int[] arr = new int[n];  
        for(int i=0;i<n;i++){  
            arr[i] = sc.nextInt();  
        }  
        System.out.println(UserMainCode.checkScores(arr, n));  
        sc.close();  
    }  
}
```

### **UserMainCode:**

```
public class UserMainCode {  
    public static boolean checkScores(int arr[], int n){  
        boolean b = false;  
        for(int i=0;i<n-1;i++){  
            if(arr[i] == 100){  
                if(arr[i+1] == 100){  
                    b = true;  
                    break;  
                }  
            }  
        }  
        return b;  
    }  
}
```

## 84. ArrayFront

Write a program to read a integer array and return true if one of the first 4 elements in the array is 9 else return false.

Note: The array length may be less than 4.

Include a class UserMainCode with a static method **scanArray** which accepts the integer array. The return type is true / false.

Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.

### Input and Output Format:

Input consists of an integer n which is the number of elements followed by n integer values.

Output consists of TRUE / FALSE.

Refer sample output for formatting specifications.

### Sample Input 1:

```
6  
1  
2  
3  
4  
5  
6
```

### Sample Output 1:

FALSE

### Sample Input 2:

```
3  
1  
2  
9
```

### Sample Output 2:

TRUE

### Main:

```
import java.util.*;  
  
public class Main {  
  
    public static void main(String []args) {  
  
        Scanner sc=new Scanner(System.in);  
  
        int s=sc.nextInt();  
  
        int []a=new int[s];  
  
        for(int i=0;i<s;i++)  
  
        {  
  
            a[i]=sc.nextInt();  
  
        }  
    }  
}
```

```

    if(UserMainCode.scanArray(a)==true)
        System.out.println("TRUE");
    else
        System.out.println("FALSE");
    sc.close();
}
}

```

### UserMainCode:

```

public class UserMainCode {
    public static boolean scanArray(int[] a)
    {
        int u=0,l=0;
        boolean b=false;
        if(a.length>=4)
            l=4;
        else
            l=a.length;
        for(int i=0;i<l;i++)
            if(a[i]==9)
                u=10;
            if(u==10)
                b=true;
        return b;
    }
}

```

## 85. Word Count

Given a string array (s) and non negative integer (n) and return the number of elements in the array which have same number of characters as the givent int N.

Include a class UserMainCode with a static method **countWord** which accepts the string array and integer. The return type is the string formed based on rules.

Create a Class Main which would be used to accept the string and integer and call the static method present in UserMainCode.

### Input and Output Format:

Input consists of a an integer indicating the number of elements in the string array followed the elements and ended by the non-negative integer (N).

Output consists of a integer .

Refer sample output for formatting specifications.

### Sample Input 1:

```

4
a
bb
b
ccc

```

1  
**Sample Output 1:**

2

**Sample Input 2:**

5

dog

cat

monkey

bear

fox

3

**Sample Output 2:**

3

**Main:**

```
import java.util.*;
public class Main {
    public static void main(String []args){
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        String[] str=new String[n];
        for(int i=0;i<n;i++)
        {
            str[i]=sc.next();
        }
        int c=sc.nextInt();
        System.out.println(UserMainCode.countWord(n,str,c));
        sc.close();
    }
}
```

**UserMainCode:**

```
public class UserMainCode {
    public static int countWord(int n,String str[],int c)
    {
        int count=0;
        for(int i=0;i<str.length;i++)
        {
            if(str[i].length()==c)
            {
                count++;
            }
        }
        return count;
    }
}
```

## 86. Find Distance

Write a Program that accepts four int inputs( $x_1, y_1, x_2, y_2$ ) as the coordinates of two points.  
Calculate the distance between the two points using the below formula.

Formula : square root of  $((x_1-x_2)^2 + (y_1-y_2)^2)$

Then, Round the result to return an int

Include a class UserMainCode with a static method **findDistance** which accepts four integers. The return type is integer representing the formula.

Create a Class Main which would be used to accept the input integers and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of four integers.

Output consists of a single integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

```
3  
4  
5  
2
```

### **Sample Output 1:**

```
3
```

### **Sample Input 2:**

```
3  
1  
5  
2
```

### **Sample Output 2:**

```
2
```

### **Main:**

```
import java.util.*;  
public class Main {  
    public static void main (String[] args)  
    {  
        Scanner s=new Scanner(System.in);  
        int a=s.nextInt();  
        int b=s.nextInt();  
        int c=s.nextInt();  
        int d=s.nextInt();  
        System.out.println(UserMainCode.findDistance(a,b,c,d));  
        s.close();  
    }  
}
```

### **UserMainCode:**

```
public class UserMainCode {  
    public static int findDistance(int a,int b,int c,int d) {  
        long q=(int)Math.round(Math.sqrt(((a-c)*(a-c))+((b-d)*(b-d))));  
        return (int) q;  
    }  
}
```

## **87. Word Count - II**

Write a program to read a string and count the number of words present in it.

Include a class UserMainCode with a static method **countWord** which accepts the string.

The return type is the integer giving out the count of words.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of a string.

Output consists of integer.

Refer sample output for formatting specifications.

**Sample Input 1:**

Today is Sunday

**Sample Output 1:**

3

**Main:**

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.nextLine();
        UserMainCode.countWord(s1);
        s.close();
    }
}
```

**UserMainCode:**

```
import java.util.StringTokenizer;
public class UserMainCode {
    public static void countWord(String s1){
        StringTokenizer st=new StringTokenizer(s1, " ");
        int n=st.countTokens();
        System.out.println(n);
    }
}
```

## 88. Sum of Max & Min

Write a Program that accepts three integers, and returns the sum of maximum and minimum numbers.

Include a class UserMainCode with a static method getSumMaxMin which accepts three integers. The return type is integer representing the formula.

Create a Class Main which would be used to accept the input integers and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of three integers.

Output consists of a single integer.

Refer sample output for formatting specifications.

**Sample Input 1:**

12

17

19

**Sample Output 1:**

31

```

Main:
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        int a=s.nextInt();
        int b=s.nextInt();
        int c=s.nextInt();
        System.out.println(UserMainCode.getSumMaxMin(a,b,c));
        s.close();
    }
}

```

```

UserMainCode:
public class UserMainCode {
    public static int getSumMaxMin(int a,int b,int c)
    {
        int d=0;
        if(a<b&&b<c)
        {
            d=a+c;
        }
        else if(a<b&&b>c)
        {
            d=b+c;
        }
        else if(a>b&&b<c)
        {
            d=a+b;
        }
        return d;
    }
}

```

## 89. Decimal to Binary Conversion

Write a Program that accepts a decimal number n, and converts the number to binary. Include a class UserMainCode with a static method **convertDecimalToBinary** which accepts an integer. The return type is long representing the binary number. Create a Class Main which would be used to accept the input integer and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of single integer.

Output consists of a single long.

Refer sample output for formatting specifications.

### **Sample Input 1:**

5

### **Sample Output 1:**

101

### **MAIN**

```

import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        System.out.println(UserMainCode.convertDecimalToBinary(n));
    }
}

```

```
s.close();
}
}
```

### UserMainCode

```
public class UserMainCode {
    public static long convertDecimalToBinary(int n){
        String s1=Integer.toBinaryString(n);
        long y=Long.parseLong(s1);
    return y;
}
}
```

## 90.String Processing - V

Write a program to read a string and also a number N. Form a new string made up of n repetitions of the last n characters of the String. You may assume that n is between 1 and the length of the string.

Include a class UserMainCode with a static method **returnLastRepeatedCharacters** which

accepts the string and the number n. The return type is the string as per the problem statement.

Create a Class Main which would be used to accept the string and integer and call the static method present in UserMainCode.

#### Input and Output Format:

Input consists of a string and integer.

Output consists of a string.

Refer sample output for formatting specifications.

#### Sample Input 1:

Hello

2

#### Sample Output 1:

lolo

#### Sample Input 2:

Hello

3

#### Sample Output 2:

lloolloo

### MAIN

```
import java.util.Scanner;
public class Main
{
    public static void main(String[] args)
    {
        Scanner s=new Scanner(System.in);
        String s1=s.nextLine();
        int n1=s.nextInt();
        System.out.println(UserMainCode.returnLastRepeatedCharacters(s1,n1));
        s.close();
    }
}
```

```
}
```

## USERMAINCODE

```
public class UserMainCode{
    public static String returnLastRepeatedCharacters(String s1, int n1)
{
StringBuffer sb = new StringBuffer();
for(int i = 0 ; i < n1 ; i++)
sb.append(s1.substring(s1.length()-n1, s1.length()));
return sb.toString();
}
```

## 91.Regular Expression - III

Given a string (s) apply the following rules.

1. String should not begin with a number.

If the condition is satisfied then print TRUE else print FALSE.

Include a class UserMainCode with a static method **validateString** which accepts the string.

The return type is the boolean formed based on rules.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### Input and Output Format:

Input consists of a string.

Output consists of TRUE or FALSE .

Refer sample output for formatting specifications.

### Sample Input 1:

ab2

### Sample Output 1:

TRUE

### Sample Input 2:

72CAB

### Sample Output 2:

FALSE

## MAIN

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args)

    {

        Scanner s=new Scanner(System.in);

        String s1=s.nextLine();
```

```

        if(UserMainCode.validateString(s1)==true)
System.out.println("TRUE");

        else System.out.println("FALSE");

        s.close();

    }

}

```

### **USERMAINCODE**

```

public class UserMainCode {
    public static boolean validateString(String s)
    {
        boolean b=false;
        if(s.charAt(0)=='0'||s.charAt(0)=='1'||s.charAt(0)=='2'||s.charAt(0)=='3'||s.charAt(0)=='4'||s.charAt(0)=='5'||s.charAt(0)=='6'||s.charAt(0)=='7'||s.charAt(0)=='8'||s.charAt(0)=='9'){
            b=false;
        }
        else
            b=true;
        return b;
    }
}

```

## **92.String Processing - TrimCat**

Write a program to read a string and return a new string which is made of every alternate characters starting with the first character. For example NewYork will generate Nwok, and Samurai will generate Smri.

Include a class UserMainCode with a static method getAlternateChars which accepts the string. The return type is the modified string.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

#### **Input and Output Format:**

Input consists of a string.

Output consists of string.

Refer sample output for formatting specifications.

#### **Sample Input 1:**

Hello

#### **Sample Output 1:**

Hlo

## **MAIN**

```

import java.util.Scanner;
public class Main
{
    public static void main(String[] args)

```

```

{
    Scanner s=new Scanner(System.in);
String s1=s.nextLine();
System.out.println(UserMainCode.getAlternateChars(s1));
s.close();
}
}

```

#### **USERMAINCODE**

```

public class UserMainCode{
public static String getAlternateChars(String s)
{
//String s1=s.replaceAll(" ", "");
StringBuffer sbf = new StringBuffer();
for(int i = 0; i < s.length() ; i=i+2)
{
sbf.append(s.charAt(i));
}
String str = sbf.toString();
return str;
}
}

```

### **93. String Processing - Username**

Write a program to read a valid email id and extract the username.

Note - user name is the string appearing before @ symbol.

Include a class UserMainCode with a static method fetchUserName which accepts the string. The return type is the modified string.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

#### **Input and Output Format:**

Input consists of a string.

Output consists of string.

Refer sample output for formatting specifications.

#### **Sample Input 1:**

[admin@xyz.com](mailto:admin@xyz.com)

#### **Sample Output 1:**

admin

### **MAIN CLASS**

```

import java.util.Scanner;

public class Main {
public static void main(String[] args) {
Scanner s=new Scanner(System.in);
String s1=s.nextLine();
System.out.println(UserMainCode.fetchUserName(s1));
s.close();
}
}

```

#### **USERMAINCODE**

```

import java.util.StringTokenizer;
public class UserMainCode {
    public static String fetchUserName(String s1) {
        StringTokenizer st=new StringTokenizer(s1,"@");
        String s2=st.nextToken();
        return(s2);
    }
}

```

## 94. String Processing - VII

Write a program to read a two strings and one int value(N). check if Nth character of first String from start and Nth character of second String from end are same or not. If both are same return true else return false.

Check need not be Case sensitive

Include a class UserMainCode with a static method **isEqual** which accepts the two strings and a integer n. The return type is the TRUE / FALSE.

Create a Class Main which would be used to read the strings and integer and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of two strings and an integer.

Output consists of TRUE / FALSE .

Refer sample output for formatting specifications.

### **Sample Input 1:**

AAAA

abab

2

### **Sample Output 1:**

TRUE

### **Sample Input 2:**

MNOP

QRST

3

### **Sample Output 2:**

FALSE

MAIN

```

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

Scanner s=new Scanner(System.in);

String s1=s.nextLine();

String s2=s.nextLine();

int n=s.nextInt();
}

```

```
boolean output=UserMainCode.isEqual(s1,s2,n);

System.out.println(output);

s.close();

}

}

USERMAINCODE

public class UserMainCode {

public static boolean isEqual(String s1,String s2,int n){

boolean a=false;

if(n<s1.length()&&n<s2.length())

{

char c=s1.charAt(n);

char d=s2.charAt(s2.length()-n);

String s3=Character.toString(c);

//System.out.println(s3);

String s4=Character.toString(d);

//System.out.println(s4);

if(s3.equalsIgnoreCase(s4))

{

a=true;

}
}
```

```
}

else

{

a=false;

}

}

return a;

}

}
```

## 95. Largest Difference

Write a program to read a integer array, find the largest difference between adjacent elements and display the index of largest difference.

### EXAMPLE:

input1: {2,4,5,1,9,3,8}

output1: 4 (here largest difference 9-1=8 then return index of 9 ie,4)

Include a class UserMainCode with a static method **checkDifference** which accepts the integer array. The return type is integer.

Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.

### Input and Output Format:

Input consists of an integer n which is the number of elements followed by n integer values.

Output consists of integer.

Refer sample output for formatting specifications.

### Sample Input 1:

```
7
2
4
5
1
9
3
8
```

### Sample Output 1:

```
4
```

### MAIN CLASS

```

import java.util.Scanner;

public class Main{
public static void main(String[] args) {
Scanner s=new Scanner(System.in);
int m=s.nextInt();
int[] n1=new int[m];
for(int i=0;i<m;i++){
    n1[i]=s.nextInt();
}
System.out.println(UserMainCode.checkDifference(n1));
s.close();
}
}

```

### USERMAIN CODE

```

public class UserMainCode {
public static int checkDifference(int[] n1){
int n2,n3=0,n4=0,i;
for(i=0;i<n1.length-1;i++){
n2=Math.abs(n1[i]-n1[i+1]);
if(n2>n3){
n3=n2;
n4=i+1; }}
return n4;
}
}

```

### 1.Start Case

Write a program to read a sentence in string variable and convert the first letter of each word to capital case. Print the final string.

Note: - Only the first letter in each word should be in capital case in final string.

Include a class **UserMainCode** with a static method **printCapitalized** which accepts a string.

The return type (String) should return the capitalized string.

Create a Class Main which would be used to accept a string and call the static method present in UserMainCode.

#### **Input and Output Format:**

Input consists of a strings.

Output consists of a String (capitalized string).

Refer sample output for formatting specifications.

#### **Sample Input:**

Now is the time to act!

#### **Sample Output:**

Now Is The Time To Act!

### MAIN CLASS

```

import java.util.Scanner;

```

```

public class Main {
    public static void main(String[] args){
        Scanner s=new Scanner(System.in);
        String s1=s.nextLine();
        System.out.println(UserMainCode.printCapitalized(s1));
        s.close();
    }
}

```

## USERMAIN CODE

```

import java.util.StringTokenizer;

public class UserMainCode{
    public static String printCapitalized(String s1){
        StringBuffer sb=new StringBuffer();
        StringTokenizer t=new StringTokenizer(s1, " ");
        while(t.hasMoreTokens()){
            String s2=t.nextToken();
            String s3=s2.substring(0,1);
            String s4=s2.substring(1, s2.length());
            sb.append(s3.toUpperCase()).append(s4).append(" ");
        }
        return sb.toString();
    }
}

```

## 2. Maximum Difference

Write a program to read an integer array and find the index of larger number of the two adjacent numbers with largest difference. Print the index.

Include a class **UserMainCode** with a static method **findMaxDistance** which accepts an integer array and the number of elements in the array. The return type (Integer) should return index.

Create a Class Main which would be used to accept an integer array and call the static method present in UserMainCode.

### Input and Output Format:

Input consists of  $n+1$  integers, where  $n$  corresponds the size of the array followed by  $n$  integers. Output consists of an Integer (index).

Refer sample output for formatting specifications.

### Sample Input :

```

6
4
8
6
1
9
4

```

### Sample Output :

```

4

```

[In the sequence 4 8 6 1 9 4 the maximum distance is 8 (between 1 and 9). The function should return the index of the greatest of two. In this case it is 9 (which is at index 4). output = 4.]

### Main:

```

import java.util.Scanner;
public class Main {
    public static void main(String[] args) {Scanner s=new Scanner(System.in);
    int n=s.nextInt();
    int a[]={};
    for(int i=0;i<n;i++)
    {
        a[i]=s.nextInt();
    }
    int max=UserMainCode.findMaxDistance(a);
    System.out.println(max);
    s.close();
}
}

UserMainCode:
public class UserMainCode {
    static int findMaxDistance(int[] a)
    {

        int max=0,index=0;
        for(int i=0;i<19;i++)
        {
            int d=Math.abs(a[i]-a[i+1]);
            if(d>max)
            {
                max=d;
                if(a[i]>a[i+1])
                {
                    index=i;
                }
                else
                {
                    index=i+1;
                }
            }
        }
        return index;
    }
}

```

### 3. Palindrome - In Range

Write a program to input two integers, which corresponds to the lower limit and upper limit respectively, and find the sum of all palindrome numbers present in the range including the two numbers. Print the sum.

Include a class **UserMainCode** with a static method **addPalindromes** which accepts two integers. The return type (Integer) should return the sum if the palindromes are present, else return 0.

Create a Class Main which would be used to accept two integer and call the static method present in UserMainCode.

Note1 : A palindrome number is a number which remains same after reversing its digits.

Note2 : A single digit number is not considered as palindrome.

**Input and Output Format:**

Input consists of 2 integers, which corresponds to the lower limit and upper limit respectively.

Output consists of an Integer (sum of palindromes).

Refer sample output for formatting specifications.

### Sample Input :

130

150

### Sample Output :

272

**(131+141 = 272)**

### Main:

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        int n1=s.nextInt();
        int n2=s.nextInt();
        System.out.println(UserMainCode.addPalindromes(n1,n2));
        s.close();
    }
}
```

### UserMainCode:

```
public class UserMainCode {
    public static int addPalindromes(int n1,int n2){
        int sum=0;
        for(int i=n1;i<=n2;i++){
            int r=0,n3=i;
            while(n3!=0){
                r=(r*10)+(n3%10);
                n3=n3/10;
            }
            if(r==i)
                sum=sum+i;
        }
        return sum;
    }
}
```

## 4. PAN Card

Write a program to read a string and validate PAN no. against following rules:

1. There must be eight characters.
2. First three letters must be alphabets followed by four digit number and ends with alphabet
3. All alphabets should be in capital case.

Print "Valid" if the PAN no. is valid, else print "Invalid".

Include a class **UserMainCode** with a static method **validatePAN** which accepts a string.

The

return type (Integer) should return 1 if the string is a valid PAN no. else return 2.

Create a Class Main which would be used to accept a string and call the static method

present in UserMainCode.

**Input and Output Format:**

Input consists of a string, which corresponds to the PAN number.

Output consists of a string - "Valid" or "Invalid"

Refer sample output for formatting specifications.

**Sample Input 1:**

ALD3245E

**Sample Output 1:**

Valid

**Sample Input 2:**

OLE124F

**Sample Output 2:**

Invalid

**Main:**

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.nextLine();
        UserMainCode.validatePAN(s1);
        s.close();
    }
}
```

**UserMainCode:**

```
public class UserMainCode {public static void validatePAN(String s1) {
    if(s1.matches("[A-Z]{3}[0-9]{4}[A-Z]{1}"))
    {
        System.out.println("Valid");
    }
    else
        System.out.println("Invalid");
}}
```

## 5. Fibonacci Sum

Write a program to read an integer n, generate fibonacci series and calculate the sum of first n numbers in the series. Print the sum.

Include a class **UserMainCode** with a static method **getSumOfNfibos** which accepts an integer n. The return type (Integer) should return the sum of n fibonacci numbers.

Create a Class Main which would be used to accept an integer and call the static method present in UserMainCode.

**Note:** First two numbers in a Fibonacci series are 0, 1 and all other subsequent numbers are sum of its previous two numbers. Example - 0, 1, 1, 2, 3, 5...

**Input and Output Format:**

Input consists of an integer, which corresponds to n.

Output consists of an Integer (sum of fibonacci numbers).

Refer sample output for formatting specifications.

**Sample Input :**

5

**Sample Output :**

7

[0 + 1 + 1 + 2 + 3 = 7]

**Main:**

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        System.out.println(UserMainCode.getSumOfNfibos(n));
        s.close();
    }
}
```

**UserMainCode:**

```
public class UserMainCode {
    public static int getSumOfNfibos(int n){
        int a=-1,b=1,c=0,d=0;
    for(int i=0;i<n;i++)
    {
        c=a+b;
        d=d+c;
        a=b;
        b=c;
    }
    return d;
}
}
```

## 6. Test Vowels

Write a program to read a string and check if given string contains exactly five vowels in any order. Print “Yes” if the condition satisfies, else print “No”.

Assume there is no repetition of any vowel in the given string and all characters are lowercase.

Include a class **UserMainCode** with a static method **testVowels** which accepts a string.

The

return type (Integer) should return 1 if all vowels are present, else return 2.

Create a Class Main which would be used to accept a string and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of a string.

Output consists of a string (“Yes” or “No”).

Refer sample output for formatting specifications.

**Sample Input 1:**

acbisouzze

**Sample Output 1:**

Yes

**Sample Input 2:**

cbisouzze

**Sample Output 2:**

No

**Main:**

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.nextLine();
        int b=UserMainCode.testVowels(s1);
        if(b==1)
            System.out.println("Yes");
        else
            System.out.println("No");
        s.close();
    }
}
```

**UserMainCode:**

```
public class UserMainCode {

    public static int testVowels(String s1) {
        int b;
        int n1=0,n2=0,n3=0,n4=0,n5=0;
        String s2=s1.toLowerCase();
        for(int i=0;i<s2.length();i++){
            char c=s2.charAt(i);
            if(c=='a')
                n1++;
            if(c=='e')
                n2++;
            if(c=='i')
                n3++;
            if(c=='o')
                n4++;
            if(c=='u')
                n5++;
            if(n1==1&&n2==1&&n3==1&&n4==1&&n5==1)
                b=1;
            else b=2;
            return b;
        }

    }
}
```

**7.Dash Check**

Write a program to read two strings and check whether or not they have dashes in the same places. Print “Yes” if the condition satisfies, else print “No”. Include a class **UserMainCode** with a static method **compareDashes** which accepts two strings. The return type (Integer) should return 1 if all dashes are placed correctly, else return 2.

Create a Class Main which would be used to accept two strings and call the static method present in UserMainCode.

**Note:** The strings must have exactly the same number of dashes in exactly the same positions. The strings might be of different length.

**Input and Output Format:**

Input consists of two strings.

Output consists of a string ("Yes" or "No").

Refer sample output for formatting specifications.

**Sample Input 1:**

hi—there-you.

12--(134)-7539

**Sample Output 1:**

Yes

**Sample Input 2:**

-15-389

-xyw-zzy

**Sample Output 2:**

No

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.nextLine();
        String s2=s.nextLine();
        int p=UserMainCode.compareDashes(s1,s2);
        if(p==1)
            System.out.println("Yes");
        else
            System.out.println("No");
        s.close();
    }
}

import java.util.ArrayList;

public class UserMainCode {
    public static int compareDashes(String s1, String s2) {
        ArrayList<Integer>l1=new ArrayList<Integer>();
        for(int i=0;i<s1.length();i++)
        {
            if(s1.charAt(i)=='-')
            {
                l1.add(i);
            }
        }
        ArrayList<Integer>l2=new ArrayList<Integer>();
        for(int i=0;i<s2.length();i++)
        {
            if(s2.charAt(i)=='-')
            {
                l2.add(i);
            }
        }
        //System.out.println(l1);
        //System.out.println(l2);
        if(l1.equals(l2))
```

```

{
    return 1;
}
else
    return 2;
}
}

```

## 8.Reverse Split

Write a program to read a string and a character, and reverse the string and convert it in a format such that each character is separated by the given character. Print the final string. Include a class **UserMainCode** with a static method **reshape** which accepts a string and a character. The return type (String) should return the final string.

Create a Class Main which would be used to accept a string and a character, and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string and a character.

Output consists of a string (the final string).

Refer sample output for formatting specifications.

### **Sample Input:**

Rabbit

### **Sample Output:**

t-i-b-b-a-R

```

import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.nextLine();
        String s2=s.next();
        System.out.println(UserMainCode.reshape(s1,s2));
        s.close();
    }
}

public class UserMainCode {
    public static String reShape(String s,String s1){
        StringBuffer sb=new StringBuffer(s);
        StringBuffer sb2=new StringBuffer();
        String s2=sb.reverse().toString();
        for(int i=0;i<s2.length();i++)
        {
            sb2.append(s2.charAt(i));
            sb2.append(s1);
        }
        sb2.deleteCharAt(sb2.length()-1);
        //System.out.println(sb2.toString());
        return sb2.toString();
    }
}

```

## 9.Remove 10's

Write a program to read an integer array and remove all 10s from the array, shift the other elements towards left and fill the trailing empty positions by 0 so that the modified array is of the same length of the given array.

Include a class **UserMainCode** with a static method **removeTens** which accepts the number

of elements and an integer array. The return type (Integer array) should return the final array.

Create a Class Main which would be used to read the number of elements and the input array, and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of  $n+1$  integers, where  $n$  corresponds to size of the array followed by  $n$  elements of the array.

Output consists of an integer array (the final array).

Refer sample output for formatting specifications.

**Sample Input :**

```
5
1
10
20
10
2
```

**Sample Output :**

```
1
20
```

```
import java.io.*;
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int i;
        int n = sc.nextInt();
        int a[] = new int[n];
        for (i = 0; i < n; i++) {
            a[i] = sc.nextInt();
        }
        UserMainCode.removeTens(a);
        sc.close();
    }
}
```

```

}

import java.util.*;
public class UserMainCode {
public static void removeTens(int a[]){
    Scanner sc = new Scanner(System.in);

    int i,k = 0;
    int b[] = new int[a.length];
    ArrayList<Integer> al = new ArrayList<Integer>();
    for (i = 0; i <a.length; i++) {
        if (a[i] != 10) {
            al.add(a[i]);
        }
    }
    if (al.size() < a.length) {
        k = a.length- al.size();
        for (i = 0; i < k; i++) {
            al.add(0);
        }
    }
    int b1[] = new int[a.length];
    for (i = 0; i < a.length; i++) {
        b1[i] = al.get(i);
        System.out.println(b1[i]);
    }
}
}

```

## 10.Last Letters

Write a program to read a sentence as a string and store only the last letter of each word of

the sentence in capital letters separated by \$. Print the final string.

Include a class **UserMainCode** with a static method **getLastLetter** which accepts a string.

The return type (string) should return the final string.

Create a Class Main which would be used to read a string, and call the static method present

in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of a string (the final string).

Refer sample output for formatting specifications.

### **Smaple Input :**

This is a cat

### **Sample Output :**

\$\$\$\$A\$T

**Main:**

```
public class Main {
```

```

public static void main(String[] args) {
    Scanner s=new Scanner(System.in);
    String input=s.nextLine();
    System.out.println(UserMainCode.getLastLetter(input));
}

}

UserMainCode:

import java.util.*;
public class UserMainCode {
    public static String getLastLetter(String input){
        String str1=null;
        StringTokenizer st=new StringTokenizer(input, " ");
        StringBuffer sb=new StringBuffer();
        while(st.hasMoreTokens()){
            str1=st.nextToken();
            // String str2=Character.toString(str1.charAt(str1.length()-1));
            String str2=str1.substring(str1.length()-1);
            String str3= str2.toUpperCase();
            sb.append(str3).append("$");
        }sb.deleteCharAt(sb.length()-1);
        return sb.toString();
    }
}

```

## 11.Largest Key in HashMap

Write a program that constructs a hashmap and returns the value corresponding to the largest key.

Include a class UserMainCode with a static method **getMaxKeyValue** which accepts a string.

The return type (String) should be the value corresponding to the largest key.

Create a Class Main which would be used to accept Input string and call the static method

present in UserMainCode.

### **Input and Output Format:**

Input consists of  $2n+1$  values. The first value corresponds to size of the hashmap. The next  $n$

pair of numbers equals the integer key and value as string.

Output consists of a string which is the value of largest key.

Refer sample output for formatting specifications.

### **Sample Input 1:**

```

3
12
amron
9
Exide
7

```

SF

### Sample Output 1:

Amron

### Main:

```
import java.util.*;
public class Main {
    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        HashMap<Integer, String> hm=new HashMap<Integer, String>();
        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        for(int i=0;i<n;i++)
        {
            int a=s.nextInt();
            String s1=s.next();
            hm.put(a,s1);

        }
        System.out.println(UserMainnode.getMaxKeyValue(hm));
    }
}
```

### Hashmap:

```
import java.util.*;
import java.util.HashMap;
import java.util.Iterator;
public class UserMainnode {
    public static String getMaxKeyValue(HashMap<Integer, String> hm) {
        int max=0;
        String s3=null;
        Iterator<Integer> itr=hm.keySet().iterator();
        while(itr.hasNext())
        {
            int b=itr.next();
            if(b>max)
            {
                max=b;
                s3=hm.get(b);
            }
        }
        return (s3);
    }
}
```

### 12.All Numbers

Write a program to read a string array and return 1 if all the elements of the array are

numbers, else return -1.

Include a class UserMainCode with a static method **validateNumber** which accepts a string

array. The return type (integer) should be -1 or 1 based on the above rules.

Create a Class Main which would be used to accept Input string array and call the static method present in UserMainCode.

The string array is said to be valid if all the elements in the array are numbers. Else it is invalid.

### **Input and Output Format:**

Input consists of an integer specifying the size of string array followed by n strings.

Refer sample output for formatting specifications.

### **Sample Input 1:**

```
4  
123  
24.5  
23  
one
```

### **Sample Output 1:**

invalid

### **Sample Input 2:**

```
2  
123  
24.5
```

### **Sample Output 2:**

valid

### **Main:**

```
import java.util.*;  
public class Main {  
    public static void main(String[] args) {  
        Scanner s = new Scanner(System.in);  
        int n = s.nextInt();  
        String[] s1 = new String[n];  
        for(int i=0;i<n;i++){  
            s1[i] = s.next();  
        }  
  
        int out=(userMainCode.validateNumber(s1));  
        System.out.println(out);  
  
    }  
}
```

### **UserMainCode:**

```
class userMainCode{  
    public static int validateNumber(String[] s1){  
        int b =0 ,count,out=0;  
        for(int i=0;i<s1.length;i++){  
            String s2 = s1[i];  
            if(s2.matches("[0-9]{1,}"))
```

```

    {
        count =0;
        for(int j=0;j<s2.length();j++)
        {
            char c = s2.charAt(j);
            if(c=='.')
                count++;
        }
        if(count>1)
            b=1;
    }
    else
        b=1;
}
if(b==0){
    out=1;
}
else out=-1;
return out;
}
}

```

### 13.Day of the Week

Write a program to read a date as string (MM-dd-yyyy) and return the day of week on that date.

Include a class UserMainCode with a static method **getDay** which accepts the string. The

return type (string) should be the day of the week.

Create a Class Main which would be used to accept Input string and call the static method

present in UserMainCode.

#### **Input and Output Format:**

Input consists of a string.

Output consists of a string.

Refer sample output for formatting specifications.

#### **Sample Input 1:**

07-13-2012

#### **Main:**

```

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;
public class Main {

    public static void main(String[] args) throws ParseException {
        Scanner sc=new Scanner(System.in);
        String s1=sc.nextLine();
        System.out.println(UserMainCode.getDay(s1));
    }
}

```

```

    // TODO Auto-generated method stub

}

UserMainCode:

import java.text.SimpleDateFormat;
import java.text.ParseException;
import java.util.Date;
public class UserMainCode {

public static String getDay(String s1) throws ParseException
{
SimpleDateFormat sdf=new SimpleDateFormat("MM-dd-yyyy");
SimpleDateFormat sdf1=new SimpleDateFormat("EEEE");
Date d=sdf.parse(s1);
String s=sdf1.format(d);
return s;
}
}

```

## 14.Max Substring

Write a program to accept two string inputs. The first being a source string and second one

a delimiter. The source string contains the delimiter at various locations. Your job is to return the substring with maximum number of characters. If two or more substrings have

maximum number of characters return the substring which appears first. The size of the delimiter is 1.

Include a class UserMainCode with a static method **extractMax** which accepts the string.

The return type (string) should be the max substring.

Create a Class Main which would be used to accept Input string and call the static method

present in UserMainCode.

### **Input and Output Format:**

Input consists of a source string and delimiter.

Output consists of a string.

Refer sample output for formatting specifications.

### **Sample Input 1:**

delhi-pune-patna

### **Sample Output 1:**

Delhi\

### **Main:**

```
import java.util.*;
```

```

public class Main {
    public static void main(String[] args){

        Scanner sc=new Scanner(System.in);
        String input1=sc.next();
        String input2=sc.next();
        System.out.println(UserMainCode.extractMax(input1,input2));

    }
}

Usermaincode:

import java.util.StringTokenizer;
import java.util.*;

public class UserMainCode {
    public static String extractMax(String input1,String input2){
        int max=0;
        String s3=null;
        StringTokenizer st=new StringTokenizer(input1,"-");
        while( st.hasMoreTokens())
        {
            String s2=st.nextToken();
            int n=s2.length();
            if(n>max)
            {
                max=n;
                s3=s2;
            }
        }
        return(s3);
    }
}

```

## 15. States and Capitals

Write a program that constructs a hashmap with “state” as key and “capital” as its value.

If

the next input is a state, then it should return capital\$state in lowercase.

Include a class UserMainCode with a static method **getCapital** which accepts a hashmap.

The return type is the string as given in the above statement

Create a Class Main which would be used to accept Input string and call the static method

present in UserMainCode.

### **Input and Output Format:**

Input consists of  $2n+2$  values. The first value corresponds to size of the hashmap. The next  $n$

pair of numbers contains the state and capital. The last value consists of the “state” input.

Output consists of a string as mentioned in the problem statement.

Refer sample output for formatting specifications.

**Sample Input 1:**

```
3
Karnataka
Bangaluru
Punjab
Chandigarh
Gujarat
Gandhinagar
Punjab
```

**Sample Output 1:**

```
chandigarh$punjab
```

**Main:**

```
import java.util.HashMap;
import java.util.Scanner;
public class Main {

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        HashMap<String, String> hm=new
        HashMap<String, String>();
        for(int i=0;i<n;i++)
        {
            String s1=sc.next();
            String s2=sc.next();
            hm.put(s1,s2);
        }
        String sa=sc.next();
        System.out.print(UserMainCode.getCapital(hm,sa));
    }
}
```

**UserMainCode:**

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
public class UserMainCode {
    public static String getCapital(HashMap<String, String>
hm, String sa)
    {
        String chan=null;
        Iterator<String>it=hm.keySet().iterator();
        StringBuffer sb=new StringBuffer();
        while(it.hasNext()){
            String a=it.next();
            if(a.equals(sa))
            {
                chan=hm.get(a);
            }
        }
        return chan;
    }
}
```

```

        sb.append(chan).append("$").append(sa);
    }
}
return sb.toString();
}
}

```

## 16.Simple String Manipulation - II

Write a program to read a string and return an integer based on the following rules.

If the first word and the last word in the String match, then return the number of characters

in the word else return sum of the characters in both words. Assume the Strings to be case - sensitive.

Include a class UserMainCode with a static method **calculateWordSum** which accepts a

string. The return type (integer) should be based on the above rules.

Create a Class Main which would be used to accept Input string and call the static method

present in UserMainCode.

### **Input and Output Format:**

Input consists of a string with maximum size of 100 characters.

Output consists of a string.

Refer sample output for formatting specifications.

### **Sample Input 1:**

COGNIZANT TECHNOLOGY SOLUTIONS COGNIZANT

### **Sample Output 1:**

9

### **Sample Input 2:**

HOW ARE YOU

### **Sample Output 2:**

6

### **Main:**

```

import java.util.*;
public class Main {

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        String inpList=sc.nextLine();
        System.out.println(UserMainCode.calculateWordSum(inpList));
    }
}

```

### **UserMainCode:**

```
import java.util.*;
```

```

public class UserMainCode {
    public static int calculateWordSum(String inp) {

        int count=0;
        String st[]=inp.split(" ");
        String s1=st[0];
        String slst=st[st.length-1];
        if(s1.equals(slst))
        {
            count=s1.length();
        }
        else
        {
            count=s1.length()+slst.length();
        }
        return count;
    }
}

```

## 17.Vowels, Arrays & ArrayLists

Write a program to read an array of strings and return an arraylist which consists of words

whose both first and last characters are vowels. Assume all inputs are in lowercase.

Include a class UserMainCode with a static method **matchCharacter** which accepts a string

array. The return type shoud be an arraylist which should contain elements as mentioned

above.

Create a Class Main which would be used to accept Input array and call the static method

present in UserMainCode.

### **Input and Output Format:**

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in

the array. The next 'n' string correspond to the elements in the array.

Output consists of strings which are elements of arraylist

Refer sample output for formatting specifications.

### **Sample Input 1:**

```

4
abcde
pqrs
abci
orto

```

### **Sample Output 1:**

```

abcde
abci
orto

```

**Main:**

```
package vowels;
import java.util.*;

public class Main {

    public static void main(String[] args) {

        int n;
        Scanner sc=new Scanner(System.in);
        n=Integer.parseInt(sc.nextLine());
        String[] str=new String[n];
        for(int i=0;i<n;i++)
        {
            str[i]=sc.nextLine();
        }
        ArrayList<String> arr=new ArrayList<String>();
        arr=UserMainCode.matchCharacter(str);
        Iterator<String> it=arr.iterator();
        while(it.hasNext())
        {
            System.out.println(it.next());
        }

    }
}
```

**Usermaincode:**

```
package vowels;
import java.util.*;
public class UserMainCode {

    public static ArrayList<String> matchCharacter (String[] ss)
    {
        ArrayList<String> as=new ArrayList<String>();
        for(int i=0;i<ss.length;i++)
        {
            String sp=ss[i];
            char[] mp=sp.toLowerCase().toCharArray();
            if((mp[0]=='a'||mp[0]=='e'||mp[0]=='i'||mp[0]=='o'||mp[0]=='u')&
&(mp[sp.length()-1]=='a'||mp[sp.length()-1]=='e'||mp[sp.length()-
1]=='i'||mp[sp.length()-
1]=='o'||mp[sp.length()-1]=='u'))
            {
                as.add(sp);
            }
        }
        return as;
    }
}
```

## 18.Transfer from Hashmap to ArrayList

Write a program that constructs a hashmap with “employee id” as key and “name” as its value. Based on the rules below, on being satisfied, the name must be added to the arraylist.

i)First character should be small and the last character should be Capital.

ii)In name at least one digit should be there.

Include a class UserMainCode with a static method **getName** which accepts a hashmap.

The

return type is an arraylist as expected in the above statement

Create a Class Main which would be used to accept Input string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of  $2n+1$  values. The first value corresponds to size of the hashmap. The next  $n$  pair of numbers contains the employee id and name.

Output consists of arraylist of strings as mentioned in the problem statement.

Refer sample output for formatting specifications.

### **Sample Input 1:**

4

1

ravi5raJ

2

sita8gitA

3

ram8sitA

4

rahul

### **Sample Output 1:**

ravi5raJ

sita8gitA

ram8sitA

### **main:**

```
import java.util.*;
import java.text.*;
public class Main {
public static void main(String[] args) {
    HashMap<Integer, String> hm1=new HashMap<Integer, String>();
    int n;
    Scanner sc=new Scanner(System.in);
    n=Integer.parseInt(sc.nextLine());
    for(int i=0;i<n;i++)
    {
        hm1.put(Integer.parseInt(sc.nextLine()),sc.nextLine());
    }
    ArrayList<String> al1=new ArrayList<String>();
    al1=UserMainCode.getName(hm1);
    Iterator<String> it=al1.iterator();
    while(it.hasNext())
    {
```

```

        System.out.println(it.next());
    }

}
}

```

**Usermaincode:**

```

import java.util.*;
import java.text.*;
public class UserMainCode {
    public static ArrayList<String> getName(HashMap<Integer, String> hm1)
    {
        ArrayList<String> al2=new ArrayList<String>();
        Iterator<Integer> it =hm1.keySet().iterator();
        while(it.hasNext())
        {
            int id=it.next();
            String name=hm1.get(id);
            if(name.matches("[a-z]{1,}.*[0-9]{1,}.*[A-Z]{1}"))
                al2.add(name);
        }
        return al2;
    }
}

```

## 19.Max Admissions

Write a program that reads details about number of admissions per year of a particular college, return the year which had maximum admissions. The details are stored in an arraylist with the first index being year and next being admissions count.

Include a class UserMainCode with a static method **getYear** which accepts a arraylist. The return type is an integer indicating the year of max admissions.

Create a Class Main which would be used to accept Input string and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of  $2n+1$  values. The first value corresponds to size of the data (year & admissions). The next  $n$  pair of numbers contains the year and admissions count.

Output consists of an integer as mentioned in the problem statement.

Refer sample output for formatting specifications.

**Sample Input 1:**

```

4
2010
200000
2011
300000
2012
45000
2013
25000

```

**Sample Output 1:**

2011

USERMAINCODE:

```
import java.util.ArrayList;

public class UserMainCode

{

public static int year (ArrayList<Integer> a1)

{

int max=0,pos=0;

for(int i=1;i<a1.size();i+=2)

{

if(a1.get(i)>max)

{

max=a1.get(i);

pos=i;

}

}

return a1.get(pos-1);

}

}
```

MAIN:

```
import java.util.*;

class Main

{

public static void main(String [] args)

{

Scanner s=new Scanner(System.in);

ArrayList<Integer> a1=new ArrayList<Integer>();
```

```

int n=s.nextInt();

n=n*2;

for(int i=0;i<n;i++)

{

a1.add(s.nextInt());

}

System.out.println(UserMainCode.year(a1));

s.close();

}

}

```

## 20.Sum Non Prime Numbers

Write a program to calculate the sum of all the non prime positive numbers less than or equal to the given number.

Note: prime is a natural number greater than 1 that has no positive divisors other than 1 and itself

Example:

input = 9

Prime numbers = 2,3,5 and 7

output = 1+4+6+8+9=28

Include a class **UserMainCode** with a static method “**addNumbers**” that accepts an integer

arguement and returns an integer.

Create a class **Main** which would get an integer as input and call the static method **validateNumber** present in the UserMainCode.

**Input and Output Format:**

Input consists of an integer.

Output consists of an integer.

**Sample Input:**

9

**Sample Output:**

28

**Main:**

```

import java.util.*;

public class Main {
    public static void main(String[] args) {
    {
        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        System.out.println(UserMainCode.addNumbers(n));
    }
}

```

```

        }
    }

Usermaincode:

public class UserMainCode {
    public static int addNumbers(int n) {
        int sum=0;int k=0;int sum1=0;
        for(int i=1; i<=n; i++)
        { k=0;
        for(int j=1; j<=i; j++)
        {
        if(i%j==0)
        k++;
        }
        if(k!=2)
        {
        sum=sum+i;
        }
        }
        return sum;
    }
}

```

## 21.Date Format Conversion

Given a date string in the format dd/mm/yyyy, write a program to convert the given date to the format dd-mm-yy.

Include a class **UserMainCode** with a static method “**convertDateFormat**” that accepts a

String and returns a String.

Create a class **Main** which would get a String as input and call the static method **convertDateFormat** present in the UserMainCode.

### **Input and Output Format:**

Input consists of a String.

Output consists of a String.

### **Sample Input:**

12/11/1998

### **Sample Output:**

12-11-98

### **Main:**

```

import java.util.*;
import java.text.*;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.next();
        UserMainCode.convertDateFormat(s1);
    }
}

```

### Usermaincode:

```

import java.util.*;
import java.text.*;
public class UserMainCode {
    public static void convertDateFormate(String s1) {
        SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
        sdf.setLenient(false);
        try {
            Date d1=sdf.parse(s1);
            SimpleDateFormat sdf1=new SimpleDateFormat("dd-MM-yy");
            String s2=sdf1.format(d1);
            System.out.println(s2);
        } catch (ParseException e) {
            e.printStackTrace();
        }
    }
}

```

## 22.Valid Date

Given a date string as input, write a program to validate if the given date is in any of the following formats:

dd.mm.yyyy  
dd/mm/yy  
dd-mm-yyyy

Include a class **UserMainCode** with a static method “**validateDate**” that accepts a String and

returns an integer. This method returns 1 if the date is valid, else return -1.

Create a class **Main** which would get a String as input and call the static method **validateDate** present in the UserMainCode.

### **Input and Output Format:**

Input consists of a String.

Output consists of a String that is either 'Valid' or 'Invalid'.

### **Sample Input 1:**

12.03.2012

### **Sample Output 1:**

Valid

### **Sample Input 2:**

27#01#1977

### **Sample Output 2:**

Invalid

### **UserMainCode:**

```

public class UserMainCode

{
    public static int dateformat(String s1) throws ParseException
    {
        String s2="";

```

```
int n=-1;

if(s1.matches("[0-9]{2}[.]{1}[0-9]{2}[.]{1}[0-9]{4}"))

{

SimpleDateFormat sdf=new SimpleDateFormat("dd.MM.yyyy");

Date d=sdf.parse(s1);

s2=sdf.format(d);

n=1;

}

else if(s1.matches("[0-9]{2}[/]{1}[0-9]{2}[/]{1}[0-9]{2}"))

{

SimpleDateFormat sdf1=new SimpleDateFormat("dd/MM/yy");

Date d1=sdf1.parse(s1);

s2=sdf1.format(d1);

n=1;

}

else if(s1.matches("[0-9]{2}[-]{1}[0-9]{2}[-]{1}[0-9]{4}"))

{

SimpleDateFormat sdf2=new SimpleDateFormat("dd-MM-yyyy");

Date d2=sdf2.parse(s1);

s2=sdf2.format(d2);

n=1;

}

else

{

n=-1;

}

return n;
```

```

}

}

MAIN:

import java.text.ParseException;

import java.util.*;

class Main

{

public static void main(String [] args) throws ParseException

{

Scanner s=new Scanner(System.in);

String s1=s.next();

int b=UserMainCode.dateFormat(s1);

if(b==1)

{



System.out.println("Valid");

}

Else

{

System.out.println("Invalid");

}

s.close();

}

}

```

## 23.Convert Format

Given a 10 digit positive number in the format XXX-XXX-XXXX as a string input, write a program to convert this number to the format XX-XX-XXX-XXX.

Include a class **UserMainCode** with a static method “**convertFormat**” that accepts a String

argument and returns a String.

Create a class **Main** which would get a String as input and call the static method **convertFormat** present in the UserMainCode.

**Input and Output Format:**

Input consists of a String.

Output consists of a String.

**Sample Input:**

555-666-1234

**Sample Output:**

55-56-661-234

**Main:**

```
import java.util.*;
import java.text.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        String s=sc.next();
        System.out.println(UserMainCode.convertFormate(s));
    }
}
```

**Usermaincode:**

```
import java.util.*;
import java.text.*;
public class UserMainCode {
    public static String convertFormate(String s) {
        StringTokenizer t=new StringTokenizer(s,"-");
        String s1=t.nextToken();
        String s2=t.nextToken();
        String s3=t.nextToken();
        StringBuffer sb=new StringBuffer();
        sb.append(s1.substring(0, s1.length()-1)).append('-');
        sb.append(s1.charAt(s1.length()-1)).append(s2.charAt(0)).append('-');
        sb.append(s2.substring(1, s2.length()))).append(s3.charAt(0)).append('-');
        sb.append(s3.substring(1, s3.length())));
        return sb.toString();
    }
}
```

## 24.Add and Reverse

Given an int array and a number as input, write a program to add all the elements in the array greater than the given number. Finally reverse the digits of the obtained sum and print

it.

Include a class **UserMainCode** with a static method “**addAndReverse**” that accepts 2 arguments and returns an integer. The first argument corresponds to the integer array and the second argument corresponds to the number.

Create a class **Main** which would get the required input and call the static method **addAndReverse** present in the UserMainCode.

**Example:**

Input Array = {10,15,20,25,30,100}

Number = 15

sum = 20 + 25 + 30 + 100 = 175

output = 571

**Input and Output Format:**

The first line of the input consists of an integer that corresponds to the number of elements in the array.

The next n lines of the input consists of integers that correspond to the elements in the array.

The last line of the input consists of an integer that corresponds to the number.

Output consists of a single integer.

**Sample Input**

```
6
10
15
20
25
30
100
15
```

**Sample Output**

```
571
```

**Main:**

```
import java.util.*;

public class Main {

    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        int a[]=new int[n];
        for(int i=0;i<n;i++)
        {
            a[i]=sc.nextInt();
        }
        int b=sc.nextInt();
        System.out.println(UserMainCode.addAndReverse(n,b,a)) ;
        sc.close();
    }
}

Usermaincode:
import java.util.*;
public class UserMainCode {
```

```

public static int addAndReverse(int n,int b,int a[])
{
    int i=0,sum=0,r=0;
    for(i=0;i<a.length;i++)
    {
        if(a[i]>b)
        {
            sum=sum+a[i];
        }
    }
    System.out.println(sum);
    while(sum!=0)
    {
        r=((r*10)+(sum%10));
        sum=sum/10;
    }
    return r;
}
}

```

## 25.Next Year day

Given a date string in dd/mm/yyyy format, write a program to calculate the day which falls on the same date next year. Print the output in small case.

The days are sunday, monday, tuesday, wednesday, thursday, friday and saturday.

Include a class **UserMainCode** with a static method “**nextYearDay**” that accepts a String and

returns a String.

Create a class **Main** which would get a String as input and call the static method **nextYearDay** present in the UserMainCode.

### **Input and Output Format:**

Input consists of a String.

Output consists of a String.

### **Sample Input:**

13/07/2012

### **Sample Output:**

Saturday

#### **Main:**

```

import java.util.*;
import java.text.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        String s1=sc.next();
        UserMainCode u=new UserMainCode();
        {
            System.out.println(u.nextYearDay(s1));
        }
    }
}

```

#### **Usercodemain:**

```

import java.util.*;
import java.text.*;

```

```

public class UserMainCode
{
    public String nextYearDay(String s1)
    {
        String s=null;
        SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
        sdf.setLenient(false);
        try {
            Date d1=sdf.parse(s1);
            Calendar cal=Calendar.getInstance();
            cal.setTime(d1);
            cal.add(Calendar.YEAR, 1);
            Date d2=cal.getTime();
            SimpleDateFormat sdf1=new SimpleDateFormat("EEEE");
            s=sdf1.format(d2);
        }
        catch (ParseException e)
        {
            e.printStackTrace();
        }
        return s;
    }
}

```

## 26.Sum Squares of Digits

Write a program that accepts a positive number as input and calculates the sum of squares of individual digits of the given number.

Include a class **UserMainCode** with a static method “**getSumOfSquaresOfDigits**” that accepts an integer argument and returns an integer.

Create a class **Main** which would get an integer as input and call the static method **getSumOfSquaresOfDigits** present in the UserMainCode.

### **Input and Output Format:**

Input consists of an integer.

Output consists of an integer.

### **Sample Input:**

321

### **Sample Output:**

14

### **Main:-**

```

import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        UserMainCode.getSumOfSquaresOfDigits(n);
        s.close();
    }
}

```

### **UserMainCode:-**

```

import java.util.*;

```

```

public class UserMainCode {
    public static void getSumOfSquaresOfDigits(int n) {
        int a=n;
        int rem=0;
        int sum=0;
        while(a!=0)
        {
            rem=a%10;
            sum=sum+(rem*rem);
            a=a/10;
        }
        System.out.println(sum);
    }
}

```

## 27.Even and Odd Index Sum

Write a program that accepts a positive number as input and calculates the sum of digits at even indexes (say evenSum) and sum of digits at odd indexes (say oddSum) in the given number. If both the sums are equal , print 'yes', else print no.

Example:

input = 23050

evenSum =  $2 + 0 + 0 = 2$

oddSum =  $3 + 5 = 8$

output = no

Include a class **UserMainCode** with a static method “**sumOfOddEvenPositioned**” that accepts an integer and returns an integer. The method returns 1 if the 2 sums are equal. Else the method returns -1.

Create a class **Main** which would get an integer as input and call the static method **sumOfOddEvenPositioned** present in the UserMainCode.

### **Input and Output Format:**

Input consists of an integer.

Output consists of a string that is either “yes” or “no”.

### **Sample Input 1:**

23050

### **Sample Output 1:**

no

### **Sample Input 2:**

231

### **Sample Output 2:**

Yes

**Main:-**

```

import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        UserMainCode.sumOfOddEvenPositioned(n);
        sc.close();
    }
}

```

```
}
```

#### UserMainCode:-

```
import java.util.*;
public class UserMainCode {
public static void sumOfOddEvenPositioned(int n) {
int rem = 0, i = 0;
int a[] = new int[10];
while (n > 0) {
rem = n % 10;
a[i] = rem;
n = n / 10;
i++;
}
int sume = 0, sumo = 0;
for (int j = i - 1; j >= 0; j--) {
if(j%2!=0)
{
sumo = sumo + a[j];
}
else
{
sume = sume + a[j];
}
}
if (sume == sumo) {
System.out.println("Yes");
} else
System.out.println("No");
}
}
```

## 28.Remove 3 Multiples

Write a program that accepts an ArrayList of integers as input and removes every 3rd element and prints the final ArrayList.

Suppose the given arrayList contains 10 elements remove the 3rd, 6th and 9th elements.  
Include a class **UserMainCode** with a static method “**removeMultiplesOfThree**” that accepts

an ArrayList<Integer> as argument and returns an ArrayList<Integer>.

Create a class **Main** which would get the required input and call the static method **removeMultiplesOfThree** present in the UserMainCode.

#### Input and Output Format:

The first line of the input consists of an integer n, that corresponds to the number of elements to be added in the ArrayList.

The next n lines consist of integers that correspond to the elements in the ArrayList.

Output consists of an ArrayList of integers.

#### Sample Input:

```
6
3
1
11
```

19

17

19

### Sample Output

3

1

19

17

### Main:-

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Scanner;
public class Main{
public static void main(String []args){
Scanner sc=new Scanner(System.in);
ArrayList<Integer> al=new ArrayList<Integer>();
ArrayList<Integer> al1=new ArrayList<Integer>();
int n=Integer.parseInt(sc.nextLine());
for(int i=0;i<n;i++)
{
al.add(sc.nextInt());
}
al1=UserMainCode.removeMultiplesOfThree(al);
Iterator it=al1.iterator();
while(it.hasNext())
{
System.out.println(it.next());
}
}
}
```

### UserMainCode:-

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.StringTokenizer;
public class UserMainCode
{
public static ArrayList<Integer> removeMultiplesOfThree(ArrayList<Integer> al)
{
ArrayList<Integer> al2=new ArrayList<Integer>();
for(int i=0;i<al.size();i++)
{
if((i+1)%3!=0)
al2.add(al.get(i));
}
return al2;
}
}
```

## 29.String Occurrences - II

Obtain two strings S1,S2 from user as input. Your program should count the number of times S2 appears in S1.

Return the count as output. Note - Consider case.

Include a class UserMainCode with a static method **getSubstring** which accepts two string variables. The return type is the count.

Create a Class Main which would be used to accept two Input strings and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of two strings with maximum size of 100 characters.

Output consists of an integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

catcowcat

cat

### **Sample Output 1:**

2

### **Sample Input 2:**

catcowcat

CAT

### **Sample Output 2:**

0

### **Main:-**

```
MAIN  
import java.util.Scanner;
```

```
public class Main {  
  
    public static void main(String[] args){  
  
        Scanner sc=new Scanner(System.in);  
  
        String s=sc.nextLine();  
  
        String s1=sc.nextLine();  
  
        System.out.println(UserMainCode.getSubstring(s, s1));  
  
        sc.close();  
  
    }  
  
}
```

#### USERMAINCODE

```
public class UserMainCode{  
  
    public static int getSubstring(String s, String s1){  
  
        int t=s1.length();  
  
  
        int count=0;  
  
        for(int i=0;i<s.length()-t+1;i++)  
  
        {  
  
            String s3=s.substring(i,t+i);  
  
            if(s3.equals(s1))  
  
            {  
  
                count++;  
  
            }  
  
        }  
  
        return count;  
  
    }  
  
}
```

## **30. Programming Logic**

Write a Program that accepts three integer values (a,b,c) and returns their sum. However, if one of the values is 13 then it does not count towards the sum and the next number also does not count. So for example, if b is 13, then both b and c do not count.

Include a class UserMainCode with a static method **getLuckySum** which accepts three integers. The return type is integer representing the sum.

Create a Class Main which would be used to accept the input integers and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of three integers.

Output consists of a single integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

1  
2  
3

### **Sample Output 1:**

6

### **Sample Input 2:**

1  
2  
13

### **Sample Output 2:**

3

### **Sample Input 3:**

13

3

8

### Sample Output 3:

8

Main:-

```
import java.util.Scanner;
public class Main{
public static void main(String[] args){
Scanner s=new Scanner(System.in);
int a=s.nextInt();

int b=s.nextInt();
int c=s.nextInt();
System.out.println(UserMainCode.LuckySum(a,b,c));
}
}
```

UserMainCode:-

```
public class UserMainCode{
public static int luckySum(int a, int b, int c)
{
if(a == 13)
return 0;
if(b == 13)
return a;
if(c == 13)
return (a + b);
return (a + b + c);
}
}
```

## 31.Triplets

Given an integer array, Write a program to find if the array has any triplets. A triplet is a value if it appears 3 consecutive times in the array.

Include a class UserMainCode with a static method **checkTriplets** which accepts an integer

array. The return type is boolean stating whether its a triplet or not.

Create a Class Main which would be used to accept the input array and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of  $n+1$  integers. The first integer would represent the size of array and the next  $n$  integers would have the values.

Output consists of a string stating TRUE or FALSE.

Refer sample output for formatting specifications.

**Sample Input 1:**

7

3

3

5

5

5

2

3

**Sample Output 1:**

TRUE

**Sample Input 2:**

7

5

3

5

1

5

2

3

### Sample Output 2:

FALSE

#### Main:-

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        int n;
        Scanner sc=new Scanner(System.in);
        n=sc.nextInt();
        int[] a=new int[n];
        for(int i=0;i<n;i++)
        {
            a[i]=sc.nextInt();
        }
        boolean s=UserMainCode.checkTriples(a);
        if(s==true)
            System.out.println("TRUE");
        else
            System.out.println("FALSE");
    }
}
```

#### UserMainCode:-

```
import java.util.*;

public class UserMainCode {
    public static boolean checkTriples(int[] a)
    {
```

```

boolean b=false;

for(int i=0;i<a.length-2;i++)
{
    if((a[i]==a[i+1])&&(a[i+1]==a[i+2]))
    {
        b=true;
    }
}

return b;
}
}

```

## 32.Repeat Front

Given a string (s) and non negative integer (n) apply the following rules.

1. Display the first three characters as front.
2. If the length of the string is less than 3, then consider the entire string as front and repeat it n times.

Include a class UserMainCode with a static method **repeatFirstThreeCharacters** which accepts the string and integer. The return type is the string formed based on rules.

Create a Class Main which would be used to accept the string and integer and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string and integer.

Output consists of a string .

Refer sample output for formatting specifications.

### **Sample Input 1:**

Coward

2

### **Sample Output 1:**

CowCow

### **Sample Input 2:**

So

3

### **Sample Output 2:**

SoSoSo

**Main:-**

```
import java.util.*;

public class Main {

    public static void main(String []args){

        Scanner sc=new Scanner(System.in);

        String s=sc.nextLine();

        int n=Integer.parseInt(sc.nextLine());

        System.out.println(UserMainCode.repeatFirstThreeCharacters(s,n));

        sc.close();

    }

}
```

**UserMaincode:-**

```
import java.util.*;

public class UserMainCode

{

    public static String repeatFirstThreeCharacters(String s,int n)

{
```

```
StringBuffer sb=new StringBuffer();
StringBuffer sb1=new StringBuffer();
if(s.length()>3)
{ sb.append(s.substring(0,3));
s=sb.toString();
}
for(int i=0;i<n;i++)
sb1.append(s);
return sb1.toString();
}
```

### 33.Sorted Array

Write a program to read a string array, remove duplicate elements and sort the array.

Note:

1. The check for duplicate elements must be case-sensitive. (AA and aa are NOT duplicates)

2. While sorting, words starting with upper case letters takes precedence.

Include a class UserMainCode with a static method **orderElements** which accepts the string

array. The return type is the sorted array.

Create a Class Main which would be used to accept the string array and integer and call the static method present in UserMainCode.

#### **Input and Output Format:**

Input consists of an integer n which is the number of elements followed by n string values.

Output consists of the elements of string array.

Refer sample output for formatting specifications.

**Sample Input 1:**

6

AAA

BBB

AAA

AAA

CCC

CCC

**Sample Output 1:**

AAA

BBB

CCC

**Sample Input 2:**

7

AAA

BBB

aaa

AAA

Abc

A

b

**Sample Output 2:**

A

AAA

Abc

BBB

aaa

b

**Main:-**

```
import java.util.*;

public class Main

{
    public static void main(String[] args)
    {
        int n;
        Scanner sin = new Scanner(System.in);
        n = sin.nextInt();
        String[] a1 = new String[n];
        for(int i=0;i<n;i++)
        {
            a1[i] = sin.next();
        }
        a1 = UserMainCode.orderElements(a1);
        for(int i=0;i<a1.length;i++)
        System.out.println(""+a1[i]);
    }
}
```

**UserMainCode:-**

```
import java.util.*;
```

```

public class UserMainCode
{
    public static String[] orderElements(String[] arr)
    {
        HashSet<String> al=new HashSet<String>();
        for(int i=0;i<arr.length;i++)
        {
            al.add(arr[i]);
        }

        Iterator<String> itr=al.iterator();
        String ar[] = new String[al.size()];
        int i =0 ;
        while(itr.hasNext()){
            ar[i] = itr.next();
            i++;
        }

        Arrays.sort(ar);
        return ar;
    }
}

```

### 34.Pattern Matcher

Write a program to read a string and check if it complies to the pattern 'CPT-XXXXXX' where XXXXXX is a 6 digit number. If the pattern is followed, then print TRUE else print FALSE.

Include a class UserMainCode with a static method **CheckID** which accepts the string. The return type is a boolean value.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

## **Input and Output Format:**

Input consists of a string.

Output should print TRUE or FALSE .

Refer sample output for formatting specifications.

### **Sample Input 1:**

CPT-302020

### **Sample Output 1:**

TRUE

### **Sample Input 2:**

CPT123412

### **Sample Output 2:**

FALSE

Main:

```
import java.util.*;

public class Main

{

    public static void main(String[] args)

    {

        Scanner sc = new Scanner(System.in);

        String s = sc.next();

        System.out.println(UserMainCode.CheckID(s));

        sc.close();

    }

}

UserMainCode:

public class UserMainCode

{

    public static boolean CheckID(String s)
```

```

{
boolean b=false;

if(s.matches("(CPT)[-]{1}[0-9]{6}"))
{
b=true;
}

else
{
b=false;
}

return b;
}
}

```

### **35.Playing with String - I**

Given a string array and non negative integer (n) apply the following rules.

1. Pick nth character from each String element in the String array and form a new String.
2. If nth character not available in a particular String in the array consider \$ as the character.
3. Return the newly formed string.

Include a class UserMainCode with a static method **formString** which accepts the string and

integer. The return type is the string formed based on rules.

Create a Class Main which would be used to accept the string and integer and call the static method present in UserMainCode.

#### **Input and Output Format:**

Input consists of a an integer which denotes the size of the array followed by the array of strings and an integer (n).

Output consists of a string .

Refer sample output for formatting specifications.

**Sample Input 1:**

4

ABC

XYZ

EFG

MN

3

**Sample Output 1:**

CZG\$

Main:

```
import java.util.Scanner;

public class Main

{
    public static void main(String[] args)

    {
        Scanner s=new Scanner(System.in);

        int n=Integer.parseInt(s.nextLine());

        String[] sc=new String[n];

        for(int i=0;i<n;i++)

        {
            sc[i]=s.nextLine();
        }

        int a=Integer.parseInt(s.nextLine());

        System.out.println(UserMainCode.formString(n,sc,a));

        s.close();
    }
}
```

```

UserMainCode:

public class UserMainCode {

    public static String formString(int n, String[] input, int a)
    {
        StringBuffer sb=new StringBuffer();
        for(int i=0;i<n;i++)
        {
            if(input[i].length()>=a)
            {
                String a1=input[i];
                sb.append(a1.charAt(a-1));
            }
            else
            {
                sb.append('$');
            }
        }
        return sb.toString();
    }
}

```

## 36. Regular Expression - 1

Given a string (s) apply the following rules.

1. String should be only four characters long.
2. First character can be an alphabet or digit.
3. Second character must be uppercase 'R'.
4. Third character must be a number between 0-9.

If all the conditions are satisfied then print TRUE else print FALSE.

Include a class UserMainCode with a static method **validate** which accepts the string. The return type is the boolean formed based on rules.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of TRUE or FALSE .

Refer sample output for formatting specifications.

### **Sample Input 1:**

vR4u

### **Sample Output 1:**

TRUE

### **Sample Input 2:**

vRau

### **Sample Output 2:**

FALSE

### **Sample Input 3:**

vrau

### **Sample Output 3:**

FALSE

```
S.36) import java.util.Scanner;  
  
public class Main {  
  
    public static void main(String []args){  
  
        Scanner sc=new Scanner(System.in);  
  
        String n=sc.nextLine();  
  
        System.out.println(UserMainCode.validate(n));  
  
        sc.close();  
    }  
}
```

```

public class UserMainCode
{
    public static String validate(String s)
    {
        String w="FALSE";
        if(s.length()==4 &&
            (Character.isDigit(s.charAt(0))||Character.isAlphabetic(s.charAt(0)))&&s.charAt(1)
            =='R')
        {
            if(Character.isDigit(s.charAt(2)))
            w="TRUE";
        }
        return w;
    }
}

```

### **37.Regular Expression – 2 (Age Validator)**

Given the age of a person as string, validate the age based on the following rules.

1. Value should contain only numbers.
2. Value should be non-negative.
3. Value should be in the range of 21 to 45'.

If all the conditions are satisfied then print TRUE else print FALSE.

Include a class UserMainCode with a static method **ValidateAge** which accepts the string.

The return type is the boolean formed based on rules.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

#### **Input and Output Format:**

Input consists of a string.

Output consists of TRUE or FALSE .

Refer sample output for formatting specifications.

**Sample Input 1:**

23

**Sample Output 1:**

TRUE

**Sample Input 2:**

-34

**Sample Output 2:**

FALSE

**Sample Input 3:**

3a

**Sample Output 3:**

FALSE

AcB/TRUE

Main:

```
import java.util.*;
public class Main {
    public static void main(String[] args){
        Scanner s=new Scanner(System.in);
        //Regular Expression - 2
        (Age Validator) pg.No:150
        String n=s.nextLine();
        boolean b=UserMainCode.ValidateAge(n);
        if(b==true)
        {
            System.out.println("TRUE");
        }
        else
            System.out.println("FALSE");
```

```

        s.close();
    }
}

UserMainCode:

public class UserMainCode {
    public static boolean ValidateAge(String n)
    {
        boolean b = false;
        if(n.matches("[0-9]{2}"))
        {
            //Regular Expression - 2
            (Age Validator) pg.No:150
            int a=Integer.parseInt(n);
            if(a>0&&a>=21&&a<=45)
            {
                b=true;
            }
            else
                b=false;
        }
        return b;
    }
}

```

### 38. Regular Expression – 3 (Phone Validator)

Given a phone number as string, validate the same based on the following rules.

1. Value should contain only numbers.
2. Value should contain 10 digits.
3. Value should not start with 00.

If all the conditions are satisfied then print TRUE else print FALSE.

Include a class UserMainCode with a static method **validatePhone** which accepts the string.

The return type is the boolean formed based on rules.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of TRUE or FALSE .

Refer sample output for formatting specifications.

### **Sample Input 1:**

9987684321

### **Sample Output 1:**

TRUE

### **Sample Input 2:**

0014623452

### **Sample Output 2:**

FALSE

Main:

```
import java.util.*;
public class Main {
public static void main(String[]args){
Scanner s=new Scanner(System.in);
String s1=s.nextLine();
boolean b1=UserMainCode.validatePhone(s1);
if(b1==true)
{
System.out.println("TRUE"); //phone validation pq.no:151
}
else
{
System.out.println("FALSE");
}
s.close();
}
```

**UserMainCode:**

```
public class UserMainCode {  
    public static boolean validatePhone(String s1)  
    {  
        boolean b=false;  
        if(s1.matches("[0]{1}[0]{1}[0-9]{8}"))  
        {  
            b=false;  
        }  
        //phone validation pg.no:151  
    }  
    else  
    {  
        b=true;  
    }  
    return b;  
}
```

**39.String Splitter**

Write a program which would accept a string and a character as a delimiter. Apply the below rules

1. Using the delimiter, split the string and store these elements in array.
2. Reverse each element of the string and convert it into lowercase.

Include a class UserMainCode with a static method **manipulateLiteral** which accepts the string and character. The return type is the string array formed.

Create a Class Main which would be used to accept the string and character and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of a string and character.

Output consists of a string array.

Refer sample output for formatting specifications.

**Sample Input 1:**

AAA/bba/ccc/DDD

/

**Sample Output 1:**

```
aaa
abb
ccc
ddd
import java.util.*;
public class Main
{
    public static void main(String[] args)
    {
        Scanner s = new Scanner(System.in);
        String ip1=s.next();
        char ip2='/';
        String op[]=UserMainCode.manipulateLiteral(ip1,ip2);
        for(int i=0;i<op.length;i++)
            System.out.println(op[i]);
        s.close();
    }
    import java.util.ArrayList;
    import java.util.StringTokenizer;
    public class UserMainCode
    {
        public static String[] manipulateLiteral(String ip1, char ip2)
        {
            StringTokenizer t1 = new StringTokenizer(ip1,"/");
            ArrayList<String> lst = new ArrayList<String>();
            while(t1.hasMoreTokens())
            {
                StringBuffer sb = new StringBuffer();
                sb.append(t1.nextToken().toLowerCase());
                lst.add(sb.toString());
            }
        }
    }
}
```

```
lst.add(sb.reverse().toString());  
}  
  
String[] op = new String[lst.size()];  
  
for(int i = 0;i<lst.size();i++)  
{  
  
op[i] = lst.get(i);  
  
}  
  
return op;  
}  
}
```

## 40.Vowel Count

Write a program to read a string and count the number of vowels present in it.

Include a class UserMainCode with a static method **tellVowelCount** which accepts the string. The return type is the integer giving out the count of vowels.

Note: The check is case-insensitive.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

NewYork

### **Sample Output 1:**

2

### **Sample Input 2:**

Elephant

## Sample Output 2:

```
3

import java.util.*;

public class Main

{

    public static void main(String[] args) // Second set: 40.Vowel
Count//



    {

Scanner sc=new Scanner(System.in);

String s=sc.nextLine();

int max=UserMainCode.tellVowelCount(s);

System.out.println(max);

sc.close();

}

}

public class UserMainCode {

    public static int tellVowelCount(String s)

    {

        int max=0;

        int count=0;

        for(int i=0;i<s.length();i++)

        {

            char c=s.charAt(i);

            if(c=='a' || c=='e' || c=='i' || c=='o' || c=='u' || c=='A' || c=='E' || c=='I' ||
c=='O' || c=='U')

            {

                count++;

            }

        }

        if(count>max)
```

```
{  
    max=count;  
}  
  
return max;  
}  
}
```

## 41.Playing with String - II

Write a program to accept a string array as input, convert all the elements into lowercase and sort the string array. Display the sorted array.

Include a class UserMainCode with a static method **sortArray** which accepts the string array.

The return type is the string array formed based on requirement.

Create a Class Main which would be used to accept the string array and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a an integer which denotes the size of the array followed by the array of strings,

Output consists of a string array.

Refer sample output for formatting specifications.

### **Sample Input 1:**

5

AAA

BB

CCCC

A

ABCDE

## Sample Output 1:

```
a  
aaa  
abcde  
bb  
cccc  
import java.util.*;  
public class Main {  
    public static void main(String[] args) {  
        Scanner s=new Scanner(System.in);  
        int n=s.nextInt();  
        String s1[]=new String[n];  
        String s2[]=new String[n];  
        for(int i=0;i<n;i++)  
            //S.41.Playing with  
String - II//  
        {  
            s1[i]=s.next();  
        }  
        s2=UserMainCode.sortArray(s1,n);  
        for (int i = 0; i < n; i++) {  
            System.out.println(s2[i]);  
        }  
        s.close();  
    }  
import java.util.Arrays;  
public class UserMainCode {  
    public static String[] sortArray(String s1[],int n){
```

```

String s2[] = new String[n];

for (int i = 0; i < n; i++)

{

    s2[i] = s1[i].toLowerCase();

}

Arrays.sort(s2);

return s2;

}
}

```

## 42. Median Calculation

Write a program to accept an int array as input, and calculate the median of the same.

Median Calculation Procedure:

1. Sort the sequence of numbers.
2. The total number count is odd, Median will be the middle number.

The total number count is even, Median will be the average of two middle numbers, After calculating the average, round the number to nearest integer.

Include a class UserMainCode with a static method **calculateMedian** which accepts the int array. The return type is the integer which would be the median.

Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a an integer which denotes the size of the array followed by the array of integers.

Output consists of a integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

7

1

2

1

4

7

1

2

**Sample Output 1:**

2

**Sample Input 2:**

6

52

51

81

84

60

88

**Sample Output 2:**

71

Main

```
import java.util.*;  
  
public class Main  
{  
    public static void main(String[] args)  
    {
```

```
int n;  
Scanner sin = new Scanner(System.in);  
n = sin.nextInt();  
int[] a1 = new int[n];  
for(int i=0;i<n;i++)  
{  
    a1[i] = sin.nextInt();  
}  
System.out.println(""+UserMainCode.calculateMedian(a1));  
sin.close();  
}  
}
```

### UserMainCode

```
import java.util.Arrays;  
  
public class UserMainCode  
{  
    public static int calculateMedian(int[] a)  
    {  
        Arrays.sort(a);  
        int length = a.length;  
        int result=0,mid=0,midNext=0;  
        if((length%2) != 0)  
        {  
            mid = (length/2)+1;  
            result = a[mid];  
        }  
        else  
        {  
            mid = length/2;  
        }  
    }  
}
```

```

        midNext = mid+1;

        float add = a[mid-1]+a[midNext-1];

        float div = add/2;

        result = Math.round(div);

    }

    return result;
}

}

```

### 43.Sequence in Array

Write a program to accept an int array as input, and check if [1,2,3] appears somewhere in the same sequence.

Include a class UserMainCode with a static method **searchSequence** which accepts the int

array. The return type is a boolean which returns true or false.

Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.

#### **Input and Output Format:**

Input consists of a an integer which denotes the size of the array followed by the array of integers.

Output should print true or false.

Refer sample output for formatting specifications.

#### **Sample Input 1:**

9

11

-2

5

1  
2  
3  
4  
5  
6

**Sample Output 1:**

TRUE

**Sample Input 2:**

6  
-2  
5  
1  
3  
2  
6

**Sample Output 2:**

FALSE

Main

```
import java.util.*;  
  
public class Main  
{  
    public static void main(String[] args)  
    {  
        Scanner s=new Scanner(System.in);  
        int n=s.nextInt();  
        int a[]={};
```

```

for(int i=0;i<n;i++){
    a[i]=s.nextInt();
}
System.out.println(UserMainCode.searchsequence(a));
s.close();
}
}

UserMainCode

public class UserMainCode {
    public static boolean searchsequence(int[] a)
    {
        boolean b = false;
        for(int i = 0 ; i< a.length-3; i++)
        {
            if(a[i]==1 && a[i+1]==2 && a[i+2]==3)
                b = true;
        }
        return b;
    }
}

```

#### 44.Asterisk & Characters

Write a program to read a string and return true or false based on the below rule:

1. Return true if for every '\*' in the string, there are same characters both side immediately before and after the star, else return false.

Include a class UserMainCode with a static method **scanStarNeighbors** which accepts the

string. The return type is the boolean TRUE or FALSE based on the rule.

Note: The check is case-insensitive.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of TRUE or FALSE.

Refer sample output for formatting specifications.

### **Sample Input 1:**

Hello\*World

### **Sample Output 1:**

FALSE

### **Sample Input 2:**

Welcome\*elizabeth

### **Sample Output 2:**

TRUE

Main

```
import java.util.*;

public class Main {

    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);

        String input=s.next();

        System.out.println( UserMainCode. scanStarNeighbors (input));
        s.close();
    }

}

UserMainCode

import java.util.StringTokenizer;
```

```

public class UserMainCode {

    public static boolean scanStarNeighbors (String input) {

        boolean b=false;

        StringTokenizer t=new StringTokenizer(input,"*");

        String s1=t.nextToken();

        String s2=t.nextToken();

        String s3=s1.substring(s1.length()-1);

        String s4=s2.substring(0,1);

        if(s3.equalsIgnoreCase(s4))

            b=true;

        return b;

    }

}

```

## 45.Occurance Count

Write a program to read a string that contains a sentence and read a word. Check the number of occurrences of that word in the sentence.

Include a class UserMainCode with a static method **countWords** which accepts the two strings. The return type is the integer giving the count.

Note: The check is case-sensitive.

Create a Class Main which would be used to accept the two strings and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of two strings.

Output consists of count indicating the number of occurrences.

Refer sample output for formatting specifications.

**Sample Input 1:**

Hello world Java is best programming language in the world  
world

**Sample Output 1:**

2

**Sample Input 2:**

hello world

World

**Sample Output 2:**

0

Main

```
import java.util.*;  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        Scanner s=new Scanner(System.in);  
  
        String s1=s.nextLine();  
        String s2=s.nextLine();  
  
        int v=UserMainCode.countWords(s1,s2);  
  
        System.out.println(v);  
  
        s.close();  
    }  
}
```

UserMainCode

```
import java.util.StringTokenizer;  
  
public class UserMainCode {  
  
    public static int countWords(String s1,String s2){  
  
        StringTokenizer t=new StringTokenizer(s1, " ");  
  
        int c=0;
```

```

while(t.hasMoreTokens())
{
    String s3=t.nextToken();
    if(s3.equals(s2))
        c++;
}
return c;
}
}

```

## 46.Regular Expressions - III

Write a program to read two strings S1 & S2, compute the number of times that S2 appears in S1.

Include a class UserMainCode with a static method **searchString** which accepts the two strings. The return type is the integer giving the count.

Note: The check is case-insensitive.

Create a Class Main which would be used to accept the two strings and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of two strings.

Output consists of count indicating the number of occurrences.

Refer sample output for formatting specifications.

### **Sample Input 1:**

Catcowcat

cat

### **Sample Output 1:**

**Sample Input 2:**

Catcowcat

catp

**Sample Output 2:**

0

Main

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.next();
        String s2=s.next();
        int v=UserMainCode.searchString(s1,s2);
        System.out.println(v);
        s.close();
    }
}
```

UserMainCode

```
public class UserMainCode {
    public static int searchString(String s1,String s2){
        int c=0;
        int t=s2.length();
        for(int i=0;i<s1.length()-t+1;i++){
            if(s2.equals(s1.substring(i,t+i))){
                c++;
            }
        }
    }
}
```

```
    return c;  
}  
  
}
```

## 47.Strings Processing

Write a program to read a string that contains comma separated fruit names and also a number N. Pick the nth fruit and return it. If the total number of elements are less than the number specified in N, then return the last element.

Include a class UserMainCode with a static method **findFruitName** which accepts the string and the number n. The return type is the string which has the fruit name.

Create a Class Main which would be used to accept the string and integer and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string and integer.

Output consists of a string.

Refer sample output for formatting specifications.

### **Sample Input 1:**

Apple,Banana,Orange

2

### **Sample Output 1:**

Banana

### **Sample Input 2:**

Apple,Banana,Orange

4

### **Sample Output 2:**

Orange

```
Main
import java.util.Scanner;

public class Main

{
    public static void main(String args[])
    {
        String str=new String();
        Scanner sc=new Scanner(System.in);
        str=sc.nextLine();
        int n=sc.nextInt();
        String k=UserMainCode.findFruitName(str, n);
        System.out.println(k);
        sc.close();
    }
}

UserMainCode
import java.util.StringTokenizer;

public class UserMainCode

{
    public static String findFruitName(String m,int n)
    {
        int i=0;
        String h=null;
        StringTokenizer st=new StringTokenizer(m,",");
        int max=st.countTokens();
        String[] ss=new String[max];
        while(st.hasMoreElements())
        {
            ss[i++]=st.nextToken();
        }
    }
}
```

```

    }

    if(n>max)

        h=ss[i-1];

    else

        h=ss[n-1];

    return h;

}
}

```

## 48.Poor Case

Write a program to read a string and convert the initial letter of each word to uppercase.

Include a class UserMainCode with a static method **changeCase** which accepts the string.

The return type is the modified string.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of a string.

Refer sample output for formatting specifications.

### **Sample Input 1:**

This is cognizant academy

### **Sample Output 1:**

This Is Cognizant Academy

Main

```

import java.util.*;

public class Main {

    public static void main(String[] args){
        Scanner s=new Scanner(System.in);

        String s1=s.nextLine();
    }
}

```

```
        System.out.println(UserMainCode.changeCase(s1));  
        s.close();  
    }  
}
```

### UserMainCode

```
import java.util.StringTokenizer;
```

```
public class UserMainCode {  
  
    public static String changeCase(String s1){  
  
        StringBuffer s5=new StringBuffer();  
        StringTokenizer t=new StringTokenizer(s1, " ");  
  
        while(t.hasMoreTokens()) {  
  
            String s2=t.nextToken();  
  
            String s3=s2.substring(0,1);  
  
            String s4=s2.substring(1, s2.length());  
  
            s5.append(s3.toUpperCase()).append(s4).append(" ");  
        }  
  
        return s5.toString();  
    }  
}
```

## 49.Length of same word

Write a program to read a string containing multiple words find the first and last words, if they are same, return the length and if not return the sum of length of the two words.

Include a class UserMainCode with a static method **compareLastWords** which accepts the

string. The return type is the length as per problem.

Create a Class Main which would be used to accept the string and call the static method

present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of a integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

This is Cognizant Academy

### **Sample Output 1:**

11

### **Sample Input 2:**

Hello World Hello

### **Sample Output 2:**

5

Main

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        // TODO Auto-generated method stub
        String s1=sc.nextLine();
        System.out.println(UserMainCode.compareLastWords(s1));
        sc.close();
    }
}
UserMainCode
import java.util.ArrayList;
```

```

import java.util.List;
import java.util.StringTokenizer;
public class UserMainCode {
    public static int compareLastWords(String s1){
        List<String> l=new ArrayList<String>();
        StringTokenizer t=new StringTokenizer(s1, " ");
        while(t.hasMoreTokens())
        {
            String s2=t.nextToken();
            l.add(s2);
        }
        String s3=l.get(0);
        String s4=l.get(l.size()-1);
        if(s3.equals(s4))
        {
            int n=s3.length();
            System.out.println(n);
        }
        else
        {
            int n1=s3.length();
            int n2=s4.length();
            int n=n1+n2;
        }
        Return n;
    }
}

```

## 50.Perfect Number

Write a program to that takes a positive integer and returns true if the number is perfect

number.

A positive integer is called a perfect number if the sum of all its factors (excluding the number itself, i.e., proper divisor) is equal to its value.

For example, the number 6 is perfect because its proper divisors are 1, 2, and 3, and  $6=1+2+3$ ; but the number 10 is not perfect because its proper divisors are 1, 2, and 5, and  $1+2+5 \neq 10$

Include a class UserMainCode with a static method **getPerfection** which accepts the number. The return type is boolean (true / false).

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a integer.

Output consists of TRUE / FALSE.

Refer sample output for formatting specifications.

### **Sample Input 1:**

28

### **Sample Output 1:**

TRUE

```
import java.util.*;  
  
public class Main {  
  
    public static void main(String[] args){  
  
        Scanner s=new Scanner(System.in);  
  
        int n=s.nextInt();  
  
        boolean j=(UserMainCode.getPerfection(n));  
  
        if(j==true)
```

```

System.out.println("TRUE");
else
System.out.println("FALSE");
}

}

public class UserMainCode {
public static boolean getPerfection(int n){
boolean b=false;
int sum=0;
for(int i=1;i<n;i++){
int r=n%i;
if(r==0)
sum=sum+i;
}
b=(sum==n);
return b;
}
}

```

## 51.Find Digits

For a given double number with atleast one decimal value, Write a program to compute the number of digits before and after the decimal point in the following format –  
noOfDigitsBeforeDecimal:noOfDigitsAfterDecimal.

Note: Ignore zeroes at the end of the decimal (Except if zero is the only digit after decimal).

Refer Example 2 and 3)

Include a class UserMainCode with a static method **findNoDigits** which accepts the decimal

value. The return type is string.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of a double.

Output consists of string.

Refer sample output for formatting specifications.

**Sample Input 1:**

843.21

**Sample Output 1:**

3:2

**Sample Input 2:**

20.130

**Sample Output 2:**

2:2

**Sample Input 3:**

20.130

```
import java.util.*;  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        Scanner s=new Scanner(System.in);  
  
        double d=s.nextDouble();  
  
        System.out.println(UserMainCode.findNoDigits(d));  
  
    }  
}
```

```
import java.util.StringTokenizer;

public class UserMainCode {
    public static String findNoDigits(double d) {
        int n1=0,n2=0;
        String s=String.valueOf(d);
        StringTokenizer t=new StringTokenizer(s,".");
        String s1=t.nextToken();
        String s2=t.nextToken();
        n1=s1.length();
        n2=s2.length();
        if(s1.charAt(0)=='0')
            n1=s1.length()-1;
        if(n2!=1)
            if(s2.charAt(s2.length()-1)=='0')
                n2=s2.length()-1;
        String s3=String.valueOf(n1)+":"+String.valueOf(n2);
        return s3;
    }
}
```

## 52.Employees & Designations

A Company wants to obtain employees of a particular designation. You have been assigned

as the programmer to build this package. You would like to showcase your skills by creating a quick prototype. The prototype consists of the following steps:

Read Employee details from the User. The details would include name and designation in the given order. The datatype for name and designation is string.

Build a hashmap which contains the name as key and designation as value.

You decide to write a function **obtainDesignation** which takes the hashmap and designation as input and returns a string List of employee names who belong to that designation as output. Include this function in class UserMainCode. Display employee name's in ascending order.

Create a Class Main which would be used to read employee details in step 1 and build the hashmap. Call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of employee details. The first number indicates the size of the employees. The next two values indicate the employee name employee designation. The last string would be the designation to be searched.

Output consists of a array values containing employee names.

Refer sample output for formatting specifications.

### **Sample Input 1:**

4

Manish

MGR

Babu

CLK

Rohit

MGR

Viru

PGR

MGR

### **Sample Output 1:**

Manish

Rohit

```
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.Scanner;
public class Main {
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        int k1=Integer.parseInt(sc.nextLine());
        LinkedHashMap<String, String> hm=new LinkedHashMap<String, String>();
        for(int i=0;i<k1;i++)
        {
            String k=sc.nextLine();
            String s=sc.nextLine();
            hm.put(k,s);
        }
        String n=sc.nextLine();
        LinkedHashMap<String, String> hm1=new LinkedHashMap<String, String>();
        hm1=UserMainCode.obtainDesignation(hm,n);
        Iterator<String> it=hm1.keySet().iterator();
        while(it.hasNext())
        {
```

```
String s2=it.next();
System.out.println(s2);
}
}
}
```

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.Map;
import java.util.Scanner;
public class UserMainCode
{
    public static LinkedHashMap<String,String>
    obtainDesignation(LinkedHashMap<String,String> h1,String n)
    {
        int k=0;
        LinkedHashMap<String,String> hm1=new LinkedHashMap<String,String>();
        Iterator<String>it=h1.keySet().iterator();
        while(it.hasNext())
        {
            String s2=it.next();
            String s3=h1.get(s2);
            if(s3.equals(n))
                hm1.put(s2,s3);
        }
        return hm1;
    }
}
```

}

### 53. Grade Calculator

A School wants to give assign grades to its students based on their marks. You have been assigned as the programmer to automate this process. You would like to showcase your skills by creating a quick prototype. The prototype consists of the following steps:

Read student details from the User. The details would include name, mark in the given order. The datatype for name is string, mark is float.

You decide to build a hashmap. The hashmap contains name as key and mark as value.

**BUSINESS RULE:**

1. If Mark is less than 60, then grade is FAIL.

2. If Mark is greater than or equal to 60, then grade is PASS.

Note: FAIL/PASS should be in uppercase.

Store the result in a new Hashmap with name as Key and grade as value.

4. You decide to write a function **calculateGrade** which takes the above hashmap as input and returns the hashmap as output. Include this function in class UserMainCode.

Create a Class Main which would be used to read student details in step 1 and build the hashmap. Call the static method present in UserMainCode.

#### **Input and Output Format:**

Input consists of student details. The first number indicates the size of the students. The next two values indicate the name, mark.

Output consists of a name and corresponding grade for each student.

Refer sample output for formatting specifications.

#### **Sample Input 1:**

3

Avi

76.36

Sunil

68.42

Raja

36.25

### **Sample Output 1:**

Avi

PASS

Sunil

PASS

Raja

FAIL

```
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.Scanner;
public class Main {
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        int k1=Integer.parseInt(sc.nextLine());
        LinkedHashMap<String, String> hm=new LinkedHashMap<String, String>();
        for(int i=0;i<k1;i++)
        {
```

```
String k=sc.nextLine();

String s=sc.nextLine();

hm.put(k,s);

}

String n=sc.nextLine();

LinkedHashMap<String,String> hm1=new LinkedHashMap<String,String>();

hm1=UserMainCode.obtainDesignation(hm,n);

Iterator<String> it=hm1.keySet().iterator();

while(it.hasNext())

{

String s2=it.next();

System.out.println(s2);

}

}

}

import java.util.HashMap;

import java.util.Iterator;

import java.util.LinkedHashMap;

import java.util.Map;

import java.util.Scanner;

public class UserMainCode

{

public static LinkedHashMap<String,String>

obtainDesignation(LinkedHashMap<String,String> h1,String n)

{

int k=0;

LinkedHashMap<String,String> hm1=new LinkedHashMap<String,String>();

Iterator<String>it=h1.keySet().iterator();
```

```
while(it.hasNext())
{
    String s2=it.next();
    String s3=h1.get(s2);
    if(s3.equals(n))
        hm1.put(s2,s3);
}
return hm1;
}}
```

## 54. DOB - Validation

Write a program to validate the Date of Birth given as input in String format (MM/dd/yyyy) as per the validation rules given below. Return true for valid dates else return false.

1. Value should not be null
2. month should be between 1-12, date should be between 1-31 and year should be a four digit number.

Include a class UserMainCode with a static method **ValidateDOB** which accepts the string. The return type is TRUE / FALSE.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of TRUE / FALSE.

Refer sample output for formatting specifications.

### **Sample Input 1:**

12/23/1985

**Sample Output 1:**

TRUE

**Sample Input 2:**

31/12/1985

**Sample Output 2:**

FALSE

```
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;
public class Main {
    public static void main(String[] args)
    {
        String str=new String();
        Scanner sc=new Scanner(System.in);
        str=sc.nextLine();
        Boolean b=UserMainCode.ValidateDOB(str);
        if(b=="true")
            System.out.println("TRUE");
        if(b=="false")
            System.out.println("FALSE");
    }
}
```

```
import java.text.SimpleDateFormat;
import java.util.Date;
public class UserMainCode {
    public static Boolean ValidateDOB(String str){
```

```
Boolean b="false";  
  
SimpleDateFormat sdf=new SimpleDateFormat("MM/dd/yyyy");  
  
sdf.setLenient(false);  
  
try  
  
{  
  
Date d1=sdf.parse(str);  
  
return b="true";  
  
}  
  
catch(Exception e)  
  
{  
  
return b="false";  
  
}  
  
}  
  
}
```

## 55.Experience Validator

Write a program to validate the experience of an employee.

An employee who has recently joined the organization provides his year of passing and total number of years of experience in String format. Write code to validate his experience against the current date.

- 1) Input consists of two String first represent the year of passed out and the second string represent the year of experience.
- 2) create a function with name **validateExp** which accepts two string as input and boolean as output.
- 3) The difference between current year and year of pass should be more than or equal to Experience

Return true if all condition are true.

Note: Consider 2015 as the current year.

Include a class UserMainCode with the static function validateExp

Create a Class Main which would be used to accept the boolean and call the static method present in UserMainCode.

### **Input and Output Formate:**

Input consists of two Strings.

output will display true if the given data are correct.

### **Sample Input:**

2001

5

### **Sample Output:**

TRUE

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Scanner;
public class Main {
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        String s=sc.nextLine();
        String s1=sc.nextLine();
        System.out.println(UserMainCode.validateExp(s,s1));
    }
}
```

```
import java.util.Calendar;
import java.util.Date;
public class UserMainCode {
    public static boolean validateExp(String s,String s1)
```

```
{  
    int y1=Integer.parseInt(s);  
    Date d=new Date();  
    Calendar c=Calendar.getInstance();  
    int y2=c.get(Calendar.YEAR);  
    int y=Math.abs(y1-y2);  
    int e=Integer.parseInt(s1);  
    if(y>=e)  
        return true;  
    else  
        return false;  
}
```

## 56. ArrayList to String Array

Write a program that performs the following actions:

Read n strings as input.

Create an arraylist to store the above n strings in this arraylist.

Write a function `convertToStringArray` which accepts the arraylist as input.

The function should sort the elements (strings) present in the arraylist and convert them into a string array.

Return the array.

Include a class `UserMainCode` with the static method **convertToStringArray** which accepts

an arraylist and returns an array.

Create a Class `Main` which would be used to read n strings and call the static method present in `UserMainCode`.

### Input and Output Format:

Input consists of n+1 integers. The first integer denotes the size of the arraylist, the next n

strings are values to the arraylist.

Output consists of an arrayas per step 4.

Refer sample output for formatting specifications.

### **Sample Input 1:**

4

a

d

c

b

### **Sample Output 1:**

a

b

c

d

```
import java.util.*;

public class Main

{
    public static void main(String[] args)

    {
        Scanner s=new Scanner(System.in);

        ArrayList<String> l=new ArrayList<String>();

        int n=s.nextInt();

        for(int i=0;i<n;i++)

        {
            l.add(s.next());
        }
    }
}
```

```

        }

        String a[]={};String[n];
        a=UserMainCode.convertToStringArray(l);
        for(int j=0;j<n;j++)
        {
            System.out.println(a[j]);
        }
    }

import java.util.ArrayList;
import java.util.Collections;

class UserMainCode
{
    public static String[] convertToStringArray(ArrayList<String> l)
    {
        Collections.sort(l);
        String [] a = l.toArray(new String[l.size()]);
        return a;
    }
}

```

## 57.State ID generator

Write a program to generate the state ID.

- 1)Read n Strings as input(as State Name).
- 2)Create a String Array to Store the above Input.
- 3)Write a function **getStatetId** which accepts String Array as input.

4)Create a `HashMap<String, String>` which stores state name as key and state Id as Value.

5)The function `getStatId` returns the `HashMap` to the Main Class.

Include UserMainCode Class With static method `getStatId` which accepts String array and return a hashmap.

Create a Class Main which would be used to read n strings and call the static method present in UserMainCode.

### **Input and Output Format:**

Input Consists of an integer n denotes the size of the string array.

Output consists of an `HashMap` displayed in the string array order.

### **Sample Input 1:**

3

Kerala

Gujarat

Goa

### **Sample Output 1:**

KER:Kerala

GUJ:Gujarat

GOA:Goa

#### **Main Class**

```
import java.util.*;  
  
public class Main  
{  
    public static void main(String[] args)  
    {  
        Scanner s=new Scanner(System.in);  
        int n=s.nextInt();
```

```
String[] s1=new String[n];

for(int i=0;i<n;i++)
{
    s1[i]=s.next();
}

HashMap<String, String> hm = new HashMap<String, String>();

hm = UserMainCode.putvalues(s1);

for(Map.Entry<String, String> ans: hm.entrySet())
{
    System.out.println(ans.getKey()+ ":" +ans.getValue());
}
}
```

User main code

```
import java.util.ArrayList;

import java.util.HashMap;

import java.util.Map;

public class UserMainCode{

    public static HashMap<String, String> putvalues(String[] s1)
    {
        HashMap<String, String> hm = new HashMap<String, String>();

        ArrayList<String> lst1 = new ArrayList<String>();

        ArrayList<String> lst2 = new ArrayList<String>();
```

```

for(String s : s1)

    lst1.add(s.toUpperCase().substring(0,3));

for(String s : s1)

    lst2.add(s);

for(int i=0;i<s1.length;i++)

{

    hm.put(lst1.get(i),lst2.get(i));

}

return hm;

}

```

## 58. ArrayList to String Array

Write a program that performs the following actions:

- 1.Read m strings as input (fruit names).
- 2.Create an arraylist to store the above m strings in this arraylist.
- 3.Read n strings as input (fruit names).
- 4.Create an arraylist to store the above n strings in this arraylist.
- 5.Write a function fruitSelector which accepts the arraylists as input.
- 6.Remove all fruits whose name ends with 'a' or 'e' from first arrayList and remove all fruits whose name begins with 'm' or 'a' from second arrayList then combine the two lists and return the final output as a String array.
- 7.If the array is empty the program will print as "No fruit found"

Include a class UserMainCode with the static method **fruitSelector** which accepts the two arraylists and returns an array.

Create a Class Main which would be used to read n strings and call the static method

present in UserMainCode.

**Input and Output Format:**

Input consists of an integer (m) denoting the size of first arraylist. The next m elements would be the values of the first arraylist. The next input would be n denoting the size of the second arraylist. The next n elements would be the values of the second arraylist.

Output consists of an array as per step 6. Refer sample output for formatting specifications.

**Sample Input 1:**

3

Apple

Cherry

Grapes

4

Orange

Mango

Melon

Apple

**Sample Output 1:**

Cherry

Grapes

Orange

**USERMAINCODE:**

```
import java.util.ArrayList;  
import java.util.*;  
  
public class UserMainCode {
```

```
public static String[] fruitSelector(ArrayList<String> a1,ArrayList<String> a2)

{
    ArrayList<String> a3=new ArrayList<String>();

    for(int i=0;i<a1.size();i++)
    {
        String s1=a1.get(i);

        if(s1.charAt(s1.length()-1)!='a'&&s1.charAt(s1.length()-1)!='e'&&s1.charAt(s1.length()-1)!='A'&&s1.charAt(s1.length()-1)!='E')

        {
            a3.add(s1);
        }
    }

    ArrayList<String> a4=new ArrayList<String>();

    for(int j=0;j<a2.size();j++)
    {
        String s2=a2.get(j);

        if(s2.charAt(0)!='m'&&s2.charAt(0)!='a'&&s2.charAt(0)!='M'&&s2.charAt(0)!='A')

        {
            a4.add(s2);
        }
    }

    a3.addAll(a4);

    Collections.sort(a3);

    String st[]=new String[a3.size()];

    for(int k=0;k<a3.size();k++)
    {
        st[k]=a3.get(k);
    }

    return st;
}
```

```
}
```

```
}
```

MAIN:

```
import java.util.*;  
  
import java.util.ArrayList;  
  
public class Main {  
  
    public static void main(String [] args)  
    {  
  
        Scanner s=new Scanner(System.in);  
  
        int m=s.nextInt();  
  
        ArrayList<String> aa1=new ArrayList<String>();  
  
        for(int i=0;i<m;i++)  
        {  
  
            aa1.add(s.next());  
  
        }  
  
        int n=s.nextInt();  
  
        ArrayList<String> aa2=new ArrayList<String>();  
  
        for(int j=0;j<n;j++)  
        {  
  
            aa2.add(s.next());  
  
        }  
  
        int k;  
  
        String st[]=UserMainCode.fruitSelector(aa1,aa2);  
  
        for( k=0;k<st.length;k++)  
        {  
  
            System.out.println(st[k]);  
  
        }  
  
        if(st.length==0)
```

```
System.out.println("No Fruit Found");

s.close();

}

}
```

## 59)Elements in ArrayList

Use Collection Methods.

Write a program that takes two ArrayLists as input and finds out all elements present either in A or B, but not in both.

Include a class UserMainCode with the static method arrayListSubtractor which accepts the two arraylists and returns an array.

Create a Class Main which would be used to read the inputs and call the static method present in UserMainCode.

### Input and Output Format:

Input consists of an integer (m) denoting the size of first arraylist. The next m elements would be the values of the first arraylist. The next input would be n denoting the size of the second arraylist. The next n elements would be the values of the second arraylist.

Output consists of an array. The elements in the output array need to be printed in sorted order.

Refer sample output for formatting specifications.

### Sample Input 1:

```
4
1
8
3
5
2
3
5
```

### Sample Output 1:

```
1
8
```

### Sample Input 2:

```
4
9
1
3
5
4
1
3
5
6
```

### Sample Output 2:

```
6
```

9

## MAIN:

```
import java.util.*;
public class Main
{
public static void main(String[] args)
{
int n,m;
Scanner sin = new Scanner(System.in);
n = sin.nextInt();
ArrayList<Integer> a1 = new ArrayList<Integer>(n);
for(int i=0;i<n;i++)
{
int k = sin.nextInt();
a1.add(k);
}
m = sin.nextInt();
ArrayList<Integer> a2 = new ArrayList<Integer>(m);
for(int i=0;i<m;i++)
{
int k = sin.nextInt();
a2.add(k);
}
int[] result = UserMainCode.arrayListSubtractor(a1,a2);
Arrays.sort(result);
for(int i=0;i<result.length;i++)
System.out.println(result[i]);
}
}
```

## USERMAINCODE:

```
import java.util.ArrayList;
public class UserMainCode
{
public static int[] arrayListSubtractor(ArrayList<Integer>
arrlist1,ArrayList<Integer>
arrlist2)
{
int count=0,key;
int max = arrlist1.size();
if(arrlist1.size() < arrlist2.size())
max = arrlist2.size();
ArrayList<Integer> temp = new ArrayList<Integer>(max);
for(int i=0;i<arrlist1.size();i++)
{
key = (int)arrlist1.get(i);
if(arrlist2.indexOf(key) == -1)
{
++count;
temp.add(key);
}
}
for(int i=0;i<arrlist2.size();i++)
{
key = (int)arrlist2.get(i);
if(arrlist1.indexOf(key) == -1)
```

```

{
if(!temp.contains(key))
{
++count;
temp.add(key);
}
}
}

int[] result = new int[count];
for(int i=0;i<count;i++)
result[i] = (int)temp.get(i);
return result;
}
}

```

## 60.Price Calculator - II

Write a small price calculator application with the below mentioned flow:

1. Read a value n indicating the total count of devices. This would be followed by the name and price of the device. The datatype for name would be String and price would be float.
2. Build a hashmap containing the peripheral devices with name as key and price as value.
3. Read a value m indicating the number of devices for which the price has to be calculated. This would be followed by device names.
4. For each devices mentioned in the array calcuate the total price.
5. You decide to write a function costEstimator which takes the above hashmap and array as input and returns the total price (float) as output with two decimal points. Include this function in class UserMainCode.

Create a Class Main which would be used to read details in step 1 and build the hashmap.

Call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of device details. The first number indicates the size of the devices. The next two values indicate the name,price.

This would be followed by m indicating the size of the device array. The next m values would be the device names.

Output consists of the total price in float.

Refer sample output for formatting specifications.

### **Sample Input 1:**

```

3
Monitor
1200.36
Mouse
100.42
Speakers
500.25
2
Speakers
Mouse

```

### **Sample Output 1:**

600.67

## MAIN:

```
import java.util.HashMap;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner S=new Scanner(System.in);
        int n=S.nextInt();

        HashMap<String, Float> m1=new HashMap<String, Float>();
        for(int i=0;i<n;i++)
        {
            String name=S.next();
            float price=S.nextFloat();
            m1.put(name,price);
        }
        int m=S.nextInt();
        String s[]=new String[m];
        for(int j=0;j<m;j++)
        {
            s[j]=S.next();
        }
        System.out.println(UserMainCode.getTheTotalCostOfPeripherals
(m1,s));
    }
}
```

## USERMAINCODE:

```
import java.util.HashMap;
import java.util.Iterator;

public class UserMainCode {
    public static float getTheTotalCostOfPeripherals(HashMap<String,Float> m1,
String[] s) {
        float f=(float) 0;
        Iterator<String> i=m1.keySet().iterator();
        while(i.hasNext()){
            String s1=i.next();
            float f1=m1.get(s1);
            for(int j=0;j<s.length;j++)
            if(s[j].equals(s1))
                f+=f1;
        }
        return f;
    }
}
```

## 61.String Processing - ZigZag

Write a program to read a string containing date in DD-MM-YYYY format. find the number of days in the given month.

Note - In leap year February has got 29 days.

Include a class UserMainCode with a static method **getLastDayOfMonth** which accepts the

string. The return type is the integer having number of days.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

12-06-2012

### **Sample Output 1:**

30

### **Sample Input 2:**

10-02-2012

### **Sample Output 2:**

29

## **MAIN:**

```
import java.io.BufferedReader;
        import java.io.IOException;
        import java.io.InputStreamReader;
        import java.text.ParseException;
        import java.text.SimpleDateFormat;
        import java.util.*;
public class Main {
    public static void main(String[] args) throws IOException,
ParseException {
        Scanner S=new Scanner(System.in);
        String s1=S.next();
        UserMainCode.getLastDayOfMonth(s1);
    }
}
```

## **USERMAINCODE:**

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

public class UserMainCode {
    public static void getLastDayOfMonth(String s1) throws ParseException{
        SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");
        Calendar cal=Calendar.getInstance();
        Date d1=sdf.parse(s1);
        cal.setTime(d1);
        int n=cal.getActualMaximum(Calendar.DAY_OF_MONTH);
        System.out.println(n);
    }
}
```

## **62.Leap Year**

Write a program to read a string containing date in DD/MM/YYYY format and check if its a leap year. If so, return true else return false.

Include a class UserMainCode with a static method **isLeapYear** which accepts the string. The

return type is the boolean indicating TRUE / FALSE.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of TRUE / FALSE.

Refer sample output for formatting specifications.

### **Sample Input 1:**

23/02/2012

### **Sample Output 1:**

TRUE

### **Sample Input 2:**

12/12/2011

### **Sample Output 2:**

FALSE

## **MAIN:**

```
import java.io.IOException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.*;
public class Main {
    public static void main(String[] args) throws IOException, ParseException {
        Scanner S=new Scanner(System.in);
        String s1=S.next();
        UserMainCode.isLeapyear(s1);
    }
}
```

## **USERMAINCODE:**

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.GregorianCalendar;
import java.util.StringTokenizer;

public class UserMainCode {
    public static void isLeapyear(String s1) throws ParseException{
        SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
        GregorianCalendar g=new GregorianCalendar();
        StringTokenizer t=new StringTokenizer(s1,"/");
        String s2=t.nextToken();
        String s3=t.nextToken();
        String s4=t.nextToken();
        int n1=Integer.parseInt(s4);
        Date d1=sdf.parse(s1);
        boolean b=g.isLeapYear(n1);
        System.out.println(b);
    }
}
```

## **63) Largest Chunk**

Write a program to read a string and return the length of the largest "chunk" in the string.

A chunk is a repetition of same character 2 or more number of times. If the given string

does not contain any repeated chunk of characters return -1.

Include a class UserMainCode with a static method **getLargestSpan** which accepts the string.

The return type is the integer.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

This place is soooo good

### **Sample Output 1:**

4

### **MAIN:**

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner S=new Scanner(System.in);
        String s1=S.nextLine();
        System.out.println(UserMainCode.getLargestSpan(s1));
    }
}
```

### **USERMAINCODE :**

```
import java.util.StringTokenizer;

public class UserMainCode {
    public static int getLargestSpan(String s1) {
        int max=0;
        StringTokenizer t=new StringTokenizer(s1, " ");
        while(t.hasMoreTokens()){
            String s2=t.nextToken();
            int n=0;
            for(int i=0;i<s2.length()-1;i++)
                if(s2.charAt(i)==s2.charAt(i+1))
                    n++;
            if(n>max)
                max=n;
        }
        return (max+1);
    }
}
```

## **64) Largest Span**

Write a program to read a integer array, find the largest span in the array.

Span is the count of all the elements between two repeating elements including the repeated elements.

Include a class UserMainCode with a static method **getLargestSpan** which accepts the integer array. The return type is integer.

Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of an integer n which is the number of elements followed by n integer values.

Output consists of integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

```
6  
4  
2  
1  
4  
5  
7
```

### **Sample Output 1:**

```
4
```

### **MAIN:**

```
import java.util.Scanner;  
public class Main {  
    public static void main(String[] args)  
    {  
        Scanner sc=new Scanner(System.in);  
        int n=sc.nextInt();  
        int []a=new int[n];  
        for(int i=0;i<n;i++)  
        {  
            a[i]=sc.nextInt();  
        }  
        System.out.print(UserMainCode.getLargestSpan(a,n));  
    }}
```

### **USERMAINCODE:**

```
public class UserMainCode {  
    public static int getLargestSpan(int[] x,int n)  
    {  
        int gap=0,max=0;  
        for(int i=0;i<n;i++)  
        {  
            for(int j=i+1;j<n;j++)  
            {  
                if(x[i]==x[j])  
                {  
                    gap=j;  
                }  
            }  
            if(gap-i>max)  
                max=gap-i;  
        }  
        return max+1;  
    }  
}
```

### **65 )Even Sum & Duplicate Elements**

Write a program to read a integer array, Remove the duplicate elements and display sum of even numbers in the output. If input array contain only odd number then return -1.

Include a class UserMainCode with a static method **sumElements** which accepts the integer

array. The return type is integer.

Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of an integer n which is the number of elements followed by n integer values.

Output consists of integer.

Refer sample output for formatting specifications.

### **Sample Input 1:**

```
7  
2  
3  
54  
1  
6  
7  
7
```

### **Sample Output 1:**

```
62
```

### **Sample Input 2:**

```
6  
3  
7  
9  
13  
17  
21
```

### **Sample Output 2:**

```
-1
```

### **MAIN:**

```
import java.util.HashMap;  
import java.util.LinkedHashMap;  
import java.util.LinkedHashSet;  
import java.util.Scanner;  
public class Main  
{  
    public static void main(String args[])  
    {  
        Scanner sc=new Scanner(System.in);  
        int n=sc.nextInt();  
        int a[]={};  
        for(int i=0;i<n;i++)  
        {  
            a[i]=sc.nextInt();  
        }  
        System.out.println(UserMainCode.sumElements(a));  
    }  
}
```

**USERMAINCODE:**

```
import java.util.Iterator;
import java.util.LinkedHashSet;
public class UserMainCode {
public static int sumElements(int a[])
{
LinkedHashSet<Integer> h1=new LinkedHashSet<Integer>();
int s=0;
for(int i=0;i<a.length;i++)
{
h1.add(a[i]);
}
Iterator<Integer> it=h1.iterator();
while(it.hasNext())
{
int k=it.next();
if(k%2==0)
{
s=s+k;
}
}
if(s>0)
return s;
else
return -1;
}
}
```

## 66. Regular Expression - III

Given a string (s) apply the following rules.

- I)At least 8 characters must be present
- II)At least one capital letter must be present
- III)At least one small letter must be present
- Iv)At least one special symbol must be present
- V)At least one numeric value must be present

If the condition is satisfied then print valid else print invalid.

Include a class UserMainCode with a static method passwordValidation which accepts the string. The return type is the string.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

Input and Output Format:

Input consists of a string.

Output consists of string (valid / invalid) .

Refer sample output for formatting specifications.

Sample Input 1:

Technology\$1213

Sample Output 1:

valid

**Main:**

```
import java.util.Scanner;
```

```

public class Main {
    public static void main(String[] args)
    {
        Scanner s=new Scanner(System.in);
        String a=s.next();
        System.out.println(UserMainCode.passwordValidation(a));
        s.close();
    }
}

UserMainCode:
public class UserMainCode
{
    public static String passwordValidation(String a)
    {
        String k;
        if(a.matches(".*[0-
9]{1,}.*")&&a.matches(".*[@#$]{1,}.*")&&a.length()>=8&&a.matches(".*[A-
Z]{1,}.*")&&a.matches(".*[a-z].*"))
        {
            k="validinput";
        }
        else
        {
            k="Invalidinput";
        }
        return k;
    }
}

```

## 67. Integer Factorial

Give an array of integer as input, store the numbers and their factorials in an hashmap and print the same.

Include a class UserMainCode with a static method getFactorial which accepts the integer array. The return type is the hashmap which is printed key:value.

Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.

Input and Output Format:

Input consists of a number denoting the size of the array and followed by the elements.

Output consists of a hashmap printed in the output format .

Refer sample output for formatting specifications.

Sample Input1:

```

4
2
3
5
4

```

Sample Output1:

2:2

3:6

5:120

4:24

**Main:**

```
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.Scanner;
```

```
public class Main
{
    public static void main(String[] args)
    {
        Scanner s=new Scanner(System.in);
        int a=Integer.parseInt(s.nextLine());
        int[] k=new int[a];
        for(int i=0;i<a;i++)
        {
            k[i]=s.nextInt();
        }
        LinkedHashMap<Integer,Integer> hm=new LinkedHashMap<Integer,Integer>();
        hm=UserMainCode.getFactorial(k);
        Iterator<Integer> it=hm.keySet().iterator();
        for(int i=0;i<a;i++)
        {
            int n=it.next();
            int fac=hm.get(n);
            System.out.println(n+":"+fac);
            s.close();
        }
    }
}
```

**UserMainCode;**

```
import java.util.LinkedHashMap;
```

```
public class UserMainCode
{
    public static LinkedHashMap<Integer,Integer> getFactorial(int[] k)
    {
        LinkedHashMap<Integer,Integer> hm1=new LinkedHashMap<Integer,Integer>();
        for(int i=0;i<k.length;i++)
        {
            int u=1;
            for(int j=1;j<=k[i];j++)
            {
                u=u*j;
            }
        }
    }
}
```

```

    }
    hm1.put(k[i],u);
}
return hm1;
}

}

```

### **68. String processing – Long + Short + Long**

Obtain two strings S1,S2 from user as input. Your program should form a string of “long+short+long”, with the shorter string inside of the longer String.

Include a class UserMainCode with a static method getCombo which accepts two string variables. The return type is the string.

Create a Class Main which would be used to accept two Input strings and call the static method present in UserMainCode.

Input and Output Format:

Input consists of two strings with maximum size of 100 characters.

Output consists of an string.

Refer sample output for formatting specifications.

Sample Input 1:

Hello

Hi

Sample Output 1:

HelloHiHello

**Main;**

```
import java.util.Scanner;
```

```

public class Main
{
public static void main(String[] args)
{
Scanner s=new Scanner(System.in);
String s1=s.next();
String s2=s.next();
System.out.println(UserMainCode.getCombo(s1,s2));
s.close();
}
}

```

**UserMainCode;**

```

public class UserMainCode
{
public static String getCombo(String s1,String s2)
{
StringBuffer sb=new StringBuffer();
int p=s1.length();
int q=s2.length();
if(p>q)

```

```

{
sb.append(s1).append(s2).append(s1);
}
else
{
sb.append(s2).append(s1).append(s2);
}
return sb.toString();
}

}

```

## **69. Age for Voting**

Given a date of birth (dd/MM/yyyy) of a person in string, compute his age as of 01/01/2015.  
If his age is greater than 18, then println eligible else println not-eligible.

Include a class UserMainCode with a static method getAge which accepts the string value.  
The return type is the string.

Create a Class Main which would be used to accept the two string values and call the static  
method present in UserMainCode.

Input and Output Format:

Input consists of two string.

Output consists of a string.

Refer sample output for formatting specifications.

Sample Input 1:

16/11/1991

Sample Output 1:

eligible

**Main:**

```
import java.util.Scanner;
```

```

public class Main {
public static void main(String[] args)
{
Scanner s=new Scanner(System.in);
String a=s.nextLine();
System.out.println(UserMainCode.getAge(a));
s.close();
}
}

```

**UserMainCode:**

```
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;
```

```

public class UserMainCode
{
public static String getAge(String n)
{

```

```

Scanner s=new Scanner(System.in);
int year=0;
String s1=s.next();
SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
try
{
Date d=sdf.parse(n);
Date d1=sdf.parse(s1);
int y=d.getYear();
int y1=d1.getYear();
int m=d.getMonth();
int m1=d1.getMonth();
int day=d.getDate();
int day1=d1.getDate();
year=y1-y;
if(m>m1)
year--;
else if(m==m1)
{if(day<day1)
year--;
}
}
catch(Exception e)
{
e.printStackTrace();
}
if(year>18)
return "eligible";
else
return "not-eligible";
}
}

```

## 1. Unique Even Sum

Write a program to read an array, eliminate duplicate elements and calculate the sum of even numbers (values) present in the array.

Include a class UserMainCode with a static method addUniqueEven which accepts a single integer array. The return type

(integer) should be the sum of the even numbers. In case there is no even number it should return -1.

Create a Class Main which would be used to accept Input array and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of  $n+1$  integers. The first integer corresponds to  $n$ , the number of elements in the array. The next ' $n$ ' integers correspond to the elements in the array.

In case there is no even integer in the input array, print no even numbers as output. Else print the sum.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20.

Sample Input 1:

```
4  
2  
5  
1  
4
```

Sample Output 1:

```
6
```

Sample Input 2:

```
3  
1  
1  
1
```

Sample Output 2:

no even numbers

**Main:**

```
import java.util.Scanner;
```

```
public class Main  
{  
    public static void main(String[] args)  
    {  
        Scanner s=new Scanner(System.in);  
        int n=s.nextInt();  
        int[] a=new int [n];  
        for(int i=0;i<n;i++)  
        {  
            a[i]=s.nextInt();  
        }  
        System.out.println(UserMainCode.addUniqueEven(a));  
        s.close();  
    }  
}
```

```
import java.util.Iterator;  
import java.util.LinkedHashSet;
```

```
public class UserMainCode  
{  
    public static int addUniqueEven(int[] a)
```

```

{
int sum=0;
LinkedHashSet<Integer> hm=new LinkedHashSet<Integer>();
for(int i=0;i<a.length;i++)
{
hm.add(a[i]);
}
Iterator<Integer> im=hm.iterator();
while(im.hasNext())
{
int b=im.next();
if(b%2==0)
sum=sum+b;
}
if(sum>0)
{
return sum;
}
else
    return -1;
}
}

```

## 2. Palindrome & Vowels

Write a program to check if a given string is palindrome and contains at least two different vowels.

Include a class UserMainCode with a static method checkPalindrome which accepts a string.  
The return type (integer)

should be 1 if the above condition is satisfied, otherwise return -1.

Create a Class Main which would be used to accept Input string and call the static method present in UserMainCode.

Note – Case Insensitive while considering vowel, i.e a & A are same vowel, But Case sensitive while considering

palindrome i.e abc CbA are not palindromes.

Input and Output Format:

Input consists of a string with maximum size of 100 characters.

Output consists of a single Integer.

Refer sample output for formatting specifications.

Sample Input 1:

abceecba

Sample Output 1:

valid

Sample Input 2:

abcd

Sample Output 2:

invalid

**Main;**

import java.util.Scanner;

```

public class Main
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        String s=sc.nextLine();
        System.out.println(UserMainCode.checkPalindrome(s));
        sc.close();
    }
}

UserMainCode;
import java.util.Iterator;
import java.util.LinkedHashSet;

public class UserMainCode
{
    public static int checkPalindrome(String s)
    {
        StringBuffer sb=new StringBuffer(s);
        int k=0;
        LinkedHashSet<Character>l1=new LinkedHashSet<Character>();
        String s2=sb.reverse().toString();
        if(s2.equals(s))
        {
            String s3=s2.toLowerCase();
            for(int i=0;i<s3.length();i++)
            {
                l1.add(s3.charAt(i));
            }
            Iterator<Character> it=l1.iterator();
            while(it.hasNext())
            {
                char a=it.next();
                if(a=='a'||a=='e'||a=='i'||a=='o'||a=='u')
                k++;
            }
        }
        if(k>=2)
        return 1;
        else
        return -1;
    }
}

```

### 3. Strings – Unique & Existing Characters

Obtain two strings from user as input. Your program should modify the first string such that all the characters are replaced

by plus sign (+) except the characters which are present in the second string.

That is, if one or more characters of first string appear in second string, they will not be replaced by +.

Return the modified string as output. Note - ignore case.

Include a class UserMainCode with a static method replacePlus which accepts two string variables. The return type is the modified string.

Create a Class Main which would be used to accept two Input strings and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of two strings with maximum size of 100 characters.

Output consists of a single string.

Refer sample output for formatting specifications.

**Sample Input 1:**

abcxyz

axdef

**Sample Output 1:**

a++ x++

**Sample Input 2:**

ABCDEF

feCBAd

**Sample Output 2:**

ABCDEF

**Main;**

import java.util.Scanner;

```
public class Main
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        String s=sc.next();
        String s1=sc.next();
        System.out.println(UserMainCode.replacePlus(s,s1));
        sc.close();
    }
}
```

**UserMainCode;**

```
public class UserMainCode
{
    public static String replacePlus(String s,String s1)
    {
        String s2=s.toLowerCase();
        String s3=s1.toLowerCase();
```

```

StringBuffer sb=new StringBuffer();
for(int i=0;i<s.length();i++)
{
char c=s2.charAt(i);
if(s3.indexOf(c)==-1)
sb.append("+");
else
sb.append(s.charAt(i));
} return sb.toString();
}
}
}
}

```

#### **4. Longest Word**

Write a Program which finds the longest word from a sentence. Your program should read a sentence as input from user and  
return the longest word. In case there are two words of maximum length return the word which comes first in the sentence.

Include a class UserMainCode with a static method getLargestWord which accepts a string  
The return type is the longest  
word of type string.

Create a Class Main which would be used to accept two Input strings and call the static  
method present in UserMainCode.

**Input and Output Format:**

Input consists of a string with maximum size of 100 characters.

Output consists of a single string.

Refer sample output for formatting specifications.

**Sample Input 1:**

Welcome to the world of Programming

**Sample Output 1:**

Programming

**Sample Input 2:**

ABC DEF

**Sample Output 2:**

ABC

**Main;**

import java.util.Scanner;

```

public class Main
{
public static void main(String[] args)
{
Scanner s=new Scanner(System.in);
String s1=s.nextLine();
System.out.println(UserMainCode.getLargestWord(s1));
s.close();
}

```

```

}
UserMainCode;

import java.util.StringTokenizer;

public class UserMainCode
{
    public static String getLargestWord(String s1)
    {
        int max=0;
        String s2=new String();
        StringTokenizer t=new StringTokenizer(s1," ");
        {
            while(t.hasMoreTokens()){
                String s3=t.nextToken();
                int n=s3.length();
                if(n>max){
                    max=n;
                    s2=s3;
                }
            }
        }
    }
}

```

## 5. String Occurrences

Obtain two strings from user as input. Your program should count the number of occurrences of second word of second sentence in the first sentence.

Return the count as output. Note - Consider case.

Include a class UserMainCode with a static method **countNumberOfWords** which accepts two string variables. The return type is the modified string.

Create a Class Main which would be used to accept two Input strings and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of two strings with maximum size of 100 characters.

Output consists of a single string.

Refer sample output for formatting specifications.

### **Sample Input 1:**

abc bcd abc bcd abc abc

av abc

**Sample Output 1:**

4

**Sample Input 2:**

ABC xyz AAA

w abc

**Sample Output 2:**

0

**UserMainCode**

```
import java.util.StringTokenizer;

public class UserMainCode {

    public static void countNoOfWords(String s1, String s2) {
        int count=0;

        StringTokenizer st=new StringTokenizer(s2, " ");
        String s3=st.nextToken();
        String s4=st.nextToken();

        //System.out.println(s4);

        StringTokenizer st1=new StringTokenizer(s1, " ");
        while(st1.hasMoreTokens())
        {
            String s5=st1.nextToken();
            if(s4.equals(s5))
            {
```

```
        count++;
    }
}

System.out.println(count);
}
}
```

## Main

```
import java.util.*;

public class Main

{
    /**
     * @param args
     */

    public static void main(String[] args)
    {
        Scanner s=new Scanner(System.in);

        String s1=s.nextLine();

        String s2=s.nextLine();

        UserMainCode.countNoOfWords(s1,s2);

        s.close();
    }
}
```

## 6. ArrayList Manipulation

Write a program that performs the following actions:

1. Read  $2n$  integers as input.
2. Create two arraylists to store  $n$  elements in each arraylist.
3. Write a function **generateOddEvenList** which accepts these two arraylist as input.
4. The function fetch the odd index elements from first array list and even index elements from second array list and add them to a new array list according to their index.
5. Return the arraylist.

Include a class UserMainCode with the static method **generateOddEvenList** which accepts two arraylist and returns an arraylist.

Create a Class Main which would be used to read  $2n$  integers and call the static method present in UserMainCode.

Note:

- The index of first element is 0.
- Consider 0 as an even number.
- Maintain order in the output array list

#### **Input and Output Format:**

Input consists of  $2n+1$  integers. The first integer denotes the size of the arraylist, the next  $n$  integers are values to the first arraylist, and the last  $n$  integers are values to the second arraylist.

Output consists of a modified arraylist as per step 4.

Refer sample output for formatting specifications.

#### **Sample Input 1:**

5

12

13

14

15

16

2

3

4

5

6

**Sample Output 1:**

2

13

4

15

6

**UserMainCode**

```
import java.util.ArrayList;
import java.util.Iterator;
public class UserMainCode
{
    public static ArrayList<Integer> generateOddEvenList
    (ArrayList<Integer>a1,ArrayList<Integer>a2)
    {
        ArrayList<Integer>a3=new ArrayList<Integer>();
        for(int i=0;i<a1.size();i++)
        {
            if(i%2==0)
                a3.add(a2.get(i));
            else
                a3.add(a1.get(i));
        }
        return a3;
    }
}
```

```
}
```

## Main

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Scanner;

public class Main {

    public static void main(String []args){

        Scanner sc=new Scanner(System.in);

        int s=Integer.parseInt(sc.nextLine());

        ArrayList<Integer>al1=new ArrayList<Integer>();
        ArrayList<Integer>al2=new ArrayList<Integer>();

        for(int i=0;i<s;i++)

            al1.add(sc.nextInt());

        for(int i=0;i<s;i++)

            al2.add(sc.nextInt());

        ArrayList<Integer>al3=new ArrayList<Integer>();

        al3=UserMainCode.generateOddEvenList(al1,al2);

        Iterator<Integer> it=al3.iterator();

        while(it.hasNext())

        {

            int n=it.next();

            System.out.println(n);

            sc.close();

        }

    }

}
```

}

## 7. Duplicate Characters

Write a Program which removes duplicate characters from the string. Your program should read a sentence (string) as input from user and return a string removing duplicate characters. Retain the first occurrence of the duplicate character. Assume the characters are case – sensitive.

Include a class UserMainCode with a static method **removeDuplicates** which accepts a string. The return type is the modified sentence of type string.

Create a Class Main which would be used to accept the input string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string with maximum size of 100 characters.

Output consists of a single string.

Refer sample output for formatting specifications.

### **Sample Input 1:**

hi this is sample test

### **Sample Output 1:**

hi tsample

### **Sample Input 2:**

ABC DEF

### **Sample Output 2:**

ABC DEF

### **UserMainCode**

```
import java.util.HashSet;  
import java.util.Iterator;  
import java.util.LinkedHashSet;
```

```
import java.util.StringTokenizer;

public class UserMainCode

{
    public static void removeDuplicates(String s1) {
        char a[]={s1.toCharArray()};

        StringBuffer sb=new StringBuffer();

        LinkedHashSet<Character>hs=new LinkedHashSet<Character>();

        for(int i=0;i<a.length;i++)
        {
            hs.add(a[i]);
        }

        Iterator<Character>itr=hs.iterator();

        while(itr.hasNext())
        {
            char o=itr.next();

            if(o!=' ')
            {
                sb.append(o);
            }
        }

        System.out.println(sb);
    }
}
```

## Main

```
import java.util.*;
```

```

public class Main {

public static void main(String[] args)

{

    Scanner s=new Scanner(System.in);

    String s1=s.nextLine();

    UserMainCode.removeDuplicates(s1);

    s.close();

}

}

```

## **8. Mastering Hashmap**

You have recently learnt about hashmaps and in order to master it, you try and use it in all of your programs.

Your trainer / teacher has given you the following exercise:

1. Read  $2n$  numbers as input where the first number represents a key and second one as value. Both the numbers are of type integers.
2. Write a function **getAverageOfOdd** to find out average of all values whose keys are represented by odd numbers.

Assume the average is an int and never a decimal number. Return the average as output. Include this function in class UserMainCode.

Create a Class Main which would be used to read  $2n$  numbers and build the hashmap. Call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a  $2n+1$  integers. The first integer specifies the value of  $n$  (essentially the hashmap size). The next pair of  $n$  numbers denote the key and value.

Output consists of an integer representing the average.

Refer sample output for formatting specifications.

### **Sample Input 1:**

4

2

34

1

4

5

12

4

22

**Sample Output 1:**

8

**UserMainCode**

```
import java.util.HashMap;
import java.util.Scanner;
import java.util.HashMap;
import java.util.Iterator;
public class UserMainCode {
    public static int getAverageOfOdd(HashMap<Integer, Integer> h1)
    {
        int av=0, c=0, s=0;
        Iterator<Integer> it=h1.keySet().iterator();
        while(it.hasNext())
        {
            int a=it.next();
            if(a%2!=0)
            {
                int b=h1.get(a);
```

```

s=s+b;

c++;
}

}

av=s/c;

return av;

}}



Main

import java.util.*;

public class Main

{

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

int n=sc.nextInt();

HashMap<Integer,Integer> h1=new HashMap<Integer,Integer>();

for(int i=0;i<n;i++)

{

h1.put(sc.nextInt(),sc.nextInt());

}

System.out.println(UserMainCode.getAverageOfOdd(h1));

sc.close();

}

}

```

## 9. Managers & Hashmaps

A Company wants to automate its payroll process. You have been assigned as the programmer to build this package. You would like to showcase your skills by creating a quick prototype. The prototype consists of the following steps:

1. Read Employee details from the User. The details would include id, designation and salary in the given order. The datatype for id is integer, designation is string and salary is integer.
2. You decide to build two hashmaps. The first hashmap contains employee id as key and designation as value, and the second hashmap contains same employee ids as key and salary as value.
3. The company decides to hike the salary of managers by 5000. You decide to write a function **increaseSalaries** which takes the above hashmaps as input and returns a hashmap with only managers id and their increased salary as output. Include this function in class UserMainCode.

Create a Class Main which would be used to read employee details in step 1 and build the two hashmaps.

Call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of employee details. The first number indicates the size of the employees. The next three values indicate the employee id, employee designation and employee salary.

Output consists of a single string. Refer sample output for formatting specifications.

**SampleInput1:**

2

2

programmer

3000

8

manager

50000

**SampleOutput1:**

8

55000

**UserMainCode**

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.HashMap;
public class UserMainCode
{public static HashMap<Integer, Integer>
display(HashMap<Integer, String>hm,HashMap<Integer, Integer>hm1)
{
HashMap<Integer, Integer>hm3=new HashMap<Integer, Integer>();
Iterator<Integer> it=hm.keySet().iterator();
while(it.hasNext())
{
int id=it.next();
String name=hm.get(id);
if(name.equals("manager"))
{int salary=hm1.get(id)+5000;
hm3.put(id,salary);
}
}
return hm3;
}
}
```

```
Main
import java.util.HashMap;
import java.util.Iterator;
import java.util.Scanner;
public class Main {
public static void main(String []args){
```

```

Scanner sc=new Scanner(System.in);

int s=Integer.parseInt(sc.nextLine());

HashMap<Integer,String>hm=new HashMap<Integer,String>();

HashMap<Integer,Integer>hm1=new HashMap<Integer,Integer>();

for(int i=0;i<s;i++)

{

int id=Integer.parseInt(sc.nextLine());

hm.put(id, sc.nextLine());

hm1.put(id,Integer.parseInt(sc.nextLine()));

}

HashMap<Integer,Integer>hm2=new HashMap<Integer,Integer>();

hm2=UserMainCode.display(hm,hm1);

Iterator<Integer> it=hm2.keySet().iterator();

while(it.hasNext())

{

int n=it.next();

int fac=hm2.get(n);

System.out.println(n);

System.out.println(fac);

}

}

}

```

#### **10. Check first and last word**

Write a program to check if the first word and the last word in the input string match.

Include a class **UserMainCode** with a static method “**check**” that accepts a string argument and returns an int. If the first word and the last word in the string match, the method returns the number of characters in the word. Else the method returns the sum of the number of characters in the first word and last word.

Create a class **Main** which would get the input as a String and call the static method **check** present in the UserMainCode.

**Input and Output Format:**

Input consists of a string.

Output is an integer.

**Sample Input 1:**

how are you you are how

**Sample Output 1:**

3

**Sample Input 2:**

how is your child

**Sample Output 2:**

8

**UserMainCode**

```
import java.util.StringTokenizer;

public class UserMainCode

{
    public static int check(String s)

    {
        int count=0;

        String fin="";
        StringTokenizer st=new StringTokenizer(s);

        String ini=st.nextToken();

        while(st.hasMoreTokens())
        {
            fin=st.nextToken();
        }
    }
}
```

```

if(ini.equals(fin))

count=ini.length();

else

count=ini.length()+fin.length();

return count;

}

}

```

## Main

```

import java.util.HashMap;

import java.util.Iterator;

import java.util.Scanner;

public class Main {

public static void main(String []args){

Scanner sc=new Scanner(System.in);

String age=sc.nextLine();

System.out.println(UserMainCode.check(age));

sc.close();

}}

```

## 11. Concatenate Characters

Given an array of Strings, write a program to take the last character of each string and make a new String by concatenating it.

Include a class **UserMainCode** with a static method “**concatCharacter**” that accepts a String array as input and returns the new String.

Create a class **Main** which would get the String array as input and call the static method **concatCharacter** present in the

UserMainCode.

**Input and Output Format:**

The first line of the input consists of an integer n that corresponds to the number of strings in the input string array.

The next n lines of the input consist of the strings in the input string array.

Output consists of a string.

**Sample Input:**

3

ab

a

abcd

**Sample Output:**

Bad

UserMainCode

```
public class UserMainCode
{
    public static String concatCharacter(String[] a)
    {
        StringBuffer sb=new StringBuffer();
        for(int i=0;i<a.length;i++)
            sb.append(a[i].charAt(a[i].length()-1));
        return sb.toString();
    }
}
```

Main

```

import java.util.*;

public class Main

{
    public static void main(String []args){

Scanner sc=new Scanner(System.in);

int s=Integer.parseInt(sc.nextLine());

String []a=new String[s];

for(int i=0;i<s;i++)

{

a[i]=sc.nextLine();

}

System.out.println(UserMainCode.concatCharacter(a));

sc.close();

}
}

```

## 12.Anagram

Write a program to check whether the two given strings are anagrams.

Note: Rearranging the letters of a word or phrase to produce a new word or phrase, using all the original letters exactly once is called Anagram."

Include a class **UserMainCode** with a static method "**getAnagram**" that accepts 2 strings as arguments and returns an int.

The method returns 1 if the 2 strings are anagrams. Else it returns -1.

Create a class **Main** which would get 2 Strings as input and call the static method **getAnagram** present in the UserMainCode.

### **Input and Output Format:**

Input consists of 2 strings. Assume that all characters in the string are lower case letters.

Output consists of a string that is either “Anagrams” or “Not Anagrams”.

**Sample Input 1:**

eleven plus two

twelve plus one

**Sample Output 1:**

Anagrams

**Sample Input 2:**

orchestra

carthorse

**Sample Output 2:**

Anagrams

**Sample Input 3:**

cognizant

technologies

**Sample Output 3:**

Not Anagrams

**UserMainCode**

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Scanner;

public class UserMainCode

{
    public static void getAnagram(String s1, String s2)
    {
        List<Character> l1=new ArrayList<Character>();
        List<Character> l2=new ArrayList<Character>();
        int n1=s1.length();
        int n2=s2.length();
        if(n1!=n2)
            System.out.println("Not Anagrams");
        else
        {
```

```
List<Character> l2=new ArrayList<Character>();

String s3=s1.replace(" ","");
String s4=s2.replace(" ","");
String s5=s3.toUpperCase();
String s6=s4.toUpperCase();

for (int i = 0; i < s5.length(); i++)
{
    l1.add(s5.charAt(i));
}

for (int i = 0; i < s6.length(); i++)
{
    l2.add(s6.charAt(i));
}

Collections.sort(l1);
Collections.sort(l2);

// System.out.println(l1);
// System.out.println(l2);

if(l1.equals(l2))
    System.out.println("Anagram");
else
    System.out.println("Not Anagram");
}
```

## Main

```
import java.util.ArrayList;
```

```

import java.util.Collections;
import java.util.List;
import java.util.Scanner;

public class Main{
    public static void main(String[] args)
    {
        Scanner sc =new Scanner(System.in);
        String s1=sc.nextLine();
        String s2=sc.nextLine();

        UserMainCode.getAnagram(s1,s2);

    }
}

```

### 13.Calculate Meter Reading

Given 2 strings corresponding to the previous meter reading and the current meter reading, write a program to calculate electricity bill.

The input string is in the format ""AAAAAXXXXX".

AAAAA is the meter code and XXXXX is the meter reading.

FORMULA: (XXXXX-XXXXX)\*4

Hint: if AAAAA of input1 and input2 are equal then separate the XXXXX from string and convert to integer. Assume that AAAAA of the 2 input strings will always be equal.

Include a class **UserMainCode** with a static method "**calculateMeterReading**" that accepts 2 String arguments and returns an integer that corresponds to the electricity bill. The 1<sup>st</sup> argument corresponds to the previous meter reading and the 2<sup>nd</sup> argument corresponds to the current meter reading.

Create a class **Main** which would get 2 Strings as input and call the static method **calculateMeterReading** present in the **UserMainCode**.

#### **Input and Output Format:**

Input consists of 2 strings. The first input corresponds to the previous meter reading and the second input corresponds to the current meter reading.

Output consists of an integer that corresponds to the electricity bill.

#### **Sample Input:**

CSECE12390

CSECE12400

#### **Sample Output:**

40

```

import java.util.Scanner;
public class Main{
    public static void main (String[] args)
    {
        // your code goes here
        Scanner sc = new Scanner(System.in);
    }
}

```

```

String input1=sc.next();
String input2=sc.next();
System.out.println(UserMainCode.calculateMeterReading(input1,input2));
sc.close();
}

public class UserMainCode {
    public static int calculateMeterReading(String input1, String input2)
    {
        int n1=Integer.parseInt(input1.substring(5,input1.length()));
        int n2=Integer.parseInt(input2.substring(5,input2.length()));
        int n=Math.abs((n2-n1)*4);
        return n;
    }
}

```

#### 14.Retirement

Given an input as HashMap which contains key as the ID and dob as value of employees, write a program to find out employees eligible for retirement. A person is eligible for retirement if his age is greater than or equal to 60.

Assume that the current date is 01/01/2014.

Include a class **UserMainCode** with a static method “retirementEmployeeList” that accepts a `HashMap<String, String>` as input and returns a `ArrayList<String>`. In this method, add the Employee IDs of all the retirement eligible persons to list and return the sorted list.

(Assume date is in dd/MM/yyyy format).

Create a class **Main** which would get the `HashMap` as input and call the static method **retirementEmployeeList** present in the `UserMainCode`.

#### Input and Output Format:

The first line of the input consists of an integer `n`, that corresponds to the number of employees.

The next 2 lines of the input consists of strings that correspond to the id and dob of employee 1.

The next 2 lines of the input consists of strings that correspond to the id and dob of employee 2.

and so on...

Output consists of the list of employee ids eligible for retirement in sorted order.

#### Sample Input :

```

4
C1010
02/11/1987
C2020
15/02/1980
C3030
14/12/1952
T4040
20/02/1950

```

#### Sample Output :

[C3030, T4040]

```

import java.text.ParseException;
import java.util.LinkedHashMap;
import java.util.Scanner;
public class Main
{
    public static void main(String args[]) throws ParseException{ Scanner sc=new
    Scanner(System.in); int
    n=Integer.parseInt(sc.nextLine());
    LinkedHashMap<String, String>a1=new LinkedHashMap<String, String>(); for(int
    i=0;i<n;i++) {
    a1.put(sc.nextLine(),sc.nextLine());
    }
    System.out.println(UserMainCode.retirementEmployeeList(a1));
    }
}
import java.text.*;
import java.util.*;
public class UserMainCode {

```

```

public static ArrayList<String>
retirementEmployeeList(LinkedHashMap<String, String>a1) throws ParseException
{
    ArrayList<String>a1=new ArrayList<String>();
    Iterator <String>it=a1.keySet().iterator();
    while(it.hasNext())
    {String s=it.next();
    String s1=a1.get(s);
    SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
    sdf.setLenient(false);
    try{
    Date d=new Date();
    Date d1=new Date();
    String a=s1;
    String b="01/01/2014";
    d=sdf.parse(a);
    d1=sdf.parse(b);
    long t=d.getTime();
    long t1=d1.getTime();
    long t3=t1-t;
    long l1=(24 * 60 * 60 * 1000);
    long l=l1*365;
    long res=t3/l;
    if(res>=60)
    {
    a1.add(s);
    }
    }
    catch (Exception e) {
    e.printStackTrace();
    }
    }
    Collections.sort(a1);
    return a1;
}
}

```

## 15.Kaprekar Number

Write a program to check whether the given input number is a Kaprekar number or not.

**Note :** A positive whole number ‘n’ that has ‘d’ number of digits is squared and split into two pieces, a right-hand piece that has ‘d’ digits and a left-hand piece that has remaining ‘d’ or ‘d-1’ digits. If the sum of the two pieces is equal to the number, then ‘n’ is a Kaprekar number.

If its Kaprekar number assign to output variable 1 else -1.

Example 1:

Input1:9

$9^2 = 81$ , right-hand piece of 81 = 1 and left hand piece of 81 = 8

$Sum = 1 + 8 = 9$ , i.e. equal to the number. Hence, 9 is a Kaprekar number.

Example 2:

Input1:45

Hint:

$45^2 = 2025$ , right-hand piece of 2025 = 25 and left hand piece of 2025 = 20

$Sum = 25 + 20 = 45$ , i.e. equal to the number. Hence, 45 is a Kaprekar number."

Include a class **UserMainCode** with a static method "**getKaprekarNumber**" that accepts an integer argument and returns an integer. The method returns 1 if the input integer is a Kaprekar number. Else the method returns -1.

Create a class **Main** which would get the an Integer as input and call the static method **getKaprekarNumber** present in the **UserMainCode**.

### Input and Output Format:

Input consists of an integer.

Output consists of a single string that is either “Kaprekar Number” or “Not A Kaprekar Number”

**Sample Input 1:**

```

9
Sample Output 1:
Kaprekar Number
Sample Input 2:
45
Sample Output 2:
Kaprekar Number
Sample Input 3:
4
Sample Output 3:
Not A Kaprekar Number

import java.util.*;
public class Main{
public static void main(String[] args)
{
Scanner sc=new Scanner(System.in);
int n=sc.nextInt();
int v=UserMainCode.getKaprekarNumber(n);
if (v==1)
System.out.println("Kaprekar Number");
else
System.out.println("Not a Kaprekar Number");
}}

public class UserMainCode {

    public static int getKaprekarNumber(int a)
    {
        int count=0,j=0;
        int a1=a;
        while(a1!=0)
        {
            count=count+1;
            a1=a1/10;
        }
        int square=a*a;
        String s=Integer.toString(square);
        String s1=s.substring(0,count);
        String s2=s.substring(count);
        int x=Integer.parseInt(s1);
        int y=Integer.parseInt(s2);
        int result =x+y;
        if(result==a){
            j=1;
        }
        else
        {
            j=2;
        }
        return j;
    }
}

```

## 16.Vowels

Given a String input, write a program to find the word which has the the maximum number of vowels. If two or more words have the maximum number of vowels, print the first word.

Include a class **UserMainCode** with a static method “**storeMaxVowelWord**” that accepts a string argument and returns the word containing the maximum number of vowels.

Create a class **Main** which would get the a String as input and call the static method **storeMaxVowelWord** present in the **UserMainCode**.

**Input and Output Format:**

Input consists of a string. The string may contain both lower case and upper case letters.  
Output consists of a string.

**Sample Input :**

What is your name?

**Sample Output :**

your

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        String s1 =sc.nextLine();
        UserMainCode.storeMaxVowelWord(s1);

    }
}

import java.util.StringTokenizer;

public class UserMainCode {
    public static void storeMaxVowelWord(String s1) {
        int i = 0;
        StringTokenizer st = new StringTokenizer(s1, " ");
        int len = 0;
        int count = 0;
        int count2 = 0;
        String s6 = null;
        while (st.hasMoreTokens()) {
            String s5 = st.nextToken();
            len = s5.length();
            count=0;
            for (i = 0; i < len; i++) {
                if (s5.charAt(i) == 'a' || s5.charAt(i) == 'e'|| s5.charAt(i) == 'i'
                || s5.charAt(i) == 'o'|| s5.charAt(i) == 'u'
                ||s5.charAt(i) == 'A' ||s5.charAt(i) == 'E' ||s5.charAt(i) == 'I'
                ||s5.charAt(i) == 'O' ||s5.charAt(i) == 'U')
                    count++;
            }
            if (count > count2)
            {
                count2 = count;
                s6 = s5;
            }
        }
        System.out.println(s6);
    }
}
```

### 17.Unique Characters REPEATED

Given a String as input , write a program to count and print the number of unique characters in it.

Include a class **UserMainCode** with a static method “**checkUnique**” that accepts a String as input and returns the number of unique characters in it. If there are no unique characters in the string, the method returns -1.

Create a class **Main** which would get a String as input and call the static method **checkUnique** present in the UserMainCode.

**Input and Output Format:**

Input consists of a string.

Output consists of an integer.

**Sample Input 1:**

HOWAREYOU

**Sample Output 1:**

(Hint :Unique characters are : H,W,A,R,E,Y,U and other characters are repeating)

**Sample Input 2:**

MAMA

**Sample Output2:**

-1

```

import java.util.*;
public class Main
{
public static void main(String[] args)
{
    Scanner sc=new Scanner(System.in);
String s1=sc.next();
UserMainCode.checkUnique(s1);
}
}

import java.util.*;
public class UserMainCode {

    public static void checkUnique(String s1)
    {
        String s2=s1.toLowerCase();
        StringBuffer sb=new StringBuffer(s2);
        int l=sb.length();
        int count=0;
        for(int i=0;i<l;i++)
        { count=0;
        for(int j=i+1;j<l;j++)
        {
            if(sb.charAt(i)==sb.charAt(j))
            {
                sb.deleteCharAt(j);
                count++;
                j--;
                l--;
            }
        }
        if(count>0)
        {
            sb.deleteCharAt(i);
            i--;
            l--;
        }
    }
    if(sb.length()==0)
    {
        System.out.println(-1);
    }
    else
        System.out.println(sb.length());
    }
}

```

## 18.average of primes

Write a program to read an array and find average of all elements located at index i, where i is a prime number. Type cast the average to an int and return as output. The index starts from 0.

Include a class UserMainCode with a static method **addPrimeIndex** which accepts a single integer array. The return type (integer) should be the average of all elements located at index i where i is a prime number.

Create a Class Main which would be used to accept Input array and call the static method present in UserMainCode.

### Input and Output Format:

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

Output consists of a single Integer.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20 and minimum number of elements is 3.

### Sample Input 1:

```
4  
2  
5  
2  
4
```

### Sample Output 1:

```
3
```

```
import java.util.Scanner;  
public class Main{  
    public static void main (String[] args)  
{  
    // your code goes here  
    Scanner sc = new Scanner(System.in);  
    int n = sc.nextInt();  
    System.out.println(UserMainCode.addPrimeIndex(n));  
}  
import java.util.*;  
public class UserMainCode {  
    public static int addPrimeIndex(int n) {  
        Scanner sc=new Scanner(System.in);  
        int[] a = new int[n];  
        for(int i=0;i<n;i++){  
            a[i] = sc.nextInt();  
        }  
        int sum=0;  
        int count=0;  
        int sum_count=0;  
        for(int i=0;i<a.length;i++)  
        {  
            count=0;  
            for(int j=1;j<=i;j++)  
            {  
                if(i%j==0)  
                {  
                    count++;  
                }  
            }  
            if(count==2)  
            {  
                sum=sum+a[i];  
                sum_count++;  
            }  
        }  
        int avg=sum/sum_count;  
        return avg;  
}}
```

## 19. ArrayList and Set Operations

Write a program that performs the following actions:

1. Read  $2n$  integers as input & a set operator (of type char).
2. Create two arraylists to store  $n$  elements in each arraylist.
3. Write a function **performSetOperations** which accepts these two arraylist and the set operator as input.
4. The function would perform the following set operations:

'+' for SET-UNION

'\*' for SET-INTERSECTION

'-' for SET-DIFFERENCE

Refer to sample inputs for more details.

5. Return the resultant arraylist.

Include a class UserMainCode with the static method **performSetOperations** which accepts two arraylist and returns an arraylist.

Create a Class Main which would be used to read  $2n+1$  integers and call the static method present in UserMainCode.

Note:

- The index of first element is 0.

### **Input and Output Format:**

Input consists of  $2n+2$  integers. The first integer denotes the size of the arraylist, the next  $n$  integers are values to the first arraylist, and the next  $n$  integers are values to the second arraylist and the last input corresponds to that set operation type.

Output consists of a modified arraylist as per step 4.

Refer sample output for formatting specifications.

#### **Sample Input 1:**

```
3  
1  
2  
3  
3  
5  
7  
+
```

#### **Sample Output 1:**

```
1  
2  
3  
5  
7
```

#### **Sample Input 2:**

```
4  
10  
9  
8  
7  
2  
4  
6  
8  
*
```

#### **Sample Output 2:**

```
8
```

#### **Sample Input 3:**

```
4  
5  
10  
15  
20  
0  
10  
12  
20  
-
```

#### **Sample Output 3:**

```
5  
15
```

#### **Main:**

```

import java.util.ArrayList;
import java.util.Scanner;
public class Main {
    public static void main(String args[]){
        Scanner sc = new Scanner(System.in);
        int n=Integer.parseInt(sc.nextLine());
        ArrayList<Integer>a1=new ArrayList<Integer>();
        ArrayList<Integer>a2=new ArrayList<Integer>();
        for(int i=0;i<n;i++)
            a1.add(Integer.parseInt(sc.nextLine()));
        for(int i=0;i<n;i++)
            a2.add(Integer.parseInt(sc.nextLine()));
        char c=sc.nextLine().charAt(0);
        System.out.println(UserMainCode.performSetOperations(a1,a2,c));
    }
}

```

**UserMainCode:**

```

import java.util.ArrayList;
import java.util.ArrayList;
public class UserMainCode {
    public static ArrayList<Integer>
    performSetOperations(ArrayList<Integer>a1,ArrayList<Integer>a2,char c)
    {
        ArrayList<Integer>op1=new ArrayList<Integer>();
        int k=0;
        switch(c)
        {
            case '+':
                a1.removeAll(a2);

```

```
a1.addAll(a2);
op1=a1;
break;
case '*':
a1.retainAll(a2);
op1=a1;
break;
case '-':
for(int i=0;i<a1.size();i++)
{
k=0;
for(int j=0;j<a2.size();j++)
{
if(a1.get(i)==a2.get(j))
k=1;
}
if(k==0)
op1.add(a1.get(i));
}
break;
}
return op1;
}}
}

return tm;
```

```
}
```

## 20.Largest Span

Write a program to read an array and find the size of largest span in the given array  
""span"" is the number of elements between two repeated numbers including both numbers. An array with single element has a span of 1.

Include a class UserMainCode with a static method **getMaxSpan** which accepts a single integer array. The return type (integer) should be the size of largest span.

Create a Class Main which would be used to accept Input array and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of n+1 integers. The first integer corresponds to n, the number of elements in the array. The next 'n' integers correspond to the elements in the array.

Output consists of a single Integer.

Refer sample output for formatting specifications.

Assume that the maximum number of elements in the array is 20.

### **Sample Input 1:**

```
5
1
2
1
1
3
```

### **Sample Output 1:**

```
4
```

### **Sample Input 2:**

```
7
1
4
2
1
4
1
5
```

### **Sample Output 2:**

```
6
```

### **MAIN:**

```
import java.util.Scanner;
public class Main {
public static void main(String[] args)
{
Scanner sc=new Scanner(System.in);
int n=sc.nextInt();
int []a=new int[n];
for(int i=0;i<n;i++)
{
a[i]=sc.nextInt();
}
System.out.print(UserMainCode.getMaxSpan(a,n));
}}
```

### **USERMAINCODE:**

```
class UserMainCode {
public static int getMaxSpan(int[] x,int n)
{
int gap=0,max=0;
for(int i=0;i<n;i++)
{
for(int j=i+1;j<n;j++)
{
if(x[i]==x[j])
```

```

gap=j;
}
if(gap-i>max)
max=gap-i;
}
return max+1;
}
}

```

## 21. Max Scorer

Write a program that performs the following actions:

1. Read n strings as input and stores them as an arraylist. The string consists of student information like name and obtained marks of three subjects. Eg: name-mark1-mark2-mark3 [suresh-70-47-12]. The mark would range between 0 – 100 (inclusive).
2. Write a function **highestScorer** which accepts these arraylist and returns the name of the student who has scored the max marks. Assume the result will have only one student with max mark.

Include a class UserMainCode with the static method **highestScorer** which accepts the arraylist and returns the name

(string) of max scorer.

Create a Class Main which would be used to read n strings into arraylist and call the static method present in UserMainCode.

### Input and Output Format:

Input consists of 1 integer and n strings. The first integer denotes the size of the arraylist, the next n strings are score pattern described above.

Output consists of a string with the name of the top scorer.

Refer sample output for formatting specifications.

### Sample Input 1:

```

3
sunil-56-88-23
bindul-88-70-10
john-70-49-65

```

### Sample Output 1:

John

### USERMAINCODE:

```

import java.util.ArrayList;
import java.util.StringTokenizer;
public class UserMainCode
{
public static String highestScorer(ArrayList<String>s1)
{
int max=0;
String s4=null;
for(int i=0;i<s1.size();i++)
{
    String s2=s1.get(i);
    StringTokenizer t=new StringTokenizer(s2, "-");

```

```

        String s3=t.nextToken();
        int n1=Integer.parseInt(t.nextToken());
        int n2=Integer.parseInt(t.nextToken());
        int n3=Integer.parseInt(t.nextToken());
        int n=n1+n2+n3;
        if(n>max)
        {
            max=n;
            s4=s3;
        }
    }
    return s4;
}
}

```

### **MAIN:**

```

import java.util.*;
public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
Scanner s=new Scanner(System.in);
int n=s.nextInt();
ArrayList<String> s1=new ArrayList<String>();
for(int i=0;i<n;i++)
{
    s1.add(s.next());
}
System.out.println(UserMainCode.highestScorer(s1));
s.close();

    }
}

```

## **22. Max Vowels**

Write a Program which fetches the word with maximum number of vowels. Your program should read a sentence as input from user and return the word with max number of vowels. In case there are two words of maximum length return the word which comes first in the sentence.

Include a class UserMainCode with a static method **getWordWithMaximumVowels** which accepts a string The return type is the longest word of type string.

Create a Class Main which would be used to accept two Input strings and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string with maximum size of 100 characters.

Output consists of a single string.

Refer sample output for formatting specifications.

### **Sample Input 1:**

Appreciation is the best way to motivate

### **Sample Output 1:**

Appreciation

### USERMAINCODE:

```
import java.util.StringTokenizer;

public class UserMainCode {
    public static String getWordWithMaximumVowels(String s1)
    {
        int i;
        StringTokenizer t=new StringTokenizer(s1, " ");
        int count=0,max=0;
        String s2=null;
        while(t.hasMoreTokens())
        {
            String s3=t.nextToken();
            count=0;
            for(i=0;i<s3.length();i++)
            {

                if(s3.charAt(i)=='a' || s3.charAt(i)=='e' || s3.charAt(i)=='i' || s3.charAt(i)=='o' || s3.charAt(i)=='u'
                   || s3.charAt(i)=='A' || s3.charAt(i)=='E' || s3.charAt(i)=='I' || s3.charAt(i)=='O'
                   || s3.charAt(i)=='U')
                    count++;

            }
            if(count>max)
            {
                max=count;
                s2=s3;
            }
        }
        return s2;
    }
}
```

### MAIN:

```
import java.util.*;
public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        Scanner s=new Scanner(System.in);
        String s1=s.nextLine();
        System.out.println(UserMainCode.getWordWithMaximumVowels(s1));
        s.close();
    }
}
```

## 23. All Vowels

Write a Program to check if given word contains exactly five vowels and the vowels are in alphabetical order. Return 1 if the condition is satisfied else return -1. Assume there is no repetition of any vowel in the given string and all letters are in lower case.

Include a class UserMainCode with a static method **testOrderVowels** which accepts a string. The return type is integer based on the condition stated above.

Create a Class Main which would be used to accept two Input strings and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of a string with maximum size of 100 characters.

Output consists of a single string.

Refer sample output for formatting specifications.

**Sample Input 1:**

acebisouzz

**Sample Output 1:**

valid

**Sample Input 2:**

Alphabet

**Sample Output 2:**

Invalid

**USERMAINCODE:**

```
public class UserMainCode {  
    public static int getOrderVowels(String s1)  
{  
  
        String s2="aeiou";  
        StringBuffer sb=new StringBuffer();  
        for(int i=0;i<s1.length();i++)  
        {  
            for(int j=0;j<s2.length();j++)  
            {  
                if(s1.charAt(i)==s2.charAt(j))  
                {  
                    sb.append(s1.charAt(i));  
                }  
            }  
        }  
        if(sb.toString().equals(s2))  
        {  
            return 1;  
        }  
        return -1;  
    }  
}
```

**MAIN:**

```
import java.util.*;  
public class Main {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        Scanner s=new Scanner(System.in);  
        String s1=s.next();
```

```

int b=UserMainCode.getOrderVowels(s1);
if(b==1)
{
    System.out.println("Valid");
}
else
    System.out.println("Invalid");
s.close();
}

}

```

## 24. Adjacent Swaps

Write a Program that accepts a string as a parameter and returns the string with each pair of adjacent letters reversed. If the string has an odd number of letters, the last letter is unchanged.

Include a class UserMainCode with a static method **swapPairs** which accepts a string. The return type is string which is reversed pair of letters.

Create a Class Main which would be used to accept two Input strings and call the static method present in UserMainCode.

**Input and Output Format:**

Input consists of a string with maximum size of 100 characters.

Output consists of a single string.

Refer sample output for formatting specifications.

**Sample Input 1:**

forget

**Sample Output 1:**

ofgrte

**Sample Input 2:**

New York

**Sample Output 2:**

eN woYkr

**USERMAINCODE:**

```

import java.util.*;
public class UserMainCode {
public static String swapPairs(String s1)
{
    int i;
    StringBuffer sb=new StringBuffer();
    for(i=0;i<s1.length()-1;i=i+2)
    {
        if(i%2==0)
        {
            char a=s1.charAt(i);
            char b=s1.charAt(i+1);
            sb.append(b).append(a);
        }
        else
            for(i=0;i<s1.length()-1;i=i+2)
        {
            char a=s1.charAt(i);
            char b=s1.charAt(i+1);

```

```

        sb.append(b).append(a);
        sb.append(s1.charAt(s1.length()-1));
    }

}

return sb.toString();
}
}

MAIN:

import java.util.*;
public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
Scanner s=new Scanner(System.in);
String s1=s.nextLine();
System.out.println(UserMainCode.swapPairs(s1));
s.close();
    }

}

```

## 25. Sum of Digits

Write a Program that accepts a word as a parameter, extracts the digits within the string and returns its sum.

Include a class UserMainCode with a static method **getdigits** which accepts a string. The return type is integer representing the sum.

Create a Class Main which would be used to accept the input string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string with maximum size of 100 characters.

Output consists of a single string.

Refer sample output for formatting specifications.

### **Sample Input 1:**

abc12de4

### **Sample Output 1:**

7

### **USERMAINCODE:**

```

public class UserMainCode {
public static int getDigits(String s1)
{
    int sum=0;
    for(int i=0;i<s1.length();i++)
    {
        char a=s1.charAt(i);
        if(Character.isDigit(a))
        {
            int b=Integer.parseInt(String.valueOf(a));
            sum=sum+b;
        }
    }
}

```

```

        }
        return sum;
    }
}

MAIN:

import java.util.*;
public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
Scanner s=new Scanner(System.in);
String s1=s.next();
System.out.println(UserMainCode.getDigits(s1));
s.close();
    }

}

```

## 26. Password

Given a String , write a program to find whether it is a valid password or not.

Validation Rule:

Atleast 8 characters

Atleast 1 number(1,2,3...)

Atleast 1 special character(@,#,%...)

Atleast 1 alphabet(a,B...)

Include a class **UserMainCode** with a static method “**validatePassword**” that accepts a String argument and returns a boolean value. The method returns true if the password is acceptable. Else the method returns false.

Create a class **Main** which would get a String as input and call the static method **validatePassword** present in the UserMainCode.

### **Input and Output Format:**

Input consists of a String.

Output consists of a String that is either “Valid” or “Invalid”.

#### **Sample Input 1:**

cts@1010

#### **Sample Output 1:**

Valid

#### **Sample Input 2:**

punitha3

**Sample Output 2:**

Invalid

**USERMAINCODE:**

```
public class UserMainCode {  
  
    public static boolean validatePassword(String s1)  
  
    {  
  
        boolean b=false;  
  
        if(s1.length()>=8)  
  
            b=true;  
  
        if(b=true)  
  
        {  
  
            if(s1.matches(".*[0-9]{1,}.*")&&s1.matches(".*[a-zA-Z]{1,}.*")&&s1.matches(".*[@#%]{1,}.*"))  
  
            {  
  
                b=true;  
  
            }  
  
        else  
  
            b=false;  
  
        }  
  
        return b;  
  
    }  
  
}
```

**MAIN:**

```
import java.util.*;  
  
public class Main {  
  
    public static void main(String [] args)  
  
    {  
  
        Scanner s=new Scanner(System.in);
```

```

String s1=s.nextLine();

boolean b=(UserMainCode.validatePassword(s1));

if(b==true)
{
    System.out.println("Valid");

}
else

System.out.println("Invalid");

s.close();
}

}

```

## **27. Employee Bonus**

A Company wants to give away bonus to its employees. You have been assigned as the programmer to automate this process. You would like to showcase your skills by creating a quick prototype. The prototype consists of the following steps:

1. Read Employee details from the User. The details would include id, DOB (date of birth) and salary in the given order. The datatype for id is integer, DOB is string and salary is integer.

2. You decide to build two hashmaps. The first hashmap contains employee id as key and DOB as value, and the second hashmap contains same employee ids as key and salary as value.

3. If the age of the employee in the range of 25 to 30 years (inclusive), the employee should get bonus of 20% of his salary and in the range of 31 to 60 years (inclusive) should get 30% of his salary. store the result in TreeMap in which Employee ID as key and revised salary as value. Assume the age is caculated based on the date 01-09-2014.

(Typecast the bonus to integer).

4. Other Rules:

a. If Salary is less than 5000 store -100.

b. If the age is less than 25 or greater than 60 store -200.

c. a takes more priority than b i.e both if a and b are true then store -100.

5. You decide to write a function **calculateRevisedSalary** which takes the above hashmaps as input and returns the treemap as output. Include this function in class UserMainCode.

Create a Class Main which would be used to read employee details in step 1 and build the two hashmaps. Call the static

method present in UserMainCode.

**Input and Output Format:**

Input consists of employee details. The first number indicates the size of the employees. The next three values indicate the employee id, employee DOB and employee salary. The Employee DOB format is “dd-mm-yyyy”

Output consists of a single string.

Refer sample output for formatting specifications.

**Sample Input 1:**

2

1010

20-12-1987

10000

2020

01-01-1985

14400

**Sample Output 1:**

1010

12000

2020

17280

**USERMAINCODE:**

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.TreeMap;
class UserMainCode {
    public static TreeMap<Integer,Integer>
    calculateRevisedSalary(LinkedHashMap<Integer,String>a1,LinkedHashMap<Integer,Integer>a2) throws
    ParseException
    {TreeMap<Integer,Integer>ans=new TreeMap<Integer,Integer>();
    ArrayList<String>al=new ArrayList<String>();
    Iterator <Integer>it=a1.keySet().iterator();
    while(it.hasNext())
    {int s=it.next();
    String s1=a1.get(s);
    SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");
    sdf.setLenient(false);
    try{
```

```

Date d=new Date();
Date d1=new Date();
String a=s1;
String b="01-09-2014";
d=sdf.parse(a);
d1=sdf.parse(b);
long t=d.getTime();
long t1=d1.getTime();
long t3=t1-t;
long l1=(24 * 60 * 60 * 1000);
long l=l1*365;
long res=t3/l;
//System.out.println("Result="+res);
if(res>=25 && res<=30)
{
float bonus=(float)((0.2*a2.get(s))+a2.get(s));
int r=(int)bonus;
ans.put(s,r );
}
else if(res>30 && res<=60)
{
float bonus=(float)((0.3*a2.get(s))+a2.get(s));
int r=(int)bonus;
ans.put(s,r );
}
else if(a2.get(s)<5000)
{
ans.put(s, -100);
}
else if(res<25 ||res>60)
{
ans.put(s, -200);
}
}
catch (Exception e) {
e.printStackTrace();
}
}
return ans;
}
}

```

## **MAIN:**

```

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.Iterator;
import java.util.LinkedHashMap;
import java.util.Scanner;
import java.util.TreeMap;
public class Main
{
public static void main(String args[]) throws ParseException{
Scanner sc=new Scanner(System.in);
int n=sc.nextInt();
LinkedHashMap<Integer,String>a1=new LinkedHashMap<Integer,String>();
LinkedHashMap<Integer,Integer>a2=new LinkedHashMap<Integer,Integer>();
TreeMap<Integer,Integer>ans=new TreeMap<Integer, Integer>();

```

```

for(int i=0;i<n;i++)
{
int id=sc.nextInt();
a1.put(id,sc.nextInt());
int salary=sc.nextInt();
a2.put(id,salary);
}
ans=UserMainCode.calculateRevisedSalary(a1,a2);
Iterator <Integer>it=ans.keySet().iterator();
while(it.hasNext())
{
int a=it.next();
int b=ans.get(a);
System.out.println(a);
System.out.println(b);
}
}
}
}

```

## **28. Grade Calculator REFER 53 FROM LEVEL2**

A School wants to assign grades to its students based on their marks. You have been assigned as the programmer to automate this process. You would like to showcase your skills by creating a quick prototype. The prototype consists of the following steps:

1. Read student details from the User. The details would include roll no, mark in the given order. The datatype for id is integer, mark is integer.

2. You decide to build a hashmap. The hashmap contains roll no as key and mark as value.

### **3. BUSINESS RULE:**

1. If Mark is greater than or equal to 80 store medal as ""GOLD"".

2. If Mark is less than 80 and greater than or equal to 60 store medal as ""SILVER"".

3 .If Mark is less than 60 and greater than or equal to 45 store medal as ""BRONZE"" else store ""FAIL"".

Store the result in TreeMap in which Roll No as Key and grade as value.

4. You decide to write a function **calculateGrade** which takes the above hashmap as input and returns the treemap as output. Include this function in class UserMainCode.

Create a Class Main which would be used to read employee details in step 1 and build the two hashmap. Call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of employee details. The first number indicates the size of the students. The next two values indicate the roll id, mark.

Output consists of a single string.

Refer sample output for formatting specifications.

**Sample Input 1:**

2

1010

80

100

40

**Sample Output 1:**

100

FAIL

1010

GOLD

**USERMAINCODE:**

```
import java.util.Iterator;
import java.util.HashMap;
import java.util.TreeMap;
public class UserMainCode
{
    public static TreeMap<Integer,String>calculateGrade(HashMap<Integer,Integer>hm)
    {
        TreeMap<Integer,String>tm=new TreeMap<Integer,String>();
        Iterator<Integer> it=hm.keySet().iterator();
        while(it.hasNext())
        {
            int id=it.next();
            int mark=hm.get(id);
            if(mark>=80)
                tm.put(id,"GOLD");
            else if(mark<80 && mark>=60)
                tm.put(id,"SILVER");
            else if(mark<60 && mark>=45)
                tm.put(id,"BRONZE");
            else
                tm.put(id,"FAIL");
        }
        return tm;
    }
}
```

**MAIN:**

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.HashMap;
import java.util.TreeMap;
import java.util.Scanner;
public class Main {
    public static void main(String []args){
        Scanner sc=new Scanner(System.in);
```

```

int s=sc.nextInt();
HashMap<Integer,Integer>hm=new HashMap<Integer,Integer>();
for(int i=0;i<s;i++)
{
hm.put(sc.nextInt(),sc.nextInt());
}
TreeMap<Integer,String>tm=new TreeMap<Integer,String>();
tm=UserMainCode.calculateGrade(hm);
Iterator<Integer> it=tm.keySet().iterator();
for(int i=0;i<s;i++)
{
int n=it.next();
String fac=tm.get(n);
System.out.println(n);
System.out.println(fac);
}
}
}

```

## 29.Digits - II

Write a program to read a non-negative integer n, compute the sum of its digits. If sum is greater than 9 repeat the process and calculate the sum once again until the final sum comes to single digit.Return the single digit. Include a class UserMainCode with a static method getDigitSum which accepts the integer value. The return type is integer. Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

**Input and Output Format:** Input consists of a integer. Output consists of integer. Refer sample output for formatting specifications.

Sample Input 1: 9999 Sample Output 1: 9

Sample Input 2: 698 Sample Output 2: 5

### **MAIN:**

```

import java.util.Scanner;

public class Main {

    public static void main(String[] args)

    {
        Scanner s=new Scanner(System.in);

```

```
int a=s.nextInt();

int sum=UserMainCode.getDigitSum(a);

System.out.println(sum);

}

}
```

### **USERMAINCODE:**

```
public class UserMainCode

{

public static int getDigitSum(int n)

{

    int sum = 0 ;

    int n1=n;

    while(n>10)

    {

        int a = 0 ; sum = 0;

        while(n!=0)

        {

            a = n%10;

            sum+=a;

            n=n/10;

        }

        n=sum;

    }

}
```

```
    return sum;  
}  
  
}
```

### 30.Anagrams

Write a program to read two strings and checks if one is an anagram of the other. An anagram is a word or a phrase that can be created by rearranging the letters of another given word or phrase. We ignore white spaces and letter case. All letters of 'Desperation' can be rearranged to the phrase 'A Rope Ends It'. Include a class UserMainCode with a static method checkAnagram which accepts the two strings. The return type is boolean which is TRUE / FALSE. Create a Class Main which would be used to accept the two strings and call the static method present in UserMainCode.

**Input and Output Format:** Input consists of two strings. Output consists of TRUE / FALSE. Refer sample output for formatting specifications. Sample Input 1: tea eat Sample Output 1: TRUE

Sample Input 2: Desperation A Rope Ends It Sample Output 2: TRUE

### **MAIN:**

```
import java.util.*;  
  
public class Main {  
  
    public static void main(String[] args){  
  
        Scanner s=new Scanner(System.in);  
  
        String s1=s.nextLine();  
  
        String s2=s.nextLine();  
  
        if(s1.equals(s2))  
            System.out.println("TRUE");  
        else  
            System.out.println("FALSE");  
    }  
}
```

```
        System.out.println(UserMainCode.checkAnagram(s1,s2));  
  
    }  
  
}
```

### **USERMAINCODE:**

```
import java.util.*;  
  
import java.text.*;  
  
public class UserMainCode {  
  
    public static boolean checkAnagram(String s1,String s2)  
  
    {  
  
        boolean b=false;  
  
  
        String aj1 =s1.toLowerCase(); //ANAGRAMS  
        String aj2=s2.toLowerCase();  
  
  
        ArrayList<Character> a1 = new ArrayList<Character>();  
        ArrayList<Character> a2 = new ArrayList<Character>();  
  
        for(int i=0;i<aj1.length();i++)  
  
        {  
  
            char c=aj1.charAt(i);  
  
            if(c!=' ')  
  
            {
```

```

    a1.add(c);

}

}

for(int j=0;j<aj2.length();j++)

{
    char c=aj2.charAt(j);

    if(c!=' ')
    {
        a2.add(c);
    }
}

if(a1.size()==a2.size())

{
    if(a1.containsAll(a2))

    {
        b= true;
    }
}

return b;
}
}

```

Write a program to read a integer array of scores, and return a version of the given array where all the 5's have been removed. The remaining elements should shift left towards the start of the array as needed,

and the empty spaces at the end of the array should be filled with 0.

So {1, 5, 5, 2} yields {1, 2, 0, 0}.

Include a class UserMainCode with a static method shiftLeft which accepts the integer array. The return type is modified array.

Create a Class Main which would be used to accept the integer array and call the static method present in UserMainCode.

Input and Output Format:

Input consists of an integer n which is the number of elements followed by n integer values.

Output consists of modified array.

Refer sample output for formatting specifications.

Sample Input 1: 7 1 5 2 4 5 3 5

Sample Output 1: 1 2 4 3 0 0 0

### **MAIN:**

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args)

    {

        Scanner sc=new Scanner(System.in);

        int size=sc.nextInt();

        int[]m=new int[size];

        int[]n=new int[size];

        for(int i=0;i<size;i++)

        {
```

```

n[i]=sc.nextInt();

}

m=UserMainCode.shiftLeft(n);

for(int i=0;i<size;i++)

{

    System.out.println(m[i]);

}

}

}

```

### **USERMAINCODE:**

```

public class UserMainCode {

    public static int[] shiftLeft(int n[])
    {
        int j=0;

        int[]m=new int[n.length];

        for(int i=0;i<n.length;i++)

        {

            if(n[i]!=5)

            {

                m[j]=n[i];

                j++;

            }

        }

        return m;
    }
}

```

```
 }  
 }
```

## 32.Word Count

Given a string array (s) with each element in the array containing alphabets or digits. Write a program to add all the digits in every string and return the sum as an integer. If two digits appear simultaneously do not consider it as one number. Ex- For 'Hyderabad 21' consider 2 and 1 as two digits instead of 21 as a number.

Include a class UserMainCode with a static method sumOfDigits which accepts the string array. The return type is the integer formed based on rules. Create a Class Main which would be used to accept the string and integer and call the static method present in UserMainCode.

**Input and Output Format:** Input consists of a an integer indicating the number of elements in the string array. Output consists of a integer . Refer sample output for formatting specifications.

Sample Input 1: 5 AAA1

B2B 4CCC A5 ABCDE Sample Output 1: 12

Sample Input 2: 3 12 C23 5CR2 Sample Output 2: 15

### **MAIN:**

```
import java.util.Scanner;  
  
public class Main {  
  
    public static void main(String[] args)  
    {  
  
        Scanner s=new Scanner(System.in);  
  
        int n=s.nextInt();  
  
        String a[]=new String[n];  
  
        for(int i=0;i<n;i++)  
        {
```

```

        a[i]=s.nextInt();

    }

    System.out.println(UserMainCode.sumOfDigits(a));

}

}

```

### **USERMAINCODE:**

```

public class UserMainCode {

    public static int sumOfDigits(String[] s1)

    {

        int sum = 0;

        for(int i=0;i<s1.length;i++)

        {

            String s = s1[i];

            for(int j = 0;j<s.length();j++)

            {

                Character c = s.charAt(j);

                if(Character.isDigit(c))

                {

                    sum+=Integer.parseInt(s.valueOf(c));

                }

            }

            return sum;

        }

    }

```

### 33.Prefix Finder

Given a string array (s) with each element in the array containing 0s and 1s. Write a program to get the number of strings in the array where one String is getting as prefixed in other String in that array . Example 1: Input: {10,101010,10001,1111} Output =2 (Since 10 is a prefix of 101010 and 10001) Example 2: Input: {010,1010,01,0111,10,10} Output =3(01 is a prefix of 010 and 0111. Also, 10 is a prefix of 1010) Note: 10 is NOT a prefix for 10.

Include a class UserMainCode with a static method findPrefix which accepts the string array. The return type is the integer formed based on rules. Create a Class Main which would be used to accept the string and integer and call the static method present in UserMainCode.

**Input and Output Format:** Input consists of a an integer indicating the number of elements in the string array followed by the array. Output consists of a integer . Refer sample output for formatting specifications.

Sample Input 1: 4 0 1 11 110 Sample Output 1: 3

#### **MAIN:**

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args)

    {

        Scanner sc = new Scanner(System.in);

        int n=Integer.parseInt(sc.nextLine());

        String s[]={};

        for(int i=0;i<n;i++)

            s[i]=sc.nextLine();

        System.out.println(UserMainCode.findPrefix(s));

    }

}
```

#### **USERMAINCODE:**

```
import java.util.ArrayList;

import java.util.Iterator;
```

```
import java.util.LinkedHashSet;
public class UserMainCode
{
    public static int findPrefix (String s[]) {
        LinkedHashSet<String>l1=new LinkedHashSet<String>();
        ArrayList<String>a1=new ArrayList<String>();
        int c=0;
        for(int i=0;i<s.length;i++)
            l1.add(s[i]);
        Iterator<String> it=l1.iterator();
        while(it.hasNext())
        {
            a1.add(it.next());
        }
        for(int i=0;i<a1.size();i++)
        {
            String s2=a1.get(i);
            for(int j=0;j<a1.size();j++)
            {
                String s3=a1.get(j);
                if(i!=j&&s3.length()>s2.length())
                {
                    String s4=s3.substring(0,s2.length());
                    if(s2.equals(s4))
                        c++;
                }
            }
        }
    }
}
```

```

        }
    }
}

return c;
}
}

```

### 34.Commons

Given two arrays of strings, return the count of strings which is common in both arrays. Duplicate entries are counted only once. Include a class UserMainCode with a static method countCommonStrings which accepts the string arrays. The return type is the integer formed based on rules. Create a Class Main which would be used to accept the string arrays and integer and call the static method present in UserMainCode.

**Input and Output Format:** Input consists of a an integer indicating the number of elements in the string array followed by the array. Output consists of a integer . Refer sample output for formatting specifications.

Sample Input 1: 3 a c e 3 b d e Sample Output 1: 1

Sample Input 2: 5 ba ba black sheep wool 5 ba ba have any wool Sample Output 2: 2

### **MAIN:**

```

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n1 = sc.nextInt();
        String[] s1 = new String[n1];
        for (int i = 0; i < n1; i++) {
            s1[i] = sc.next();
        }
        int count = 0;
        for (int i = 0; i < n1; i++) {
            for (int j = i + 1; j < n1; j++) {
                if (s1[i].equals(s1[j])) {
                    count++;
                }
            }
        }
        System.out.println(count);
    }
}

```

```

    }

    int n2 = sc.nextInt();

    String[] s2 = new String[n2];

    for (int i = 0; i < n2; i++) {

        s2[i] = sc.next();

    }

    System.out.println(UserMainCode.countCommonStrings(s1,s2));

}

}

```

### **USERMAINCODE:**

```

import java.util.ArrayList;

public class UserMainCode {

    public static int countCommonStrings(String[] s1,String[] s2)

    {

        int count=0;

        ArrayList<String> al = new ArrayList<String>();

        for (int i = 0; i < s1.length; i++) {

            for (int j = 0; j < s2.length; j++) {

                if(s1[i].equals(s2[j])){

                    if(!al.contains(s1[i])){

                        count++;

                    }

                    al.add(s1[i]);

                }

            }

        }

        return count;

    }

}

```

## 35.Sequence Sum

Write a program to read a non-negative integer n, and find sum of fibonanci series for n number..

Include a class UserMainCode with a static method getFibonacciSum which accepts the integer value. The return type is integer.

The fibonacci sequence is a famous bit of mathematics, and it happens to have a recursive definition.

The first two values in the sequence are 0 and 1.

Each subsequent value is the sum of the previous two values, so the whole sequence is 0,1,1,2,3,5 and so on.

You will have to find the sum of the numbers of the Fibonaaci series for a given int n.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

## Input and Output Format:

Input consists of a integer.

Output consists of integer.

Refer sample output for formatting specifications.

## Sample Input 1:

5

## Sample Output 1:

**MAIN:**

```

import java.util.Scanner;

public class Main {

    public static void main(String[] args)
    {
        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        System.out.println(UserMainCode.getFibonacciSum(n));
    }
}

```

**USERMAINCODE:**

```

import java.util.ArrayList;
import java.util.Scanner;

public class UserMainCode {

    public static int getFibonacciSum(int n){
        int a=0,b=1,c=0,d=1;
        for(int i=3;i<=n;i++){
            c=a+b;
            a=b;
            b=c;
        }
        return c;
    }
}

```

```

d=d+c;

}

return d;

}

}

```

### 36.E-Mail Validation

Write a program to read a string and validate the given email-id as input. Validation Rules: 1. Ensure that there are atleast 5 characters between '@' and '.' 2. There should be only one '.' and one '@' symbol. 3. The '.' should be after the '@' symbol. 4. There must be atleast three characters before '@'. 5. The string after '.' should only be 'com'

Include a class UserMainCode with a static method ValidateEmail which accepts the string. The return type is TRUE / FALSE as per problem. Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

**Input and Output Format:** Input consists of a string. Output consists of TRUE / FALSE. Refer sample output for formatting specifications.

Sample Input 1: test@gmail.com Sample Output 1: TRUE

Sample Input 2: academy@xyz.com Sample Output 2: FALSE

#### **MAIN:**

```

import java.util.*;

public class Main {

public static void main(String[] args) {

Scanner s=new Scanner(System.in);

String ip;

ip=s.nextLine();

boolean b=UserMainCode.ValidateEmail(ip);

if(b==true)

System.out.println("TRUE");

else

```

```
        System.out.println("FALSE");  
    }  
}
```

### USERMAINCODE:

```
import java.util.StringTokenizer;  
  
public class UserMainCode {  
  
    public static boolean ValidateEmail(String ip) {  
  
        int i=0;  
  
        boolean b=false;  
  
        StringTokenizer t=new StringTokenizer(ip,"@");  
  
        String s1=t.nextToken();  
  
        String s2=t.nextToken();  
  
        StringTokenizer t1=new StringTokenizer(s2,".");  
  
        String s3=t1.nextToken();  
  
        String s4=t1.nextToken();  
  
        if(ip.contains("@") && ip.contains("."))  
  
            i++;  
  
        if(i==1)  
  
            if(s3.length()==5)  
  
                if(s1.length()>=3)  
  
                    if(s4.equals("com"))  
  
                        b=true;  
  
        return b;  
    }  
}
```

## 37.Symmetric Difference

Write a program to read two integer array and calculate the symmetric difference of the two arrays. Finally Sort the array.

Symmetric difference is the difference of A Union B and A Intersection B ie. [ (A U B) - (A ^ B)]

Union operation merges the two arrays and makes sure that common elements appear only once. Intersection operation

includes common elements from both the arrays.

Ex - A={12,24,7,36,14} and B={11,26,7,14}.

A U B ={ 7,11,12,14,24,26,36} and

A ^ B = {7,14}

Symmetric difference of A and B after sorting= [A U B] - [ A ^ B] = {11,12,24,26,36}.

Include a class UserMainCode with a static method **getSymmetricDifference** which accepts the integer array. The return

type is an integer array.

Create a Class Main which would be used to accept the two integer arrays and call the static method present in

UserMainCode.

### **Input and Output Format:**

Input consists of an integer n which is the number of elements followed by n integer values. The same sequence is followed

for the next array.

Output consists of sorted symmetric difference array.

Refer sample output for formatting specifications.

### **Sample Input 1:**

11

5

14

26

3

3

5

3

1

### **Sample Output 1:**

1

11

14

26

### **MAIN:**

```
import java.util.*;

public class Main

{
    public static void main(String[] args)
    {
        int n,m;

        Scanner sin = new Scanner(System.in);
    }
}
```

```

n = sin.nextInt();

int[] a1 = new int[n];

for(int i=0;i<n;i++)

{

a1[i] = sin.nextInt();

}

m = sin.nextInt();

int[] a2 = new int[m];

for(int i=0;i<m;i++)

{

a2[i] = sin.nextInt();

}

int[] result = UserMainCode.getSymmetricDifference (a1,a2);

for(int i=0;i<result.length;i++)

System.out.println(result[i]);

}
}

```

## USERMAINCODE:

```

import java.util.*;

public class UserMainCode

{

public static int[] getSymmetricDifference (int[] a1,int[] a2)

{

```

```
//int[] a1 = new int[]{11,5,14,26,3};  
  
//int[] a2 = new int[]{5,3,1};  
  
int[] union,inter,result;  
  
int count=0;  
  
int max = a1.length+a2.length;  
  
ArrayList<Integer> temp = new ArrayList<Integer>(max);  
  
/*union*/  
  
for(int i=0;i<a1.length;i++)  
  
{  
  
if(!temp.contains(a1[i]))  
  
{  
  
++count;  
  
temp.add(a1[i]);  
  
}  
  
}  
  
for(int i=0;i<a2.length;i++)  
  
{  
  
if(!temp.contains(a2[i]))  
  
{  
  
++count;  
  
temp.add(a2[i]);  
  
}  
  
}  
  
union = new int[count];  
  
for(int i=0;i<count;i++)
```

```
{  
union[i] = (int)temp.get(i);  
}  
  
Arrays.sort(union);  
  
/*intersection*/  
  
temp = new ArrayList<Integer>(max);  
  
count =0;  
  
Arrays.sort(a2);  
  
for(int i=0;i<a1.length;i++)  
  
{  
  
if(Arrays.binarySearch(a2,a1[i]) >= 0)  
  
{  
  
++count;  
  
temp.add(a1[i]);  
  
}  
  
}  
  
inter = new int[count];  
  
for(int i=0;i<count;i++)  
  
{  
  
inter[i] = (int)temp.get(i);  
  
}  
  
Arrays.sort(inter);  
  
/*difference */  
  
temp = new ArrayList<Integer>(max);  
  
count =0;
```

```

Arrays.sort(inter);

for(int i=0;i<union.length;i++)

{

if(Arrays.binarySearch(inter,union[i]) < 0)

{

++count;

temp.add(union[i]);

}

}

result = new int[count];

for(int i=0;i<count;i++)

{

result[i] = (int)temp.get(i);

}

Arrays.sort(result);

//System.out.println("resultant array : \n "+Arrays.toString(result));

return result;

}
}

```

## 38.Day of Week

Write a program to read a string containing date in DD/MM/YYYY format and prints the day of the week that date falls on.

Return the day in lowercase letter (Ex: monday)

Include a class UserMainCode with a static method **getDayOfWeek** which accepts the string. The return type is the string.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of a string.

Refer sample output for formatting specifications.

### **Sample Input 1:**

02/04/1985

### **Sample Output 1:**

Tuesday

### **MAIN:**

```
-  
  
import java.text.ParseException;  
  
import java.text.SimpleDateFormat;  
  
import java.util.Date;  
  
import java.util.Scanner;  
  
public class Main {  
  
    public static void main(String[] args) throws ParseException {  
  
        Scanner sc=new Scanner(System.in);
```

```
String s1=sc.nextLine();

System.out.println(UserMainCode.getDayOfWeek(s1));

}

}
```

## USERMAINCODE:

```
import java.text.SimpleDateFormat;

import java.text.ParseException;

import java.util.Date;

public class UserMainCode {

    public static String getDayOfWeek(String s1) throws ParseException

    {

        SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");

        SimpleDateFormat sdf1=new SimpleDateFormat("EEEEEE");

        Date d=sdf.parse(s1);

        String s=sdf1.format(d);

        return s.toLowerCase();

    }

}
```

## **39.Add Time**

Write a program to read two String variables containing time intervals in hh:mm:ss format.  
Add the two time intervals and

return a string in days:hours:minutes:seconds format where DD is number of days.

Hint: Maximum value for hh:mm:ss is 23:59:59

Include a class UserMainCode with a static method **addTime** which accepts the string values.  
The return type is the string.

Create a Class Main which would be used to accept the two string values and call the static  
method present in

UserMainCode.

### **Input and Output Format:**

Input consists of two string.

Output consists of a string.

Refer sample output for formatting specifications.

### **Sample Input 1:**

12:45:30

13:50:45

### **Sample Output 1:**

1:2:36:15

### **Sample Input 2:**

23:59:59

23:59:59

### **Sample Output 2:**

1:23:59:58

## MAIN:

```
-  
  
import java.util.*;  
  
import java.io.IOException;  
  
import java.text.*;  
  
public class Main {  
  
    public static void main(String[] args) throws IOException, ParseException {  
  
        Scanner s = new Scanner(System.in);  
  
        String s1=s.next();  
  
        String s2=s.next();  
  
        System.out.println(UserMainCode.addTime(s1,s2));  
  
    }  
  
}
```

## USERMAINCODE:

```
-  
  
import java.util.*;  
  
import java.io.IOException;  
  
import java.text.*;
```

```

public class UserMainCode {

    public static String addTime(String s1, String s2) throws IOException,
ParseException{

    SimpleDateFormat sdf=new SimpleDateFormat("HH:mm:ss");

    sdf.setTimeZone(TimeZone.getTimeZone("UTC"));

    sdf.setTimeZone(TimeZone.getTimeZone("s1"));

    sdf.setTimeZone(TimeZone.getTimeZone("s2"));

    Date d1=sdf.parse(s1);

    Date d2=sdf.parse(s2);

    long add=d1.getTime()+d2.getTime();

    String s=sdf.format(add);

    Calendar cal=Calendar.getInstance();

    cal.setTime(sdf.parse(s));

    int FindDay=cal.get(Calendar.DAY_OF_MONTH);

    if(FindDay>1)

        FindDay=FindDay-1;

    String op=FindDay+":" +s;

    return op;

}

}

```

## 40.ISBN Validation

Write a program to read a string and validate the given ISBN as input.

Validation Rules:

1. An ISBN (International Standard Book Number) is a ten digit code which uniquely identifies a book.
2. To verify an ISBN you calculate 10 times the first digit, plus 9 times the second digit, plus 8 times the third ..all the way until you add 1 times the last digit.

If the final number leaves no remainder when divided by 11 the code is a valid ISBN.

Example 1:

Input:0201103311

Calculation:  $10*0 + 9*2 + 8*0 + 7*1 + 6*1 + 5*0 + 4*3 + 3*3 + 2*1 + 1*1 = 55$ .

$55 \bmod 11 = 0$

Hence the input is a valid ISBN number

Output: true

Include a class UserMainCode with a static method **validateISBN** which accepts the string.  
The return type is TRUE /

FALSE as per problem.

Create a Class Main which would be used to accept the string and call the static method present in UserMainCode.

### **Input and Output Format:**

Input consists of a string.

Output consists of TRUE / FALSE.

Refer sample output for formatting specifications.

### **Sample Input 1:**

0201103311

### **Sample Output 1:**

TRUE

### **MAIN:**

-

```
import java.util.*;  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        Scanner s = new Scanner(System.in);  
  
        String ip=s.next();  
  
        System.out.println(UserMainCode.ISBNnumber(ip));  
  
    }  
}
```

### **USERMAINCODE:**

-

```
import java.util.*;  
  
import java.text.*;  
  
public class UserMainCode {  
  
    public static String ISBNnumber(String ip) {
```

```

String b="FALSE";

int sum=0;

for(int i=0,j=ip.length();i<ip.length();i++,j--){
    String s=String.valueOf(ip.charAt(i));
    int n=Integer.parseInt(s);
    sum+=(n*j);
}

//System.out.println(sum);

if(sum%11==0)
    b="TRUE";
}

return b;
}
}

```

## 41.Date Format

Write a program to read two String variables in DD-MM-YYYY. Compare the two dates and return the older date in

'MM/DD/YYYY' format.

Include a class UserMainCode with a static method **findOldDate** which accepts the string values. The return type is the

string.

Create a Class Main which would be used to accept the two string values and call the static method present in

UserMainCode.

## **Input and Output Format:**

Input consists of two string.

Output consists of a string.

Refer sample output for formatting specifications.

### **Sample Input 1:**

05-12-1987

8-11-2010

### **Sample Output 1:**

12/05/1987

### **MAIN:**

```
-  
  
import java.text.ParseException;  
  
import java.text.SimpleDateFormat;  
  
import java.util.*;  
  
public class Main {  
  
  
    public static void main(String[] args) throws ParseException {  
        Scanner s = new Scanner(System.in);  
  
        String s1=s.next();  
  
        String s2=s.next();  
  
        System.out.println(UserMainCode.findOldDate(s1,s2));  
    }  
}
```

## USERMAINCODE:

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

public class UserMainCode {

    public static String findOldDate(String s1, String s2) throws ParseException
    {
        SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
        SimpleDateFormat sdf1=new SimpleDateFormat("MM-dd-yyyy");
        Date d1=sdf.parse(s1);
        Date d2=sdf.parse(s2);
        Calendar cal=Calendar.getInstance();
        cal.setTime(d1);
        long y=cal.getTimeInMillis();
        cal.setTime(d2);
        long y1=cal.getTimeInMillis();
        String s3=sdf1.format(d1);
        String s4=sdf1.format(d2);
        if(y<y1)
            return s3;
    }
}
```

```
    else  
  
        return s4;  
  
    }  
  
}  
  
-  
-
```

## 42. Interest Calculation

Write a program to calculate amount of the account holders based on the below mentioned prototype:

1. Read account details from the User. The details would include id, DOB (date of birth) and amount in the given order. The datatype for id is string, DOB is string and amount is integer.
2. You decide to build two hashmaps. The first hashmap contains employee id as key and DOB as value, and the second hashmap contains same employee ids as key and amount as value.
3. Rate of interest as on 01/01/2015:
  - a. If the age greater than or equal to 60 then interest rate is 10% of Amount.
  - b. If the age less than or equal to 60 and greater than or equal to 30 then interest rate is 7% of Amount.
  - v. If the age less than 30 interest rate is 4% of Amount.
4. Revised Amount = principle Amount + interest rate.
5. You decide to write a function **calculateInterestRate** which takes the above hashmaps as input and returns the treemap as output. Include this function in class UserMainCode.

Create a Class Main which would be used to read employee details in step 1 and build the two hashmaps. Call the static

method present in UserMainCode.

**Input and Output Format:**

Input consists of account details. The first number indicates the size of the account. The next three values indicate the user id, DOB and amount. The Employee DOB format is “dd-mm-yyyy”

Output consists of the user id and the amount for each user one in a line.

Refer sample output for formatting specifications.

**Sample Input 1:**

4

SBI-1010

20-01-1987

10000

SBI-1011

03-08-1980

15000

SBI-1012

05-11-1975

20000

SBI-1013

02-12-1950

30000

**Sample Output 1:**

SBI-1010:10400

SBI-1011:16050

SBI-1012:21400

SBI-1013:33000

## MAIN

```
-  
  
import java.util.HashMap;  
  
import java.util.Iterator;  
  
import java.util.Scanner;  
  
import java.util.TreeMap;  
  
public class Main {  
  
    public static void main(String []args){  
  
        Scanner sc=new Scanner(System.in);  
  
        int s=Integer.parseInt(sc.nextLine());  
  
        HashMap<String, String> hm=new HashMap<String, String>();  
  
        HashMap<String, Integer> hm1=new HashMap<String, Integer>();  
  
        for(int i=0;i<s;i++)  
  
        {  
  
            String id=sc.nextLine();  
  
            hm.put(id, sc.nextLine());  
  
            hm1.put(id, Integer.parseInt(sc.nextLine()));  
  
        }  
  
        TreeMap<String, Integer> tm=new TreeMap<String, Integer>();  
  
        tm=UserMainCode.calculateInterestRate(hm,hm1);  
  
        Iterator<String> it=tm.keySet().iterator();  
  
        while(it.hasNext())  
  
        {  
  
            String n=it.next();  
  
            int fac=tm.get(n);  
  
        }  
}
```

```
System.out.println(n+":"+fac);

}

}

}
```

## USERMAINCODE

```
import java.text.DecimalFormat;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.HashMap;

import java.util.Iterator;

import java.util.HashMap;

import java.util.TreeMap;

public class UserMainCode

{

    public static TreeMap<String, Integer> calculateInterestRate

(HashMap<String, String>hm,HashMap<String, Integer>hm1)

    {

        int year=0,amount=0;

        double dis=0;

        String now="01/01/2015";

        TreeMap<String, Integer>tm=new TreeMap<String, Integer>();

        Iterator<String> it=hm.keySet().iterator();
```

```
while(it.hasNext())
{
String id=it.next();
String dor=hm.get(id);
amount=hm1.get(id);

SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");
SimpleDateFormat sdf1=new SimpleDateFormat("dd/MM/yyyy");

try
{
Date d=sdf.parse(dor);
Date d1=sdf1.parse(now);
sdf.setLenient(false);

int y=d.getYear();
int y1=d1.getYear();

int m=d.getMonth();
int m1=d1.getMonth();

int day=d.getDay();
int day1=d1.getDay();

year=y1-y;

if(m>m1)
year--;
else if(m==m1)
{if(day<day1)
year--;
}
}
```

```
if(year>=60)

dis=0.1*amount+amount;

else if(year<60 && year>=30 )

dis=0.07*amount+amount;

else

dis=0.04*amount+amount;

tm.put(id,(int)dis);

}

catch(Exception e)

{

e.printStackTrace();

}

}

return tm;
```

## 43. Discount Rate Calculation

Write a program to calculate discount of the account holders based on the transaction amount and registration date using below mentioned prototype:

1. Read account details from the User. The details would include id, DOR (date of registration) and transaction amount in the

given order. The datatype for id is string, DOR is string and transaction amount is integer.

2. You decide to build two hashmaps. The first hashmap contains employee id as key and DOR as value, and the second hashmap contains same employee ids as key and amount as value.

3. Discount Amount as on 01/01/2015:

a. If the transaction amount greater than or equal to 20000 and registration greater than or equal to 5 year then discount rate is 20% of transaction amount.

b. If the transaction amount greater than or equal to 20000 and registration less than to 5 year then discount rate is 10% of transaction amount.

c. If the transaction amount less than to 20000 and registration greater than or equal to 5 year then discount rate is 15% of transaction amount.

d. If the transaction amount less than to 20000 and registration less than to 5 year then discount rate is 5% of transaction amount.

4. You decide to write a function **calculateDiscount** which takes the above hashmaps as input and returns the treemap as output. Include this function in class UserMainCode.

Create a Class Main which would be used to read employee details in step 1 and build the two hashmaps. Call the static method present in UserMainCode.

#### **Input and Output Format:**

Input consists of transaction details. The first number indicates the size of the employees. The next three values indicate the user id, user DOR and transaction amount. The DOR (Date of Registration) format is “dd-mm-yyyy”

Output consists of a string which has the user id and discount amount one in a line for each user.

Refer sample output for formatting specifications.

#### **Sample Input 1:**

4

A-1010

20-11-2007

25000

B-1011

04-12-2010

30000

C-1012

11-11-2005

15000

D-1013

02-12-2012

10000

**Sample Output 1:**

A-1010:5000

B-1011:3000

C-1012:2250

D-1013:500

**MAIN:**

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.TreeMap;
import java.util.Scanner;

public class Main{
    public static void main(String []args){
        Scanner sc=new Scanner(System.in);
        int s=Integer.parseInt(sc.nextLine());
        HashMap<String, String> hm=new HashMap<String, String>();
        HashMap<String, Integer> hm1=new HashMap<String, Integer>();
        for(int i=0;i<s;i++)
            hm.put(sc.nextLine(), hm1.get(sc.nextLine()));
        hm1.put(sc.nextLine(), s);
    }
}
```

```

{
String id=sc.nextLine();

hm.put(id, sc.nextLine());

hm1.put(id,Integer.parseInt(sc.nextLine()));

}

TreeMap<String, Integer>tm=new TreeMap<String, Integer>();

tm=UserMainCode.calculateDiscount(hm,hm1);

Iterator<String> it=tm.keySet().iterator();

while(it.hasNext())

{

String n=it.next();

int fac=tm.get(n);

System.out.println(n+":"+fac);

}

}

}

```

### USERMAINCODE

```

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.HashMap;

import java.util.*;

public class UserMainCode

{

public static TreeMap<String, Integer> calculateDiscount

(HashMap<String, String>hm,HashMap<String, Integer>hm1)

```

```
{  
  
int amount=0;  
  
double dis=0;  
  
String now="01/01/2015";  
  
TreeMap<String, Integer>tm=new TreeMap<String, Integer>();  
  
Iterator<String> it=hm.keySet().iterator();  
  
while(it.hasNext())  
  
{  
  
String id=it.next();  
  
String dor=hm.get(id);  
  
amount=hm1.get(id);  
  
SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");  
  
sdf.setLenient(false);  
  
try{  
  
Date d=new Date();  
  
Date d1=new Date();  
  
String a=dor;  
  
String b="01-01-2014";  
  
d=sdf.parse(a);  
  
d1=sdf.parse(b);  
  
long t=d.getTime();  
  
long t1=d1.getTime();  
  
long t3=t1-t;  
  
long l1=(24 * 60 * 60 * 1000);  
  
long l=l1*365;
```

```
long year1=t3/l;

//System.out.println("Result="+year1);

if(year1>=5 && amount>=20000)
dis=0.2*amount;

else if(year1<5 && amount>=20000)
dis=0.1*amount;

else if(year1>=5 && amount<20000)
dis=0.15*amount;

else
dis=0.05*amount;

tm.put(id,(int)dis);

}

catch(Exception e)
{
e.printStackTrace();
}

}

return tm;
}
```



## EmployeeDetailsReport/database.properties

```
1 #IF NEEDED, YOU CAN MODIFY THIS PROPERTY FILE
2 #ENSURE YOU ARE NOT CHANGING THE NAME OF THE PROPERTY
3 #YOU CAN CHANGE THE VALUE OF THE PROPERTY
4 #LOAD THE DETAILS OF DRIVER CLASS, URL, USERNAME AND PASSWORD using this properties file
only.
5 #Do not hard code the values
6
7 DB_DRIVER_CLASS=oracle.jdbc.driver.OracleDriver
8 DB_URL=jdbc:oracle:thin:@127.0.0.1:1521:XE
9 DB_USERNAME=${sys:db_username}
10 DB_PASSWORD=${sys:db_password}
```

## EmployeeDetailsReport/inputfeed.txt

```
1 PR47856,Steven,level1,24
2 TR87965,Sujatha,level1,12
3 PR78965,Seetha,level3,12
4 PR7895,Vaishnav,level2,6
5 PR78995,Yamuna,level2,25
6 PR78999,John,level4,13
7 TR88888,Wilsy,level4,16
8 PR12356,Nelson,level1,27
9 PR85203,Mithun,level3,11
```

## EmployeeDetailsReport/src/com/cts/employeedetailsreport/client/EmployeeDetailsMain.java

```
1 package com.cts.employeedetailsreport.client;
2 import com.cts.employeedetailsreport.skeleton.SkeletonValidator;
3 import com.cts.employeedetailsreport.service.HospitalManagement;
4
5 public class EmployeeDetailsMain {
6
7     public static void main(String[] args) {
8         // CODE SKELETON - VALIDATION STARTS
9         // DO NOT CHANGE THIS CODE
10
11         new SkeletonValidator();
12
13         // CODE SKELETON - VALIDATION ENDS
14
15 // TYPE YOUR CODE HERE
16     new HospitalManagement();
17 }
18
19 }
20
21
```

## EmployeeDetailsReport/src/com/cts/employeedetailsreport/dao/DBConnectionManager.java

```
1 package com.cts.employeedetailsreport.dao;
2
3
4 import java.io.FileInputStream;
5 import java.io.IOException;
6 import java.sql.Connection;
7 import java.sql.DriverManager;
8 import java.util.Properties;
9
10 import com.cts.employeedetailsreport.exception.InvalidEmployeeNumberException;
11
12
```

```

13 public class DBConnectionManager {
14
15
16
17     private static Connection con = null;
18     private static DBConnectionManager instance;
19     public DBConnectionManager() throws InvalidEmployeeNumberException
20     {
21         FileInputStream fis=null;
22         try{
23             fis = new FileInputStream("database.Properties");
24             Properties props = new Properties();
25             props.load(fis);
26
27             Class.forName(props.getProperty("DB_DRIVER_CLASS"));
28
29             con =
DriverManager.getConnection(props.getProperty("DB_URL"),props.getProperty("DB_USERNAME"),props.getProperty("DB_PASSWORD"));
30         }
31         catch(Exception e){
32             e.printStackTrace();
33         }finally{
34             try{
35                 fis.close();
36             }
37             catch(IOException e)
38             {
39                 e.printStackTrace();
40             }
41         }
42
43         //FILL THE CODE HERE
44     }
45     public static DBConnectionManager getInstance() throws InvalidEmployeeNumberException {
46         //FILL THE CODE HERE
47         instance = new DBConnectionManager();
48
49         return instance;
50     }
51     public Connection getConnection() {
52         return con;
53     }
54 }
55

```

## EmployeeDetailsReport/src/com/cts/employeedetailsreport/dao/DetailsDAO.java

```

1 package com.cts.employeedetailsreport.dao;
2
3 import java.sql.Statement;
4 import java.sql.Connection;
5 import java.sql.SQLException;
6 import java.util.List;
7
8 import com.cts.employeedetailsreport.exception.InvalidEmployeeNumberException;
9 import com.cts.employeedetailsreport.model.EmployeeDetails;
10
11 public class DetailsDAO {
12
13
14     public boolean insertEmployeeList(List <EmployeeDetails> eList) throws
InvalidEmployeeNumberException {
15
16         boolean recordsAdded = false;
17         DBConnectionManager db = new DBConnectionManager();

```

```

18         DBCConnectionManager.getInstance();
19         Connection conne = db.getConnection();
20
21         // FILL THE CODE HERE
22         try{
23             Statement st = conne.createStatement();
24             for(int i=0; i<eList.size(); i++)
25             {
26                 String ins = "INSERT INTO EmployeeDetails
VALUES("+eList.get(i).getEmployeeName()+","+eList.get(i).getEmployeeNumber()+"."++eList.get(i).getLevel()+"."
+eList.get(i).getExtraWorkingHours()+"."++eList.get(i).getTotalSalary());
27                 st.executeUpdate(ins);
28             }
29             conne.commit();
30             recordsAdded = true;
31         }
32         catch(SQLException e)
33         {
34             e.printStackTrace();
35             try{
36                 conne.rollback();
37             }
38             catch(Exception k)
39             {
40                 k.printStackTrace();
41             }
42         }finally{
43             try{
44                 conne.close();
45             }catch(Exception e){
46                 e.printStackTrace();
47             }
48         }
49         return recordsAdded;
50     }
51 }
52

```

## EmployeeDetailsReport/src/com/cts/employeedetailsreport/exception/InvalidEmployeeNumberException.java

```

1 package com.cts.employeedetailsreport.exception;
2
3 public class InvalidEmployeeNumberException extends Exception
4 {
5
6     String strMsg1;
7     Throwable strMsg2;
8
9
10    public InvalidEmployeeNumberException() {
11        super();
12    }
13    public InvalidEmployeeNumberException(String strMsg1)
14    {
15        super(strMsg1);
16    }
17
18    public InvalidEmployeeNumberException(String strMsg1, Throwable strMsg2) {
19        super();
20        this.strMsg1 = strMsg1;
21        this.strMsg2 = strMsg2;
22    }
23 }

```

## EmployeeDetailsReport/src/com/cts/employeedetailsreport/model/EmployeeDetails.java

```
1 package com.cts.employeedetailsreport.model;
2
3 public class EmployeeDetails {
4     private String employeeNumber;
5     private String employeeName;
6     private String level;
7     private int extraWorkingHours;
8     private double totalSalary;
9
10
11
12
13 //Constructors
14
15     public EmployeeDetails(String string1, String string2, String string3, int i, double sal) {
16         this.employeeNumber=string1;
17         this.employeeName=string2;
18         this.level=string3;
19         this.extraWorkingHours=i;
20         this.totalSalary=sal;
21     }
22
23
24
25
26
27
28     public EmployeeDetails() {
29 }
30
31
32
33
34 //getters and setters
35
36     public String getEmployeeNumber() {
37         return employeeNumber;
38     }
39
40
41
42
43
44
45     public void setEmployeeNumber(String employeeNumber) {
46         this.employeeNumber = employeeNumber;
47     }
48
49
50
51
52
53
54     public String getEmployeeName() {
55         return employeeName;
56     }
57
58
59
60
61
62
63     public void setEmployeeName(String employeeName) {
64         this.employeeName = employeeName;
65     }
66
67
```

```
68
69
70
71
72     public String getLevel() {
73         return level;
74     }
75
76
77
78
79
80
81     public void setLevel(String level) {
82         this.level = level;
83     }
84
85
86
87
88
89
90     public int getExtraWorkingHours() {
91         return extraWorkingHours;
92     }
93
94
95
96
97
98
99     public void setExtraWorkingHours(int extraWorkingHours) {
100        this.extraWorkingHours = extraWorkingHours;
101    }
102
103
104
105
106
107
108     public double getTotalSalary() {
109         return totalSalary;
110     }
111
112
113
114
115
116
117     public void setTotalSalary(double totalSalary) {
118         this.totalSalary = totalSalary;
119     }
120
121
122
123
124
125
126     @Override
127     public String toString() {
128         return "EmployeeDetails [employeeNumber=" + employeeNumber + ", employeeName="
+ employeeName + ", level="
129                                         + level + ", extraWorkingHours=" + extraWorkingHours + ",
130         totalSalary=" + totalSalary + "]";
131     }
132
```

```
133  
134  
135 }  
136
```

## EmployeeDetailsReport/src/com/cts/employeedetailsreport/service/HospitalManagement.java

```
1 package com.cts.employeedetailsreport.service;  
2  
3 import java.util.ArrayList;  
4 import java.util.List;  
5 import com.cts.employeedetailsreport.exception.InvalidEmployeeNumberException;  
6 import com.cts.employeedetailsreport.model.EmployeeDetails;  
7 import com.cts.employeedetailsreport.util.ApplicationUtil;  
8  
9 public class HospitalManagement {  
10     private List<String>employeeRecords;  
11     public List<String> getEmployeeRecords(){  
12         return employeeRecords;  
13     }  
14     public void setEmployeeRecords(List<String>employeeRecords){  
15         this.employeeRecords = employeeRecords;  
16     }  
17  
18     public static ArrayList <EmployeeDetails> buildEmployeeList(List <String> employeeRecords) {  
19  
20  
21         final String COMMADELIMITER = ",";  
22         ArrayList <EmployeeDetails> empList = new ArrayList<EmployeeDetails>();  
23  
24         //fill the code here  
25         int listSize = employeeRecords.size();  
26         int i=0;  
27         EmployeeDetails empdet;  
28         while(listSize-->0){  
29             String[] employeeDetailsString = employeeRecords.get(i++).split(COMMADELIMITER);  
30             try{  
31                 if(ApplicationUtil.validate(employeeDetailsString[0])){  
32                     int extraHours = Integer.parseInt(employeeDetailsString[3]);  
33                     double sal = calculateTotalSalary(employeeDetailsString[2],extraHours);  
34                     empdet = new  
EmployeeDetails(employeeDetailsString[0],employeeDetailsString[1],employeeDetailsString[2],extraHours,sal);  
35                     empList.add(empdet);  
36                 }  
37             }  
38             catch(InvalidEmployeeNumberException in)  
39             {  
40                 System.out.println(in);  
41             }  
42         }  
43  
44         return empList;  
45     }  
46  
47  
48  
49  
50  
51     public boolean addEmployeeList(String inputFeed) throws InvalidEmployeeNumberException  
52     {  
53         //fill the code here  
54         try{  
55             this.setEmployeeRecords(ApplicationUtil.readFile(inputFeed));  
56             return true;  
57         }  
58         catch(Exception e)
```

```

59     {
60         e.printStackTrace();
61     }
62
63         return false;
64     }
65
66
67     public static double calculateTotalSalary(String level,int extraWorkingHours)
68     {
69         double sal=0.0;
70         //fill the code here
71         if(level.equals("level1"))
72         {
73             sal = 75000+(1000 * extraWorkingHours);
74         }
75         else if (level.equals("level2"))
76         {
77             sal = 50000+(1000 * extraWorkingHours);
78         }
79         else if(level.equals("level3"))
80         {
81             sal = 35000+(1000 * extraWorkingHours);
82         }
83         else if (level.equals("level4"))
84         {
85             sal = 25000+(1000 * extraWorkingHours);
86         }
87
88         return sal;
89     }
90
91
92 }
93

```

## EmployeeDetailsReport/src/com/cts/employeedetailsreport/skeleton/SkeletonValidator.java

```

1 package com.cts.employeedetailsreport.skeleton;
2
3
4 import java.lang.reflect.Method;
5 import java.util.logging.Level;
6 import java.util.logging.Logger;
7
8
9
10 /**
11  * @author TJ
12 *
13 * This class is used to verify if the Code Skeleton is intact and not modified by participants thereby ensuring
smooth auto evaluation
14 *
15 */
16 public class SkeletonValidator {
17     private static final Logger LOG = Logger.getLogger("SkeletonValidator");
18     public SkeletonValidator() {
19
20         validateClassName("com.cts.employeedetailsreport.dao.DetailsDAO");
21         validateClassName("com.cts.employeedetailsreport.dao.DBConnectionManager");
22         validateClassName("com.cts.employeedetailsreport.model.EmployeeDetails");
23         validateClassName("com.cts.employeedetailsreport.service.HospitalManagement");
24
25         validateClassName("com.cts.employeedetailsreport.exception.InvalidEmployeeNumberException");
26         validateClassName("com.cts.employeedetailsreport.util.ApplicationUtil");
27         //----
28     }
29
30 }
31

```

```

27     validateMethodSignature("buildEmployeeList:ArrayList,addEmployeeList:boolean", "com.cts.employee
detailsreport.service.HospitalManagement");
28     validateMethodSignature("insertEmployeeList:boolean", "com.cts.employeedetailsreport.dao.DetailsD
AO");
29     validateMethodSignature("getInstance:DBConnectionManager getConnection:Connection", "com.cts.e
mployeedetailsreport.dao.DBConnectionManager");
30
31
32
33     }
34
35
36
37     protected final boolean validateClassName(String className) {
38
39         boolean iscorrect = false;
40         try {
41             Class.forName(className);
42             iscorrect = true;
43             LOG.info("Class Name " + className + " is correct");
44
45         } catch (ClassNotFoundException e) {
46             LOG.log(Level.SEVERE, "You have changed either the " + "class
name/package. Use the correct package "
47                                         + "and class name as provided in the skeleton");
48
49         } catch (Exception e) {
50             LOG.log(Level.SEVERE,
51                     "There is an error in validating the " + "Class Name.
Please manually verify that the "
52                                         + "Class name is same as
53 skeleton before uploading");
54         }
55         return iscorrect;
56     }
57
58     protected final void validateMethodSignature(String methodWithExcptn, String className) {
59         Class cls = null;
60         try {
61
62             String[] actualmethods = methodWithExcptn.split(",");
63             boolean errorFlag = false;
64             String[] methodSignature;
65             String methodName = null;
66             String returnType = null;
67
68             for (String singleMethod : actualmethods) {
69                 boolean foundMethod = false;
70                 methodSignature = singleMethod.split(":");
71
72                 methodName = methodSignature[0];
73                 returnType = methodSignature[1];
74                 cls = Class.forName(className);
75                 Method[] methods = cls.getMethods();
76                 for (Method findMethod : methods) {
77                     if (methodName.equals(findMethod.getName())) {
78                         foundMethod = true;
79                         if
80                         !(findMethod.getReturnType().getSimpleName().equals(returnType))) {
81                             errorFlag = true;
82                             LOG.log(Level.SEVERE, " You
have changed the " + "return type in " + methodName

```

```
82
method. Please stick to the " + "skeleton provided");
83
84
85
+ methodName + " is valid");
86
87
88
89
90
91
92
public method " + methodName
93
94
public method name. " + "Verify it with the skeleton");
95
96
97
98
99
100
101
102
103
104
105
106
107
108 }
109
110
```

EmployeeDetailsReport/src/com/cts/employeedetailsreport/util/ApplicationUtil.java

```
1 package com.cts.employeedetailsreport.util;  
2  
3 import java.util.ArrayList;  
4 import java.io.FileInputStream;  
5 import java.io.FileNotFoundException;  
6 import java.io.InputStreamReader;  
7 import java.nio.charset.StandardCharsets;  
8 import java.util.List;  
9 import java.io.BufferedReader;  
10  
11 import com.cts.employeedetailsreport.exception.InvalidEmployeeNumberException;  
12  
13 public class ApplicationUtil {  
14     public static List<String> readFile(String filePath) throws FileNotFoundException,  
InvalidEmployeeNumberException  
15     {  
16         List<String> employeeList=new ArrayList<String>();  
17  
18         // FILL THE CODE HERE  
19         try(BufferedReader br = new BufferedReader(new InputStreamReader(new  
FileInputStream(filePath),StandardCharsets.UTF_8));)  
20         {  
21             String line;  
22             while((line = br.readLine()) !=null)  
23             {  
24                 employeeList.add(line);  
25             }  
26         }  
27     }  
28 }
```

```

27             catch(Exception e){
28                 }
29             return employeeList;
30         }
31     }
32     public static boolean validate(String employeeNumber) throws InvalidEmployeeNumberException
33     {
34         boolean val=false;
35         // FILL THE CODE HERE
36         int n = employeeNumber.length();
37         if(n != 7) throw new InvalidEmployeeNumberException("Invalid Employee
38 Number");
39         char[] charArray = employeeNumber.toCharArray();
40         if(charArray[0] != 'P' && charArray[1] != 'R') throw new
41 InvalidEmployeeNumberException("Invalid Employee Number");
42         for(int i=2; i<n; i++)
43         {
44             if(!Character.isDigit(charArray[i]))
45             {
46                 throw new InvalidEmployeeNumberException("Invalid Employee
47 Number");
48             }
49         val=true;
50     }
51     return val;
52
53
54
55     }
56
57
58 }
59

```

## **EmployeeDetailsMain.java**

```
package com.cts.employeedetailsreport.client;  
  
import com.cts.employeedetailsreport.skeleton.SkeletonValidator;  
  
import com.cts.employeedetailsreport.service.HospitalManagement;  
  
public class EmployeeDetailsMain {  
  
    public static void main(String[] args) {  
  
        // CODE SKELETON - VALIDATION STARTS  
  
        // DO NOT CHANGE THIS CODE  
  
        new SkeletonValidator();  
  
        // CODE SKELETON - VALIDATION ENDS  
  
        // TYPE YOUR CODE HERE  
  
        new HospitalManagement();  
  
    }  
  
}
```

---

## **DBConnectionManager.java**

```
package com.cts.employeedetailsreport.dao;  
  
import java.io.FileInputStream;  
import java.io.IOException;  
import java.sql.DriverManager;  
import java.sql.Connection;  
import java.util.Properties;  
  
  
  
import com.cts.employeedetailsreport.exception.InvalidEmployeeNumberException;
```

```
public class DBConnectionManager {  
  
    private static Connection con = null;  
    private static DBConnectionManager instance;  
    public DBConnectionManager() throws InvalidEmployeeNumberException  
    {  
        FileInputStream fis=null;  
        try{  
            fis=new FileInputStream("database.properties");  
            Properties props=new Properties();  
            props.load(fis);  
            Class.forName(props.getProperty("DB_DRIVER_CLASS"));  
  
            con=DriverManager.getConnection(props.getProperty("DB_URL"),props.getProperty("DB_U  
SERNAME"),props.getProperty("DB_PASSWORD"));  
        }catch(Exception e){  
            e.printStackTrace();  
        }finally{  
            try{  
                fis.close();  
  
            }catch(IOException e){  
                e.printStackTrace();  
            }  
        }  
    }  
}
```

```

        //FILL THE CODE HERE
    }

    public static DBConnectionManager getInstance() throws InvalidEmployeeNumberException
{
    //FILL THE CODE HERE
    instance=new DBConnectionManager();
    return instance;
}

public Connection getConnection(){
    return con;
}

}
=====
```

## **DetailsDAO.java**

```

package com.cts.employeedetailsreport.dao;

import java.sql.Statement;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.List;

import com.cts.employeedetailsreport.exception.InvalidEmployeeNumberException;
import com.cts.employeedetailsreport.model.EmployeeDetails;

public class DetailsDAO {
```

```
    public boolean insertEmployeeList(List <EmployeeDetails> eList) throws
InvalidEmployeeNumberException {

        boolean recordsAdded = false;
        DBConnectionManager db=new DBConnectionManager();
        DBConnectionManager.getInstance();
        Connection conne=db.getConnection();

        // FILL THE CODE HERE

    try{
        Statement st=conne.createStatement();
        for(int i=0;i<eList.size();i++)
        {
            String ins="INSERT INTO EmployeeDetails
VALUES("+eList.get(i).getEmployeeName()+","+eList.get(i).getEmployeeNumber()+","+eList.get(i).get
Level()+","+eList.get(i).getExtraWorkingHours()+","+eList.get(i).getTotalSalary());
            st.executeUpdate(ins);
        }
        conne.commit();
        recordsAdded=true;
    }
    catch(SQLException e){
        e.printStackTrace();
    }

    try{
        conne.rollback();
    }catch(Exception k){
        k.printStackTrace();
    }
}finally{
    try{
```

```
conne.close();

}catch(Exception e){

    e.printStackTrace();

}

}

return recordsAdded;

}

=====
```

## **InvalidEmployeeNumberException.java**

```
package com.cts.employeedetailsreport.exception;

public class InvalidEmployeeNumberException extends Exception

{



String strMsg1;
Throwable strMsg2;





public InvalidEmployeeNumberException() {
    super();
}

public InvalidEmployeeNumberException(String strMsg1)
{
    super(strMsg1);
}

public InvalidEmployeeNumberException(String strMsg1, Throwable strMsg2) {
    super();
    this.strMsg1 = strMsg1;
```

```
        this.strMsg2 = strMsg2;  
    }  
}
```

=====

## **EmployeeDetails.java**

```
package com.cts.employeedetailsreport.model;  
  
public class EmployeeDetails {  
    private String employeeNumber;  
    private String employeeName;  
    private String level;  
    private int extraWorkingHours;  
    private double totalSalary;  
  
    //Constructors  
  
    public EmployeeDetails(String string1, String string2, String string3, int i,double sal) {  
        this.employeeNumber=string1;  
        this.employeeName=string2;  
        this.level=string3;  
        this.extraWorkingHours=i;  
        this.totalSalary=sal;  
    }  
  
    public EmployeeDetails() {  
    }  
  
    //getters and setters
```

```
public String getEmployeeNumber() {
    return employeeNumber;
}

public void setEmployeeNumber(String employeeNumber) {
    this.employeeNumber = employeeNumber;
}

public String getEmployeeName() {
    return employeeName;
}

public void setEmployeeName(String employeeName) {
    this.employeeName = employeeName;
}

public String getLevel() {
    return level;
}

public void setLevel(String level) {
    this.level = level;
}

public int getExtraWorkingHours() {
    return extraWorkingHours;
}

public void setExtraWorkingHours(int extraWorkingHours) {
    this.extraWorkingHours = extraWorkingHours;
}
```

```

public double getTotalSalary() {
    return totalSalary;
}

public void setTotalSalary(double totalSalary) {
    this.totalSalary = totalSalary;
}

@Override
public String toString() {
    return "EmployeeDetails [employeeNumber=" + employeeNumber + ", employeeName=" + employeeName + ", level="
           + level + ", extraWorkingHours=" + extraWorkingHours + ", totalSalary=" + totalSalary + "]";
}

}

```

---

## HospitalManagement.java

```

package com.cts.employeedetailsreport.service;

import java.util.ArrayList;
import java.util.List;
import com.cts.employeedetailsreport.exception.InvalidEmployeeNumberException;
import com.cts.employeedetailsreport.model.EmployeeDetails;
import com.cts.employeedetailsreport.util.ApplicationUtil;

```

```

public class HospitalManagement {
    private List<String>employeeRecords;
    public List<String> getEmployeeRecords(){
        return employeeRecords;
    }
    public void setEmployeeRecords(List<String>employeeRecords){
        this.employeeRecords=employeeRecords;
    }
}

public static ArrayList <EmployeeDetails> buildEmployeeList(List <String> employeeRecords) {
    final String COMMADELIMITER = ",";
    ArrayList <EmployeeDetails> empList = new ArrayList<EmployeeDetails>();

    //fill the code here
    int listSize=employeeRecords.size();
    int i=0;
    EmployeeDetails empdet;
    while(listSize-->0){
        String[]
        employeeDetailsString=employeeRecords.get(i++).split(COMMADELIMITER);
        try{
            if(ApplicationUtil.validate(employeeDetailsString[0])){
                int extraHours=Integer.parseInt(employeeDetailsString[3]);
                double sal=calculateTotalSalary(employeeDetailsString[2],extraHours);
                empdet =new
                EmployeeDetails(employeeDetailsString[0],employeeDetailsString[1],employeeDetailsString[2],extra
                Hours,sal);
                empList.add(empdet);
            }
        }
    }
}

```

```
        }

        catch(InvalidEmployeeNumberException in){

            System.out.print(in);

        }

    }

    return empList;

}
```

```
public boolean addEmployeeList(String inputFeed) throws  
InvalidEmployeeNumberException
```

```
{  
    //fill the code here  
    try{  
        this.setEmployeeRecords(ApplicationUtil.readFile(inputFeed));  
        return true;  
  
    }  
    catch(Exception e){  
        e.printStackTrace();  
    }  
    return false;  
}
```

```
public static double calculateTotalSalary(String level,int extraWorkingHours)
```

```
{  
    double sal=0.0;  
    //fill the code here  
    if(level.equals("level1")){  
        sal=75000+(1000*extraWorkingHours);  
    }  
}
```

```
        else if(level.equals("level2")){
            sal=50000+(1000*extraWorkingHours);
        }
        else if(level.equals("level3")){
            sal=35000+(1000*extraWorkingHours);
        }
        else if(level.equals("level4")){
            sal=25000+(1000*extraWorkingHours);
        }
        return sal;
    }

}
```

```
=====
=====
```

## **ApplicationUtil.java**

```
package com.cts.employeedetailsreport.util;

import java.util.ArrayList;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.InputStreamReader;
import java.nio.charset.StandardCharsets;
import java.util.List;
import java.io.BufferedReader;
import com.cts.employeedetailsreport.exception.InvalidEmployeeNumberException;

public class ApplicationUtil {
```

```

public static List<String> readFile(String filePath) throws
FileNotFoundException,InvalidEmployeeNumberException

{
    List<String> employeeList=new ArrayList<String>();

    // FILL THE CODE HERE

    try(BufferedReader br=new BufferedReader(new InputStreamReader(new
FileInputStream(filePath),StandardCharsets.UTF_8))){
        String line;
        while((line=br.readLine())!=null){
            employeeList.add(line);
        }
    }

    catch(Exception e){

    }

    return employeeList;

}

public static boolean validate(String employeeNumber) throws
InvalidEmployeeNumberException

{
    boolean val=false;
    // FILL THE CODE HERE

    int n=employeeNumber.length();
    if(n!=7)throw new InvalidEmployeeNumberException("Invalid Employee Number");

    char[] charArray=employeeNumber.toCharArray();

    if(charArray[0]!='P'&&charArray[1]!='R')throw new
InvalidEmployeeNumberException("Invalid Employee Number");
}

```

```
for(int i=2;i<n;i++)  
{  
    if(!Character.isDigit(charArray[i]))  
    {  
        throw new InvalidEmployeeNumberException("Invalid Employee Number");  
    }  
}  
  
val=true;  
return val;  
}  
  
}
```

## BILLCONTROLLER.JAVA

```
package com.controller;

import java.util.HashMap;
import java.util.Map;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.ModelMap;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PostMapping;

import com.model.Order;
import com.service.BillService;

//use appropriate annotation to configure BillController as Controller
@Controller
public class BillController {

    @Autowired
    BillService service;

    @ModelAttribute("productList")
    public Map<String, String> populateProductType() {
        Map<String, String> serviceMap = new HashMap<String, String>();
        serviceMap.put("HeadPhone", "HeadPhone");
    }
}
```

```

        serviceMap.put("TravelAdapter", "TravelAdapter");
        serviceMap.put("MemoryCard", "MemoryCard");
        serviceMap.put("PenDrive", "PenDrive");
        serviceMap.put("USBCable", "USBCable");

        return serviceMap;
    }

    @GetMapping("/showPage")
    public String showPage(@ModelAttribute("order") Order order) {
        return "showpage";
    }

    // invoke the service class - calculateTotalCost method.

    @PostMapping("/billDesk")
    public String calculateTotalCost(@Valid @ModelAttribute("order") Order order,
        BindingResult result,
        ModelMap model) {

        // fill the code
        if (result.hasErrors()) {
            return "showpage";
        } else {
            double totalCost = service.calculateTotalCost(order);
            model.put("customerName", order.getCustomerName());
            model.put("totalCost", totalCost);
            return "billdesk";
        }
    }
}

```

```
}
```

## INTERNATIONALIZATIONCONFIG.JAVA

```
package com.controller;

import java.util.Locale;

import org.springframework.context.MessageSource;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.support.ReloadableResourceBundleMessageSource;
import org.springframework.validation.beanvalidation.LocalValidatorFactoryBean;
import org.springframework.web.servlet.LocaleResolver;
import org.springframework.web.servlet.config.annotation.InterceptorRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
import org.springframework.web.servlet.i18n.LocaleChangeInterceptor;
import org.springframework.web.servlet.i18n.SessionLocaleResolver;

@Configuration
public class InternationalizationConfig implements WebMvcConfigurer {

    //Create a SessionLocaleResolver object and set the default locale to English
    return the SessionLocaleResolver object
    @Bean
    public LocaleResolver localeResolver() {
        SessionLocaleResolver localeResolver = new SessionLocaleResolver();
        localeResolver.setDefaultLocale(Locale.US);
        return localeResolver;
    }

    // Create LocaleChangeInterceptor object and set the parameter name as
    language
    // and return the localeChangeInterceptor
    @Bean
    public LocaleChangeInterceptor localeChangeInterceptor() {
        LocaleChangeInterceptor interceptor = new LocaleChangeInterceptor();
        interceptor.setParamName("language");
        return interceptor;
    }

    @Bean
    public LocalValidatorFactoryBean getValidator() {
        LocalValidatorFactoryBean bean = new LocalValidatorFactoryBean();
        bean.setValidationMessageSource(messageSource());
        return bean;
    }

    @Bean
    public MessageSource messageSource() {
        ReloadableResourceBundleMessageSource messageSource = new
        ReloadableResourceBundleMessageSource();
```

```

        messageSource.setBasename("classpath:messages");
        messageSource.setDefaultEncoding("UTF-8");
        return messageSource;
    }

    // register the LocaleChangeInterceptor
    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(localeChangeInterceptor());
    }
}

```

## MOBILEACCESSORIESSHOPAPPLICATION

```

package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;

@ComponentScan("com")
public class MobileAccessoriesShopApplication {

    public static void main(String[] args) {
        SpringApplication.run(MobileAccessoriesShopApplication.class, args);
    }
}

```

## ORDER.JAVA

```

package com.model;

import javax.validation.constraints.NotBlank;
import javax.validation.constraints.Positive;

//pojo class with required attributes, getters and setters
public class Order {
    @NotBlank(message="{error.customerName.blank}")
    private String customerName;
    @NotBlank(message="{error.contactNumber.blank}")
    private String contactNumber;
    private String productName;
    @Positive(message="{error.quantity.negative}")
    private int quantity;

    public String getCustomerName() {
        return customerName;
    }
    public void setCustomerName(String customerName) {
        this.customerName = customerName;
    }
}

```

```

    }
    public String getContactNumber() {
        return contactNumber;
    }
    public void setContactNumber(String contactNumber) {
        this.contactNumber = contactNumber;
    }
    public String getProductName() {
        return productName;
    }
    public void setProductName(String productName) {
        this.productName = productName;
    }
    public int getQuantity() {
        return quantity;
    }
    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }
}

```

## BILLSERVICE.JAVA

```

package com.service;

import org.springframework.stereotype.Service;

import com.model.Order;

//use appropriate annotation to configure BillService as a Service
@Service
public class BillService {

    // calculate the totalCost and return the cost
    public double calculateTotalCost(Order order) {
        double totalCost = 0.0;
        double cost = 0.0;
        int perProductPrice = 0;
        if (order.getProductName().equalsIgnoreCase("HeadPhone")) {
            perProductPrice = 450;
        }
        else if (order.getProductName().equalsIgnoreCase("TravelAdapter")) {
            perProductPrice = 1000;
        }
        else if (order.getProductName().equalsIgnoreCase("MemoryCard")) {
            perProductPrice = 300;
        }
        else if (order.getProductName().equalsIgnoreCase("PenDrive")) {
            perProductPrice = 650;
        }
        else if (order.getProductName().equalsIgnoreCase("USBCable")) {
            perProductPrice = 800;
        }
        cost = order.getQuantity() * perProductPrice ;
        totalCost = cost +(cost*3)/100;
    }
}

```

```

        // fill the code
    return totalCost;
}

}

package com.service;

import org.springframework.stereotype.Service;

import com.model.Order;

//use appropriate annotation to configure BillService as a Service
@Service
public class BillService {

    // calculate the totalCost and return the cost
    public double calculateTotalCost(Order order) {
        double totalCost = 0.0;
        double cost = 0.0;
        int perProductPrice = 0;
        if (order.getProductName().equalsIgnoreCase("HeadPhone")) {
            perProductPrice = 450;
        }
        else if (order.getProductName().equalsIgnoreCase("TravelAdapter")) {
            perProductPrice = 1000;
        }
        else if (order.getProductName().equalsIgnoreCase("MemoryCard")) {
            perProductPrice = 300;
        }
        else if (order.getProductName().equalsIgnoreCase("PenDrive")) {
            perProductPrice = 650;
        }
        else if (order.getProductName().equalsIgnoreCase("USBCable")) {
            perProductPrice = 800;
        }
        cost = order.getQuantity() * perProductPrice ;
        totalCost = cost +(cost*3)/100;
        // fill the code
        return totalCost;
    }

}

```

## APPLICATION.PROPERTIES

```

#Do not change the configurations
#value for server.port can be changed
#include other needed configurations
server.port=9095
spring.mvc.view.prefix = /WEB-INF/views/
spring.mvc.view.suffix = .jsp
spring.mvc.static-path-pattern=/resources/**

```

## Message\_de.properties

```
label.customerName=Customer Name in German
```

```
label.contactNumber=Contact Number in German
label.productName=Product Name in German
label.quantity=Quantity in German

error.customerName.blank=Customer Name cannot be empty in German
error.contactNumber.blank=Contact Number cannot be empty in German
error.quantity.negative=Quantity should not be zero or a negative number in German
```

### message\_fr.properties

```
label.customerName=Customer Name in French
label.contactNumber=Contact Number in French
label.productName=Product Name in French
label.quantity=Quantity in French

error.customerName.blank=Customer Name cannot be empty in French
error.contactNumber.blank=Contact Number cannot be empty in French
error.quantity.negative=Quantity should not be zero or a negative number in French
```

### messages.properties

```
label.customerName=Customer Name in English
label.contactNumber=Contact Number in English
label.productName=Product Name in English
label.quantity=Quantity in English

error.customerName.blank=Customer Name cannot be empty in English
error.contactNumber.blank=Contact Number cannot be empty in English
error.quantity.negative=Quantity should not be zero or a negative number in English
```

## MOBILEACCESSORIESSHOPAPPLICATIONTEST

```
//package com.example.demo;
//
//import org.junit.jupiter.api.Test;
//import org.springframework.boot.test.context.SpringBootTest;
//
//{@SpringBootTest
//class MobileAccessoriesShopApplicationTests {
//
//    @Test
//    void contextLoads() {
//    }
//
//}}
```

## BILLDESK.JSP

```
<%@page isELIgnored="false" %>
<html>
<body bgcolor="Lavender">

<h2> Hi ${customerName}!!! You need to pay Rs.${totalCost}</h2>

</body>
</html>
```

## SHOWPAGE.JSP

```
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>
<%@ taglib prefix="sf" uri="http://www.springframework.org/tags/form"%>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1" isELIgnored="false"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<style>
.center {
    margin-left: auto;
    margin-right: auto;
}
</style>

</head>
<body style="background-color: Lavender">
    <h1>
        <center>Welcome to Mobile Accessories Store</center>
    </h1>

    <!-- Do not change the below line -->
    <a href="/showPage?Language=en">English</a>|
    <a href="/showPage?Language=de">German</a>|
    <a href="/showPage?Language=fr">French</a>
</align>
<!-- Design the page as per the requirements-->

<form:form modelAttribute="order" method="post" action="/billDesk">
```

```


|                                                                                                                                                                                                                                       |                                                                                                                                                               |                                                      |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------|
| <spring:message code="Label.customerName" /></td> <td>&lt;form:input path="customerName" id="customerName" name="customerName" /&gt;&lt;/td&gt; <td>&lt;form:errors path="customerName"&gt;&lt;/form:errors&gt;&lt;/td&gt;</td> </td> | <form:input path="customerName" id="customerName" name="customerName" /></td> <td>&lt;form:errors path="customerName"&gt;&lt;/form:errors&gt;&lt;/td&gt;</td> | <form:errors path="customerName"></form:errors></td> |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------|

| <spring:message code="Label.contactNumber" /></td>  <form:input path="contactNumber" id="contactNumber" name="contactNumber" /></td>  <form:errors path="contactNumber"></form:errors></td> | | |
| <spring:message code="Label.productName" /></td>  <form:select path="productName" id="productName" name="productName" items="${productList}" /></td> | |
| <spring:message code="Label.quantity" /></td>  <form:input path="quantity" id="quantity" name="quantity" /></td>  <form:errors path="quantity"></form:errors></td> | | |
| <input type="submit" id="submit" name="submit" value="CalculateCost" /></td>  <input type="reset" id="reset" name="reset" value="Cancel" /></td> | |


</form:&form>
</body>
</html>

```

<https://allhackerranksolutionsbykaira.blogspot.com/search/label/HackerRank>

<https://awesomelivesource.com/project/Java-aid/Hackerrank-Solutions#data-structures>

## MEANDERING ARRAY

```
// Java program to print the array in given order

import java.util.Arrays;

public class GFG {

    // Function which arrange the array.

    static void rearrangeArray(int arr[], int n)

    {

        // Sorting the array elements

        Arrays.sort(arr);

        int[] tempArr = new int[n]; // To store modified array

        // Adding numbers from sorted array to

        // new array accordingly

        int ArrIndex = 0;

        // Traverse from begin and end simultaneously

        for (int i = 0, j = n-1; i <= n / 2 || j > n / 2;

             i++, j--) {

            if(ArrIndex < n)

            {

                tempArr[ArrIndex] = arr[i];

                ArrIndex++;

            }

            if(ArrIndex < n)

            {

                tempArr[ArrIndex] = arr[j];

                ArrIndex++;

            }

        }

    }

}
```

```

}

// Modifying original array

for (int i = 0; i < n; i++)
    arr[i] = tempArr[i];

}

// Driver Code

public static void main(String args[])
{
    int arr[] = { 5, 8, 1, 4, 2, 9, 3, 7, 6 };
    int n = arr.length;
    rearrangeArray(arr, n);
    for (int i = 0; i < n; i++)
        System.out.print(arr[i]+" ");
}
}

```

### **DISTINCT DIGIT NUMBER**

```

// Java implementation of brute

// force solution.

import java.util.LinkedHashSet;

class GFG

{
    // Function to check if the given

    // number has repeated digit or not

    static int repeated_digit(int n)

    {
        LinkedHashSet<Integer> s = new LinkedHashSet<>();

        // Traversing through each digit

        while (n != 0)

        {
            int d = n % 10;

```

```
// if the digit is present
// more than once in the
// number
if (s.contains(d))
{
    // return 0 if the number
    // has repeated digit
    return 0;
}
s.add(d);
n = n / 10;
}

// return 1 if the number has
// no repeated digit
return 1;
}

// Function to find total number
// in the given range which has
// no repeated digit
static int calculate(int L, int R)
{
    int answer = 0;
    // Traversing through the range
    for (int i = L; i < R + 1; ++i)
    {
        // Add 1 to the answer if i has
        // no repeated digit else 0
        answer = answer + repeated_digit(i);
    }
    return answer;
}
```

```

// Driver Code

public static void main(String[] args)
{
    int L = 80, R = 120;

    // Calling the calculate

    System.out.println(calculate(L, R));
}
}

```

### TWIN STRING PROBLEM

```

import java.io.IOException;
import java.util.Scanner;

public class TwinStringProblem {
    //creating method for checking two string
    static boolean[] twins(String[] a, String[] b) {
        boolean[] result = new boolean[a.length];

        for (int i = 0; i < a.length; i++) {
            String aVal = a[i].toLowerCase();
            String bVal = b[i].toLowerCase();
            String[] aValArray = aVal.split("");
            String[] bValArray = bVal.split("");

            for (String s : aValArray) {
                for (int index = 0; index < aValArray.length; index++) {
                    if (bValArray[index].equals(s)) { // checking whether the
index match or not
                        if ((s.indexOf(s) % 2 == 0 && index % 2 == 0) || // ch
ecking even values
                            (s.indexOf(s) % 2 != 0 && index % 2 != 0)) {
                            result[i] = false;
                        } else if ((s.indexOf(s) % 2 == 0 && index % 2 != 0)
                            || (s.indexOf(s) % 2 != 0 && index % 2 == 0))
{
                            result[i] = true;
                            break;
                        }
                    }
                }
            }
        }
        return result;
    }
}

```

```
}

//main Method
public static void main(String[] args) throws IOException {
    Scanner in = new Scanner(System.in);

    int n = Integer.parseInt(in.nextLine().trim());
    String[] a = new String[n];
    for (int i = 0; i < n; i++) {
        a[i] = in.nextLine();
    }

    int m = Integer.parseInt(in.nextLine().trim());
    String[] b = new String[m];
    for (int i = 0; i < m; i++) {
        b[i] = in.nextLine();
    }

    // call twins function
    boolean[] results = twins(a, b);

    for (int i = 0; i < results.length; i++) {
        System.out.println(results[i] ? "Yes" : "No");
    }
}
```

## ARRAY GAME

```
20     * The function accepts INTEGER_ARRAY numbers as parameter.  
21     */  
22  
23     public static long countMoves(List<Integer> numbers) {  
24         // Write your code here  
25         long arrsum,smallest,arr_size=numbers.size();  
26         arrsum=0;  
27         smallest=(long) numbers.get(0);  
28         for(int i=0;i<arr_size;i++){  
29             long temp=(long) numbers.get(i);  
30             if(temp<smallest){  
31                 smallest=temp;  
32             }  
33             arrsum+=temp;  
34         }  
35         long minoperation = arrsum-arr_size*smallest;  
36         return minoperation;  
37     }  
38  
39 }  
40 > public class Solution { ...
```

## PRODUCT SORT

```
10  
11     class FreqComparator implements Comparator<Integer> {  
12         private final Map<Integer, Integer> freq;  
13         FreqComparator(Map<Integer, Integer> tempFreqMap)  
14         {  
15             this.freq = tempFreqMap;  
16         }  
17         @Override  
18         public int compare(Integer k1, Integer k2)  
19         {  
20             int freqCompare = freq.get(k1).compareTo(freq.get(k2));  
21             int valueCompare = k1.compareTo(k2);  
22             if (freqCompare == 0)  
23                 return valueCompare;  
24             else  
25                 return freqCompare;  
26         }  
27     }  
28  
29     class Result {  
30         /* Complete the "lenshort" function below.  
31         ... */  
32     }
```

```
56     * The function accepts INTEGER_ARRAY,
57     */
58
59     /* public static List<Integer> itemsSort(List<Integer> items) {
60     // Write your code here
61     }*/
62
63     public static List<Integer> itemsSort(List<Integer> items){
64         HashMap<Integer, Integer> myMap = new HashMap<>();
65         List<Integer> output = new ArrayList<>();
66         Integer temp=0;
67         for(Integer curr:items){
68             //Integer count = map.getOrDefault(curr, temp);
69             // map.put(curr, count + 1);
70             //output.add(curr);
71             Integer res=myMap.get(curr);
72             if(res==null) myMap.put(curr, myMap.get(curr) + 1);
73             else myMap.put(curr,1);
74         }
75         FreqComparator comp=new FreqComparator(myMap);
76         Collections.sort(items,comp);
77         return items;
78     }
79
80     /**
81      * This method is used to sort the array based on frequency of elements.
82      * @param arr
83      * @return
84     */
85     public static int[] sortBasedOnFrequency(int[] arr) {
86         List<Integer> items = new ArrayList<>(Arrays.asList(arr));
87         List<Integer> sortedItems = itemsSort(items);
88         int[] result = new int[sortedItems.size()];
89         for (int i = 0; i < sortedItems.size(); i++) {
90             result[i] = sortedItems.get(i);
91         }
92         return result;
93     }
94 }
```

Test Results      Custom Input

Run Code      Run Tests      Submit

## ANOTHER SOLUTION

```
49m left
1. Prc
public static List<int> itemsSort(List<int> items)
{
    Console.WriteLine(items.Count);
    Dictionary<int,int> myMap=new Dictionary<int,int>();
    List<int> output=new List<int>();
    foreach(int item in items)
    {
        int count=0;
        if(!myMap.ContainsKey(item))
        {
            myMap.Add(item,1);
        }
        else if(myMap.ContainsKey(item))
        {
            int values=myMap[item];
            values=values+1;
            myMap[item]=values;
        }
    }
    var keysorted=from entry in myMap orderby entry.Key,entry.Value descending select entry;
    var sorted=from entry in keysorted orderby entry.Key,entry.Value select entry;
    foreach(var k in sorted)
    {
        The occ
        The occ
    }
}
```

```
tail YouCompleteMe Language: C# Run Tests Run Code
37     }
38     else if(myMap.ContainsKey(item))
39     {
40         int values=myMap[item];
41         values=values+1;
42         myMap[item]=values;
43     }
44     var keysorted=from entry in myMap orderby entry.Key,entry.Value descending select entry;
45     var sorted=from entry in keysorted orderby entry.Value select entry;
46     foreach(var k in sorted)
47     {
48         for(int i=0;i<k.Value;i++)
49         {
50             output.Add(k.Key);
51         }
52     }
53     return output;
54 }
55 }
56 The
57 occ
58 }
59 > class Solution ...
The
occ
```

**VOWELS**

**RESTRUCTURED ARRAY**

(another solution also available)

The screenshot shows a Java code editor with the following code:

```
language Java 8
● Autocomplete Ready ⓘ
```

```
1 > import java.io.*;
2 > class Result {
3
4     /*
5      * Complete the 'getElements' function below.
6      *
7      * The function is expected to return an INTEGER_ARRAY.
8      * The function accepts following parameters:
9      * 1. INTEGER_ARRAY arr
10     * 2. 2D_INTEGER_ARRAY queries
11     */
12
13     public static List<Integer> getElements(List<Integer> arr, List<List<Integer>> queries) {
14         // Write your code here
15         int s;
16         List<Integer> row = new ArrayList<Integer>();
17         for(int i=0;i<queries.size();i++) {
18             s=((queries.get(i).get(0)-1)*arr.get(0))+queries.get(i).get(1);
19             row.add(arr.get(s));
20         }
21         return (row);
22     }
23 }
24
25 > public class Solution { ... }
```

## IS IT POSSIBLE

```
static LinkedList<Pair<Integer, Integer>> pairs = new LinkedList<Pair<Integer, Integer>>();
```

```
public static String isItPossible(Integer a, Integer b, Integer c, Integer d){
    pairs.addLast(new Pair<Integer, Integer>(a,b));
    while (!pairs.isEmpty()){
        Pair<Integer, Integer> pair = pairs.poll();
        Integer key = pair.getKey();
        Integer value = pair.getValue();
        if(key.equals(c) &&
```

```
    value.equals(d)){  
        return "YES";  
    }  
  
    int sum=key+value;  
  
    if (sum<=c){  
        pairs.addLast(new Pair<Integer, Integer>(sum,value));  
    }  
  
    if (sum<=d){  
        pairs.addLast(new Pair<Integer, Integer>(key,sum));  
    }  
  
}  
  
return "NO";  
}
```

## VOWELS

Language Java 8    Autocomplete Ready

```
1 > import java.io.*;
16
17     // Complete the hasVowels function below.
18     static List<Integer> hasVowels(List<String> strArr, List<String> query) {
19         List<Character> vowels = new ArrayList<>();
20         List<Integer> result = new ArrayList<>();
21         vowels.add('a');
22         vowels.add('e');
23         vowels.add('i');
24         vowels.add('o');
25         vowels.add('u');
26         int count,l,r;
27         Character firstChar, lastChar;
28         int[] vowelArray = new int[strArr.size()];
29         for(int i=0; i < strArr.size(); i++){
30             String str = strArr.get(i);
31             firstChar = str.charAt(0);
32             lastChar = str.charAt(str.length()-1);
33             if(vowels.contains(firstChar) && vowels.contains(lastChar)){
34                 vowelArray[i] = 1;
35             }
36             else
37                 vowelArray[i] = 0;
38         }
39         for(String qry: query) {
40             count = 0;
41             String[] lrStringArray = qry.split("-");
42             l = Integer.parseInt(lrStringArray[0]) -1;
43             r = Integer.parseInt(lrStringArray[1]) - 1;
44             // System.out.println(l+" "+ r)
45             for(int i =l;i<= r;i++){
46                 count += vowelArray[i];
47             }
48             result.add(count);
49         }
50     }
51     return result;
52 }
53 }
54
55 > public static void main(String[] args) throws IOException { ... }
```

Test Results    Custom Input

## ANOTHER SOLUTION



## AUTOSCALE POLICY

The screenshot shows a Java code editor with a dark theme. The code is a static method named `finalInstances` that takes two parameters: `int instances` and `List<Integer> averageUtil`. The method iterates through the `averageUtil` list to calculate a new value for `instances` based on specific conditions. The code uses nested if statements and arithmetic operations like division and multiplication.

```
29
30
31     public static int finalInstances(int instances, List<Integer> averageUtil){
32
33         for(int i=0; i<averageUtil.size(); i++){
34
35             if(averageUtil.get(i) < 25){
36
37                 if(instances != 1 && (instances & 1) == 0){
38
39                     instances = instances / 2;
40                     i = i + 10;
41
42                 }else if(instances != 1 && (instances & 1) == 1){
43
44                     instances = (instances / 2) + 1;
45                     i = i + 10;
46
47                 }else if(averageUtil.get(i) > 60){
48
49                     if( instances <= 100000000 ){
50                         instances = 2 * instances;
51                     }
52
53                 }
54             }
55         }
56         return instances;
57     }
58
59
60 }
61
62 }
```

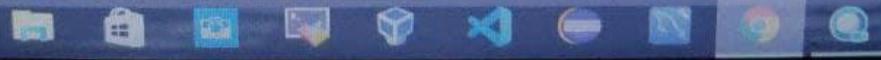
## MINIMUM SWAPS

```
1 > import java.io.*;
2 class Result {
3
4
5
6     /*
7      * Complete the 'minimumSwaps' function below.
8      *
9      * The function is expected to return an INTEGER.
10     * The function accepts INTEGER_ARRAY popularity as parameter.
11     */
12
13
14     public static int minimumSwaps(List<Integer> popularity) {
15         // Write your code here
16         Collections.reverse(popularity);
17         int swapCounter=0;
18         for(int i=0; i<popularity.size(); i++){
19             if(popularity.get(i)==(i+1)){
20                 continue;
21             }
22             Collections.swap(popularity, i, popularity.get(i)-1);
23             swapCounter++;
24             i--;
25         }
26         return swapCounter;
27     }
28
29 }
30
31
32
33
34
35 }
```

```
40 > public class Solution {...
```

Test Results

Custom Input



## ANOTHER SOLUTION

The screenshot shows a code editor window with the following details:

- Language:** C#
- Autocomplete Ready**: A status indicator.
- Code Content:**

```
1 > using System.CodeDom.Compiler;...
16 class Result
17 {
18
19     /*
20      * Complete the 'minimumSwaps' function below.
21      *
22      * The function is expected to return an INTEGER.
23      * The function accepts INTEGER_ARRAY popularity as parameter.
24      */
25
26     public static int minimumSwaps(List<int> popularity)
27     {
28         popularity.Reverse();
29         int swapCounter=0;
30         for(int i=0;i<popularity.Count;i++)
31         {
32             if(popularity[i]==(i+1))
33             {
34                 continue;
35             }
36             int temp = popularity[popularity[i]-1];
37             popularity[popularity[i]-1]=popularity[i];
38             popularity[i]=temp;
39             swapCounter++;
40             i--;
41         }
42     }
43     return swapCounter;
44 }
45 }
46
47 }
48 > class Solution...
```

## ARE ALMOST EQUIVALENT

The screenshot shows a Java code editor with the following code:

```
public static List<String> areAlmostEquivalent(List<String> s, List<String> t) {  
    // Write your code here  
    List<String> result = new ArrayList<String>();  
    for(int j=0; j<s.size();j++){  
        String x=s.get(j);  
        String y=t.get(j);  
        if(x.length() != y.length()){  
            result.add("NO");  
        }  
        else{  
            Map<Character, Integer> mapx= new HashMap<Character, Integer>();  
            Map<Character, Integer> mapy= new HashMap<Character, Integer>();  
            for(int i = 0;i<x.length();i++){  
                if(mapx.containsKey(x.charAt(i))){  
                    mapx.put(x.charAt(i),mapx.get(x.charAt(i))+1);  
                }  
                else{  
                    mapx.put(x.charAt(i), 1);  
                }  
            }  
            for(int i=0;i<y.length();i++){  
                if(mapy.containsKey(y.charAt(i))){  
                    mapy.put(y.charAt(i),mapy.get(y.charAt(i))+1);  
                }  
                else{  
                    mapy.put(y.charAt(i), 1);  
                }  
            }  
            int count=0;  
            for(char p : mapx.keySet()){  
                if(mapy.containsKey(p)){  
                    if(Math.abs(mapx.get(p)-mapy.get(p)) >= 4){  
                        count++;  
                    }  
                }  
            }  
            for(char q : mapy.keySet()){  
                if(mapx.containsKey(q)){  
                    if(Math.abs(mapx.get(q)-mapy.get(q)) >= 4){  
                        count++;  
                    }  
                }  
                else if(mapy.get(q)>=4){  
                    count++;  
                }  
            }  
            result.add(count==0? "YES" : "NO");  
        }  
    }  
    return result;  
}
```

Below the code editor are tabs for "Test Results" and "Custom Input". At the bottom right are buttons for "Run Code", "Run Tests", and "Submit". The status bar at the bottom indicates "Line: 14 Col: 3".

The screenshot shows a Java code editor with the following code:

```
int count=0;  
for(char p : mapx.keySet()){  
    if(mapy.containsKey(p)){  
        if(Math.abs(mapx.get(p)-mapy.get(p)) >= 4){  
            count++;  
        }  
    }  
    else if(mapx.get(p)>=4){  
        count++;  
    }  
}  
for(char q : mapy.keySet()){  
    if(mapx.containsKey(q)){  
        if(Math.abs(mapx.get(q)-mapy.get(q)) >= 4){  
            count++;  
        }  
    }  
    else if(mapy.get(q)>=4){  
        count++;  
    }  
}  
result.add(count==0? "YES" : "NO");  
return result;
```

Below the code editor are tabs for "Test Results" and "Custom Input". At the bottom right are buttons for "Run Code", "Run Tests", and "Submit". The status bar at the bottom indicates "Line: 14 Col: 3".

## ARRANGING COINS

Language Java 8 Autocomplete Ready ⓘ

```
24     public static void arrangeCoins(List<Long> coins) {  
25         // Write your code here  
26         for(Long a: coins){  
27             int res = (int)(Math.sqrt(8*a+1));  
28             Integer res = (int)((temp-1)/2);  
29             System.out.println(res);  
30         }  
31     }  
32 }
```

Test Results Custom Input ⌂ R

## COUNT DUPLICATE ELEMENTS

The screenshot shows a Java code editor interface. At the top, there are tabs for "Activity Details - SumTotal" and "External Site". Below the tabs, the code language is set to "Java 8" and the status bar indicates "Autocomplete Ready". The code itself is a Java function named "countDuplicate" that takes a list of integers as input. It uses a nested loop to compare each element with every other element in the list, adding them to a list if they are found to be equal. Finally, it returns the size of this list, which represents the count of duplicate elements. The code is annotated with Javadoc-style comments at the top.

```
* The function is expected to return an INTEGER.
* The function accepts INTEGER_ARRAY numbers as parameter.
*/
public static int countDuplicate(List<Integer> numbers) {
    // Write your code here
    int[] arr=new int[numbers.size()];
    for(int i=0;i<numbers.size();i++){
        arr[i]=numbers.get(i);
    }
    List<Integer> li=new ArrayList<Integer>();
    for(int i=0;i<arr.length;i++){
        for(int j=i+1;j<arr.length;j++){
            if(arr[i]==arr[j]){
                li.add(arr[i]);
            }
        }
    }
    List<Integer> listwithout=new ArrayList<>(new HashSet<>(li));
    return listwithout.size();
}
}
> public class Solution { ... }
```

Test Results    Custom Input    Run Code    Run Test

## ANOTHER SOLUTION

```
public static int countDuplicate(List<int> numbers)
{
    Dictionary<int,int> frequency = new Dictionary<int, int>();
    for(int i=0; i<numbers.Count;i++){
        if(frequency.ContainsKey(numbers[i])){
```

```
if(frequency.ContainsKey(numbers[i])){
    frequency[numbers[i]] = frequency[numbers[i]]+1;
}
else{
    frequency.Add(numbers[i],1);
}
}

List<int> result = new List<int>();
foreach(var keys in frequency.Keys){
    if(frequency[keys]>1){
        result.Add(keys);
    }
}
return result.Count;
}
```

## BINARY NUMBER IN A LINKED LIST

```
75     * }
76     *
77     */
78
79     public static long getNumber(SinglyLinkedListNode binary) {
80         // Write your code here
81         long ans = 0;
82         SinglyLinkedListNode a = binary;
83         while(a!=null){
84             ans = (ans<<1)+(a.data);
85             a = a.next;
86         }
87         return ans;
88
89     }
90
91 }
```

## DISTINCT NUMBERS

```
static void countNumbers(List<List<Integer>> arr)
{
    int A[][]=new int[arr.size()][2];
    int x=0;
    for(List<Integer> item:arr)
    {
        for(int i=0;i<item.size();i++)
        {
            A[x][i]=item.get(i);
        }
        x++;
    }
    for(int i=0;i<arr.size();i++)
    {
        int l=A[i][0];
        int r=A[i][1];
        long count=0;
        for(int j=l;j<=r;j++)
        {
            int num=j;
            boolean visited[]=new boolean[10];
            while(num!=0)
            {
                if(visited[num%10])
                    break;
                visited[num%10]=true;
                num=num/10;
            }
            if(num==0)
                count++;
        }
        System.out.println(count);
    }
}
public static void main(String[] args) throws IOException {
```

```

25
26     public static int numPlayers(int k, List<Integer> scores) {
27         // Write your code here
28         if(k<=0)
29             return 0;
30         Collections.sort(scores, Collections.reverseOrder());
31         int rank=1;
32         int res=0;
33         for(int i=0;i<scores.size();i++){
34             if(i==0){
35                 rank=1;
36             }
37             else if(scores.get(i)!=scores.get(i-1)){
38                 rank=i+1;
39             }
40             if(rank<=k && scores.get(i)>0)
41                 res++;
42             else break;
43         }
44         return res;
45     }
46 }
47 }
48 > public class Solution {

```

## competitive gaming soln 2

3. Competitive Gaming

A group of friends are playing a video game together. During the game, each player earns a number of points. At the end of a round, players who achieve at least a certain rank get to "level up" their characters to gain increased abilities.

Given the scores of the players at the end of a round, how many players will be able to level up?

**Note:** Players with equal scores will have equal ranks, but the player with the next lower score will be ranked based on the position within the list of all players' scores. For example, if there are four players, and three players tie for first place, their ranks are 1, 1, 1, and 4.

**Note:** No player with a score of 0 can level up, regardless of rank.

**Example**

n = 4

```

21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45

```

Language: Java 7    Autocomplete Ready

Test Results    Custom Input    Run Code    Run Tests    2020-03-22 16:53

## Consolidated partitions

The screenshot shows a code editor window with the following details:

- Title Bar:** One Cognizant, Email - Das, Payel L..., iPRIMED LVDS: Log..., Cognizant, Chat | Microsoft Tea..., Attachments - One..., MyPractice, Other bookmarks, Reading list.
- Language:** Java 8
- Code Content:**

```
21  * The function accepts following parameters
22  * 1. INTEGER_ARRAY used
23  * 2. INTEGER_ARRAY totalCapacity
24
25  public static int minPartitions(List<Integer> used, List<Integer> totalCapacity) {
26      // Write your code here
27      int siz = used.size();
28      int sum=0;
29      int count=0;
30      Collections.sort(totalCapacity);
31      Collections.reverse(totalCapacity);
32      int[] u= new int[siz];
33      int[] t=new int[siz];
34      for(int i=0; i<siz; i++){
35          u[i] = used.get(i);
36          t[i] = totalCapacity.get(i);
37          sum=sum+u[i];
38      }
39      for(int j=0; j<siz; j++){
40          sum=sum-t[j];
41          count++;
42          if(sum <= 0){
43              break;
44          }
45      }
46      return count;
47  }
48
49 }
50 > public class Solution { ... }
```
- Bottom Bar:** Run Code, Run Tests, Submit, Line: 14 Col: 1, Custom Input, ENG 16:12.

consolidate partitions soln 2

The screenshot shows a computer monitor displaying a Java code editor on a Hackerrank challenge page. The challenge title is "1. Consolidating Partitions". The code is a Java program that takes two integer arrays as input: "used" and "totalCapacity". It calculates the minimum number of moves required to move all data from the "used" array to the "totalCapacity" array. The code uses Java 8 features like streams and arrays. The editor shows code lines 15 to 38. Below the code editor are tabs for "Test Results" and "Custom Input". At the bottom right, there are buttons for "Run Code", "Run Tests", and "Submit", along with a timestamp "2021/03/22 16:52".

```
15  /*Integer sum = used.stream()
16  .mapToInt(Integer::intValue).sum();*/
17  int partitions = used.size();
18  int counterTotal = 0;
19  int countTotal=0;
20  Collections.sort(totalCapacity);
21  Collections.reverse(totalCapacity);
22  Int[] arr1 = new int[partitions];
23  Int[] arr2 = new int[partitions];
24  for(int i=0;i<partitions;i++) {
25      arr1[i] = used.get(i);
26      arr2[i] = totalCapacity.get(i);
27      counterTotal=counterTotal+arr1[i];
28  }
29  for(int j=0;j<partitions;j++) {
30      counterTotal=counterTotal-arr2[j];
31      countTotal++;
32      if(counterTotal<=0) {
33          break;
34      }
35  }
36  return countTotal;
37 }
38 }
```

## Construction management

Hackerrank

rs/4629/tabc2h/questions/59554n38

### Management

In building a new neighborhood, and they to the design. Each house will be built using trials (e.g., wood, brick, or concrete), but no n be made of the same material. Because varying size and complexity, the cost of the se varies. Given the cost of using each h, what is the minimum cost needed to...  
house?

there are  $n = 3$  houses to be built. Also,  $\text{cost} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{bmatrix}$ , denoting the cost of materials for each of murn cost to build all the houses is 4, as seen

The cheapest material is the first one, which at house, the materials cost the same as with the same material can't be used because the de. The next best option is the second material, the cheapest material for the third house is with costs 1. Therefore, the total cost to build all  $t = 4$ .

minCost in the editor below.

swing-parameter

Dimensions: 3 x 3 x 3

Language: Java 8    Autocomplete Ready

```
1 > import java.io.*;
2
3 class Result {
4
5     /*
6      * Complete the 'minCost' function below.
7      *
8      * The function is expected to return an INTEGER.
9      *
10     * The function accepts 3D_INTEGER_ARRAY cost as parameter.
11     */
12
13
14     public static int minCost(List<List<Integer>> cost) {
15         int len,min_cost;
16         Integer[][] array=new Integer[cost.size()]();
17         Integer[] array1=new Integer[0];
18         for(int i=0;i<cost.size();i++){
19             array[i]=cost.get(i).toArray(array1);
20         }
21         for(int i=1;i<array.length;i++){
22             array[i][0]=Math.min(array[i-1][1],array[i-1][2]);
23             array[i][1]=Math.min(array[i-1][0],array[i-1][2]);
24             array[i][2]=Math.min(array[i-1][0],array[i-1][1]);
25         }
26         len=array.length;
27         min_cost=Math.min(array[len-1][0],Math.min(array[len-1][1],array[len-1][2]));
28         return min_cost;
29     }
30
31     public class Solution {
32
33     }
34
35 }
36
37
38
39
40
41
42
43
44
45 > public class Solution {
```

Test Results    Custom Input

SLOWEST KEY PRESS

**2. Slowest Key Press**

Engineers have redesigned a keypad used by ambulance drivers in urban areas. In order to determine which key takes the longest time to press, the keypad is tested by a driver. Given the results of that test, determine which key takes the longest to press.

**Example**

`keyTimes = [[0, 2], [1, 5], [0, 9], [2, 15]]`

Elements in `keyTimes[i][0]` represent encoded characters in the range `ascii[a-z]` where `a = 0, b = 1, ..., z = 25`. The second element, `keyTimes[i][1]` represents the time the key is pressed since the start of the test. The elements will be given in ascending time order. In the example, keys pressed, in order are `0102`, encoded = `abac` at times `2, 5, 9, 15`. From the start time, it took `2 - 0 = 2` to press the first key, `5 - 2 = 3` to press the second, and so on. The longest time it

```
public static char slowestKey(List<List<Integer>> keyTimes) {
    int arr[][] = new int[keyTimes.size()][2];
    int arr2[][] = new int[keyTimes.size()][2];

    for(int i=0;i<keyTimes.size();i++) {
        List<Integer> list = keyTimes.get(i);
        for(int j=0;j<2;j++) {
            arr[i][j] = list.get(j);
            arr2[i][j] = list.get(j);
        }
    }

    int prev = 0;
    for(int i=0;i<keyTimes.size();i++) {
        arr2[i][1] = arr[i][1]-prev;
        prev = arr[i][1];
    }

    int max = Integer.MIN_VALUE;
    int ans = 0;
    for(int i=0;i<keyTimes.size();i++) {
        int value = arr2[i][1];
        if(value>max) {
            ans = arr2[i][0];
            max=arr2[i][1];
        }
    }

    int result = 97+ans;
    char c= (char)result;
    return c;
}
```

Line: 18 Col: 15  
Test Results Custom Input Run Code Run Tests Submit 2021/03/22 16:53 453 PM 22-Mar-21

**2. Slowest Key Press**

Engineers have redesigned a keypad used by ambulance drivers in urban areas. In order to determine which key takes the longest time to press, the keypad is tested by a driver. Given the results of that test, determine which key takes the longest to press.

**Example**

`keyTimes = [[0, 2], [1, 5], [0, 9], [2, 15]]`

Elements in `keyTimes[i][0]` represent encoded characters in the range `ascii[a-z]` where `a = 0, b = 1, ..., z = 25`. The second element, `keyTimes[i][1]` represents the time the key is pressed since the start of the test. The elements will be given in ascending time order. In the example, keys pressed, in order are `0102`, encoded = `abac` at times `2, 5, 9, 15`. From the start time, it took `2 - 0 = 2` to press the first key, `5 - 2 = 3` to press the second, and so on. The longest time it

```
int prev = 0;
for(int i=0;i<keyTimes.size();i++) {
    arr2[i][1] = arr[i][1]-prev;
    prev = arr[i][1];
}

int max = Integer.MIN_VALUE;
int ans = 0;
for(int i=0;i<keyTimes.size();i++) {
    int value = arr2[i][1];
    if(value>max) {
        ans = arr2[i][0];
        max=arr2[i][1];
    }
}

int result = 97+ans;
char c= (char)result;
return c;
```

Line: 10 Col: 15  
Test Results Custom Input Run Code Run Tests Submit 2021/03/22 16:53 453 PM 22-Mar-21

AREA OF THE BOX

The screenshot shows a Java code editor with the following code:

```
// Write your code here
int n, m ;
int min ;
long res=0;
List<Long> fin = new ArrayList<>();

for(List<Integer> lst : queries) {
    n = lst.get(0);
    m = lst.get(1);
    min = Math.min(m,n);
    res = 0 ;

    for(int i=1; i<=min; ++i) {
        res += (n-i+1) * (m-i+1);
    }
    fin.add(res);
}

return fin;
}
```

The code is intended to calculate the number of ways to partition two integers  $n$  and  $m$  into  $i$  parts each, where the order of parts does not matter. The result is stored in a list `fin`.

At the bottom of the editor, there are tabs for "Test results", "Custom Input", and "Run Tests". The "Run Tests" button is highlighted in blue.

## CAR INHERITANCE

```
class WagonR extends Car{  
    int mileage;  
    public WagonR(Integer mileage){  
        super(false, "4");  
        this.mileage = mileage;  
    }  
  
    @Override  
    public String getMileage(){  
        String mil=Integer.toString(mileage);  
        return mileage+" kmpl";  
    }  
}
```

```
class HondaCity extends Car{  
    int mileage;  
    public HondaCity(Integer mileage){  
        super(true, "4");  
        this.mileage = mileage;  
    }  
  
    @Override  
    public String getMileage(){  
        String mil=Integer.toString(mileage);  
        return mil+" kmpl";  
    }  
}
```

```
import java.util.*;
public class tourism {
    static String name;
    static String place;
    static int days;
    static int tickets;
    static double price = 0.00;
    static double total = 0.00;
    public static void main(String[] args){
        Scanner in = new Scanner(System.in);
        System.out.println("Enter the passenger name");
        name = in.nextLine();
        System.out.println("Enter the place name");
        place=in.nextLine();
        if(place.equalsIgnoreCase("beach")
            ||place.equalsIgnoreCase("pilgrimage")||
place.equalsIgnoreCase("heritage")||place.equalsIgnoreCase("Hills")||
place.equalsIgnoreCase("palls")||place.equalsIgnoreCase("adventure")){
            System.out.println("Enter the number of days");
            days = in.nextInt();
            if(days>0){
                System.out.println("Enter the number of Tickets");
                tickets = in.nextInt();
                if(tickets>0){
                    if(place.equalsIgnoreCase("beach")){
                        price = tickets*270;
                        if(price>1000){
                            total = 85*price/100;
                            System.out.printf("Price:%.2f",total);
                        }
                        else {
                            System.out.printf("Price:%.2f",price);
                        }
                    }
                    else if(place.equalsIgnoreCase("prilgrimage")){
                        price = tickets*350;
                        if(price>1000){
                            total = 85*price/100;
                            System.out.printf("Price:%.2f",total);
                        }
                        else {
                            System.out.printf("Price:%.2f",price);
                        }
                    }
                    else if(place.equalsIgnoreCase("heritage")){
                        price = tickets*430;
                        if(price>1000){
                            total = 85*price/100;
                            System.out.printf("Price:%.2f",total);
                        }
                        else {
                            System.out.printf("Price:%.2f",price);
                        }
                    }
                    else if(place.equalsIgnoreCase("hills")){
                        price = tickets*780;
                        if(price>1000){
                            total = 85*price/100;
                            System.out.printf("Price:%.2f",total);
                        }
                        else {
                            System.out.printf("Price:%.2f",price);
                        }
                    }
                }
            }
        }
    }
}
```

```

else if(place.equalsIgnoreCase("palls")){
    price = tickets*1200;
    if(price>1000){
        total = 85*price/100;
        System.out.printf("Price:%.2f",total);
    }
    else {
        System.out.printf("Price:%.2f",price);
    }
}
else {
    price = tickets*4500;
    if(price>1000){
        total = 85*price/100;
        System.out.printf("Price:%.2f",total);
    }
    else {
        System.out.printf("Price:%.2f",price);
    }
}

else{
    System.out.println(tickets+" is an Invalid no. of tickets");
}
}
else{
    System.out.println(days+" is an Invalid no. of days");
}
}
else {
    System.out.println(place+" is an Invalid place");
}
}
}

```

## InvalidCarrierException:

```
public class InvalidCarrierException extends Exception{
    //FILL THE CODE HERE
    public InvalidCarrierException (String message){
        super(message);
    }
}
```

## Passenger:

```
//DO NOT EDIT OR ADD ANY CODE
public class Passenger {

    private String passengerName;
    private long phoneNumber;
    private String emailId;
    private String carrierName;
    private String dateOfJourney;
    private String source;
    private String destination;

    public Passenger() {
        super();
        // TODO Auto-generated constructor stub
    }

    public Passenger(String passengerName, long phoneNumber, String emailId,
String carrierName, String dateOfJourney,
        String source, String destination) {
        super();
        this.passengerName = passengerName;
        this.phoneNumber = phoneNumber;
        this.emailId = emailId;
        this.carrierName = carrierName;
        this.dateOfJourney = dateOfJourney;
        this.source = source;
        this.destination = destination;
    }

    public String getPassengerName() {
```

```
        return passengerName;
    }
    public void setPassengerName(String passengerName) {
        this.passengerName = passengerName;
    }

    public long getPhoneNumber() {
        return phoneNumber;
    }
    public void setPhoneNumber(long phoneNumber) {
        this.phoneNumber = phoneNumber;
    }
    public String getEmailId() {
        return emailId;
    }
    public void setEmailId(String emailId) {
        this.emailId = emailId;
    }
    public String getCarrierName() {
        return carrierName;
    }
    public void setCarrierName(String carrierName) {
        this.carrierName = carrierName;
    }
    public String getDateOfJourney() {
        return dateOfJourney;
    }
    public void setDateOfJourney(String dateOfJourney) {
        this.dateOfJourney = dateOfJourney;
    }
    public String getSource() {
        return source;
    }
    public void setSource(String source) {
        this.source = source;
    }
    public String getDestination() {
        return destination;
    }
    public void setDestination(String destination) {
        this.destination = destination;
    }
}
```

## PassengerCategorization:

```
//DO NOT ADD OR EDIT ANY CODE HERE
import java.util.List;
@FunctionalInterface
public interface PassengerCategorization {
    abstract public List<Passenger> retrievePassenger_BySource(List<Passenger>
passengerRecord, String source);

}
```

## PassengerUtility:

```
import java.util.List;
import java.io.*;
import java.util.*;

public class PassengerUtility {

    public List<Passenger> fetchPassenger(String filePath) throws Exception{

        //FILL THE CODE HERE
        List<Passenger> list = new ArrayList<Passenger>();
        String line = "";
        String splitBy = ",";

        BufferedReader br = new BufferedReader(new FileReader(filePath));
        while((line=br.readLine())!=null){
            String[] p = line.split(splitBy);
            Passenger passenger = new
Passenger(p[0],Long.parseLong(p[1]),p[2],p[3],p[4],p[5],p[6]);
            if(isValidCarrierName(passenger.getCarrierName())){
                list.add(passenger);
            }
        }

        return list;
    }

    public boolean isValidCarrierName (String carrierName)
```

```

{
    //FILL THE CODE HERE
    String temp = carrierName;
    if((temp.toLowerCase()).equals("bella")){
        return true;
    }else{
        try{
            throw new InvalidCarrierException(carrierName+" is an Invalid
carrier name.");
        }
        catch(InvalidCarrierException e){
            System.out.println(e.getMessage());
        }
    }
    return false;
}

}

```

## SkeletonValidator:

```

import java.lang.reflect.Method;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.stream.Stream;

/**
 * @author TJ
 *
 * This class is used to verify if the Code Skeleton is intact and not
modified by participants thereby ensuring smooth auto evaluation
 */
public class SkeletonValidator {

    public SkeletonValidator() {

        validateClassName("PassengerCategorization");
        validateClassName("Passenger");
        validateClassName("InvalidCarrierException");
        validateClassName("PassengerUtility");
    }
}

```

```

validateMethodSignature(
    "retrievePassenger_BySource:java.util.List",
    "PassengerCategorization");
validateMethodSignature(
    "fetchPassenger:java.util.List",
    "PassengerUtility");
validateMethodSignature(
    "isValidCarrierName:boolean",
    "PassengerUtility");
validateMethodSignature(
    "searchPassengerRecord:PassengerCategorization",
    "UserInterface");
}

private static final Logger LOG = Logger.getLogger("SkeletonValidator");

protected final boolean validateClassName(String className) {

    boolean iscorrect = false;
    try {
        Class.forName(className);
        iscorrect = true;
        LOG.info("Class Name " + className + " is correct");

    } catch (ClassNotFoundException e) {
        LOG.log(Level.SEVERE, "You have changed either the " + "class
name/package. Use the correct package "
            + "and class name as provided in the skeleton");

    } catch (Exception e) {
        LOG.log(Level.SEVERE,
            "There is an error in validating the " + "Class Name.
Please manually verify that the "
            + "Class name is same as skeleton before
uploading");
    }
    return iscorrect;
}

protected final void validateMethodSignature(String methodWithExcptn,
String className) {
    Class cls = null;
    try {

        String[] actualmethods = methodWithExcptn.split(",");
        boolean errorFlag = false;
        String[] methodSignature;

```

```

String methodName = null;
String returnType = null;

for (String singleMethod : actualmethods) {
    boolean foundMethod = false;
    methodSignature = singleMethod.split(":");

    methodName = methodSignature[0];
    returnType = methodSignature[1];
    cls = Class.forName(className);
    Method[] methods = cls.getMethods();
    for (Method findMethod : methods) {
        if (methodName.equals(findMethod.getName())) {
            foundMethod = true;
            if
(!findMethod.getReturnType().getName().equals(returnType)) {
                errorFlag = true;
                LOG.log(Level.SEVERE, " You have changed the " +
"return type in '" + methodName
                           + "' method. Please stick to the " +
"keleton provided");

            } else {
                LOG.info("Method signature of " + methodName + "
is valid");
            }
        }
    }
    if (!foundMethod) {
        errorFlag = true;
        LOG.log(Level.SEVERE, " Unable to find the given public
method " + methodName
                           + ". Do not change the " + "given public method
name. " + "Verify it with the skeleton");
    }
}

if (!errorFlag) {
    LOG.info("Method signature is valid");
}

} catch (Exception e) {
    LOG.log(Level.SEVERE,
           " There is an error in validating the " + "method
structure. Please manually verify that the "
                           + "Method signature is same as the skeleton before
uploading");
}

```

```
        }
    }
}
```

## UserInterface:

```
import java.util.*;
import java.io.*;

public class UserInterface{
    public static PassengerCategorization searchPassengerRecord(){

        //FILL THE CODE HERE
        return (list,source)->{
            List<Passenger> result = new ArrayList<Passenger>();
            for(Passenger pass : list){
                if((pass.getSource().toLowerCase()).equals(source.toLowerCase())){
                    result.add(pass);
                }
            }
            return result;
        };
    }

    public static void main(String [] args)
    {
        //VALIDATION STARTS
        new SkeletonValidator();
        //DO NOT DELETE THIS CODE
        //VALIDATION ENDS

        PassengerCategorization pc = searchPassengerRecord();
        //FILL THE CODE HERE
        System.out.println("Invalid Carrier Records are:");
        PassengerUtility pu = new PassengerUtility();
        List<Passenger> list = null;
        try{
            list = pu.fetchPassenger(new String("PassengerRecord.txt"));
        }
        catch(FileNotFoundException e){
            e.printStackTrace();
        }
    }
}
```

```

        catch(IOException e){
            e.printStackTrace();
        }
        catch(Exception e){
            e.printStackTrace();
        }
        System.out.println("Enter the source to search");
        Scanner sc = new Scanner(System.in);
        String inp = sc.next();

        List<Passenger> result = pc.retrievePassenger_BySource(list,inp);
        if(result.size()==0){
            System.out.println("No Passenger Record");
        }
        else{
            for(Passenger passenger: result){
                System.out.println(passenger.getPassengerName()+" "+passenger.getPhoneNumber()+" "+passenger.getDateOfJourney()+" "+passenger.getDestination());
            }
        }
    }
}

```

## PassengerRecord:

Athara,7397439291,athara@gmail.com,Bella,01-01-2021,kochi,Chennai

Neil,9197439291,neil@gmail.com,Bella,01-01-2021,Mumbai,Chennai

Gita,9167448791,gita@gmail.com,Parveen,01-01-2021,chennai,Hyderabad

Manikandan,9167439291,manikandan@gmail.com,Bella,01-01-2021,Bangalore,Hyderabad

Vishnu,9167438791,vishnu@gmail.com,Bella,01-01-2021,Goa,Hyderabad

Venkatesh,3467448791,venky@gmail.com,Bella,01-01-2021,Chennai,Hyderabad

Arun,3467448723,arun@gmail.com,Bella,01-01-2021,hyderabad,Kochi

Anumol,3467448423,anu@gmail.com,bella,01-01-2021,Hyderabad,Kochi

Sharon,9193439291,Sharon@gmail.com,bella,01-01-2021,mumbai,Chennai

Mani,3167439291,manin@gmail.com,bella,01-01-2021,bangalore,Hyderabad

Athira,7397439191,Athira@gmail.com,Stella,01-01-2021,Kochi,Chennai

```
1 //ApplicationUtil
2
3 package com.cts.unoadm.util;
4 import java.util.*;
5 import java.io.*;
6 import java.text.*;
7
8 import com.cts.unoadm.exception.StudentAdmissionException;
9
10 public final class ApplicationUtil {
11
12     /**
13      * @param fileName
14      * @return List<String>
15      * @throws StudentAdmissionException
16      */
17
18     private ApplicationUtil(){}
19     public static List<String> readFile(String fileName) throws StudentAdmissionException {
20         List<String> studentAdmissionList = new ArrayList<String>();
21         //Code here..
22         FileReader fr = null;
23         BufferedReader br = null;
24         try{
25             fr=new FileReader(fileName);
26             br = new BufferedReader(fr);
27             String line = null;
28             while((line=br.readLine())!=null)
29             {
30                 String []res = line.split(",");
31                 String managerApproval = res[7];
32                 Date dtOfCounseling = convertStringToDate(res[2]);
33                 Date dtOfAdmission = convertStringToDate(res[4]);
34                 if(checkIfValidAdmission(dtOfCounseling,dtOfAdmission,managerApproval))
35                 {
36                     studentAdmissionList.add(line);
37                 }
38             }
39         }catch(Exception e){e.printStackTrace();}
40         return studentAdmissionList;
41     }
42
43     /**
44      * @param util
45      *          Date
46      * @return sql Date
47      */
48     public static java.sql.Date convertUtilToSqlDate(java.util.Date uDate) {
49
50         java.sql.Date sDate = new java.sql.Date(uDate.getTime());
51
52         //Code here..
53
54         return sDate;
55
56
57 }
```

```
58     }
59
60    /**
61     * @param inDate
62     * @return Date
63     */
64    public static Date convertStringToDate(String inDate) {
65
66        //Code here..
67        try{
68            SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd", Locale.ENGLISH);
69            return format.parse(inDate);
70        }catch(Exception e){e.printStackTrace();
71            return null;
72        }
73    }
74
75    public static boolean checkIfValidAdmission(Date dtOfCounseling, Date dtOfAdmission,
76 String manager) {
77        boolean admissionValidity = false;
78
79        //Code here..
80        if("Approved".equals(manager) && ((dtOfAdmission.getTime() - dtOfCounseling.getTime()
81        ())/(1000*60*60*24))%365<= 10)
82        {
83            admissionValidity = true;
84        }
85
86    }
87
88
89 //DBConnectionManager
90
91 * Don't change this code
92 */
93 package com.cts.unoadm.util;
94 import java.io.FileInputStream;
95 import java.io.FileNotFoundException;
96 import java.io.IOException;
97 import java.sql.Connection;
98 import java.sql.DriverManager;
99 import java.sql.SQLException;
100 import java.util.Properties;
101
102 import com.cts.unoadm.exception.StudentAdmissionException;
103
104
105 public final class DBConnectionManager {
106
107    public static final String PROPERTY_FILE = "database.properties";
108    public static final String DRIVER = "drivername";
109    public static final String URL = "url";
110    public static final String USER_NAME = "username";
111    public static final String PASSWORD = "password";
112
```

```
113     private static Connection connection = null;
114     private static Properties props = null;
115     private static DBConnectionManager instance = null;
116     /**
117      * @throws StudentAdmissionException
118      */
119     private DBConnectionManager() throws StudentAdmissionException {
120         loadProperties();
121         try {
122             Class.forName(props.getProperty(DRIVER));
123             connection = DriverManager.getConnection(props.getProperty(URL), props.getProperty
124             (USER_NAME),
125                 props.getProperty(PASSWORD));
126         } catch (ClassNotFoundException ex) {
127             ex.printStackTrace();
128             //throw new StudentAdmissionException("Could not find Driver class ", ex.getCause
129             ());
130         } catch (SQLException e) {
131             e.printStackTrace();
132             //throw new StudentAdmissionException("Database Connection Creation Failed",
133             e.getCause());
134         }
135         catch(Exception e)
136         {
137             e.printStackTrace();
138             //throw new StudentAdmissionException("Database Connection Creation Failed",
139             e.getCause());
140         }
141     }
142     /**
143      * @return Connection
144      */
145     public Connection getConnection() {
146         return connection;
147     }
148     /**
149      * @return DBConnectionManager
150      * @throws StudentAdmissionException
151      */
152     public static DBConnectionManager getInstance() throws StudentAdmissionException {
153         // Code here
154         instance = new DBConnectionManager();
155
156         return instance;
157     }
158     /**
159      * @throws StudentAdmissionException
160      */
161     private void loadProperties() throws StudentAdmissionException {
162         FileInputStream inputStream = null;
163         try {
164             inputStream = new FileInputStream(PROPERTY_FILE);
```

```
166         props = new Properties();
167         props.load(inputStream);
168     } catch (FileNotFoundException e) {
169         e.printStackTrace();
170         //throw new StudentAdmissionException("Database Property File Not Found",
171         e.getCause());
171     } catch (IOException e) {
172         e.printStackTrace();
173         //throw new StudentAdmissionException("Exception during property file I/O",
174         e.getCause());
174     } finally {
175         if (inputStream != null) {
176             try {
177                 inputStream.close();
178             } catch (IOException e) {
179                 e.printStackTrace();
180                 //throw new StudentAdmissionException("Exception during property file
181                 I/O", e.getCause());
181             }
182         }
183     }
184 }
185 }
186
187
188 //MainApp
189
190 package com.cts.unoadm.main;
191
192 import com.cts.unoadm.skeletonvalidator.SkeletonValidator;
193 import com.cts.unoadm.service.*;
194 import com.cts.unoadm.util.*;
195 public final class MainApp {
196     private MainApp(){}
197     public static void main(String[] args) {
198         //Don't delete this code
199         //Skeletonvalidatlon starts
200         new SkeletonValidator();
201         //Skeletonvalidation ends
202
203         //Write your code here..
204         try{
205             StudentAdmissionService service = new StudentAdmissionService();
206             System.out.println(service.addStudentAdmissionDetails("inputFeed.txt"));
207             System.out.println(service.searchStudentAdmission("A005"));
208             }catch(Exception e){e.printStackTrace();}
209             //List<StudentAdmission> studentAdmissionList = service.buildStudentAdmissionsList
210             (ApplicationUtil.readFile("inputFeed.txt"));
211             /*for(StudentAdmission e:studentAdmissionList)
212             {
213                 System.out.println(e);
214             }*/
215
216 }
217
218 }
```

```
219 //SkeletonValidator
220
221 package com.cts.unoadm.skeletonvalidator;
222
223 import java.lang.reflect.Array;
224 import java.lang.reflect.Method;
225 import java.util.logging.Level;
226 import java.util.logging.Logger;
227
228 /**
229  * @author t-aarti3
230  *         This class is used to verify if the Code Skeleton is intact and not
231  *         modified by participants thereby ensuring smooth auto evaluation
232  */
233
234 public class SkeletonValidator {
235     private static final Logger LOG = Logger.getLogger("SkeletonValidator");
236     public SkeletonValidator() {
237         validateClassName("com.cts.unoadm.util.DBConnectionManager");
238         validateClassName("com.cts.unoadm.util.ApplicationUtil");
239         validateClassName("com.cts.unoadm.service.StudentAdmissionService");
240         validateClassName("com.cts.unoadm.dao.StudentAdmissionDAO");
241         validateClassName("com.cts.unoadm.vo.StudentAdmission");
242         validateClassName("com.cts.unoadm.exception.StudentAdmissionException");
243
244
245         validateMethodSignature(
246             "addStudentAdmissionDetails:boolean,getAllStudentAdmissionDetails>List",
247             "com.cts.unoadm.dao.StudentAdmissionDAO");
248         validateMethodSignature(
249             "buildStudentAdmissionsList>List,addStudentAdmissionDetails:boolean,calculateT
250             otalCollegeFee:double[],searchStudentAdmission:boolean",
251             "com.cts.unoadm.service.StudentAdmissionService");
252         validateMethodSignature(
253             "readFile>List,convertUtilToSqlDate>Date,convertStringToDate>Date,checkIfValid
254             Admission:boolean",
255             "com.cts.unoadm.util.ApplicationUtil");
256         validateMethodSignature(
257             "getConnection:Connection,getInstance:DBConnectionManager",
258             "com.cts.unoadm.util.DBConnectionManager");
259
260     }
261
262     protected final boolean validateClassName(String className) {
263
264         boolean iscorrect = false;
265         try {
266             Class.forName(className);
267             iscorrect = true;
268             LOG.info("Class Name " + className + " is correct");
269
270         } catch (ClassNotFoundException e) {
271             LOG.log(Level.SEVERE, "You have changed either the " + "class name/package. Use
272             the correct package "
273                     + "and class name as provided in the skeleton");
274     }
```

```
273         } catch (Exception e) {
274             LOG.log(Level.SEVERE,
275                     "There is an error in validating the " + "Class Name. Please manually
276                     verify that the "
277                     + "Class name is same as skeleton before uploading");
278         }
279         return iscorrect;
280     }
281
282     protected final void validateMethodSignature(String methodWithExcptn, String className) {
283         Class cls = null;
284         try {
285
286             String[] actualmethods = methodWithExcptn.split(",");
287             boolean errorFlag = false;
288             String[] methodSignature;
289             String methodName = null;
290             String returnType = null;
291
292             for (String singleMethod : actualmethods) {
293                 boolean foundMethod = false;
294                 methodSignature = singleMethod.split(":");
295
296                 methodName = methodSignature[0];
297                 returnType = methodSignature[1];
298                 cls = Class.forName(className);
299                 Method[] methods = cls.getMethods();
300                 for (Method findMethod : methods) {
301                     if (methodName.equals(findMethod.getName())) {
302                         foundMethod = true;
303                         if (!(findMethod.getReturnType().getSimpleName().equals(returnType)))
304                         {
305                             errorFlag = true;
306                             LOG.log(Level.SEVERE, " You have changed the " + "return type in
307                                     " + methodName
308                                     + "' method. Please stick to the " + "skeleton provided");
309                         }
310                     }
311                 }
312             }
313         }
314         if (!foundMethod) {
315             errorFlag = true;
316             LOG.log(Level.SEVERE, " Unable to find the given public method " +
317                     methodName
318                     + ". Do not change the " + "given public method name. " + "Verify
319                     it with the skeleton");
320     }
321     if (!errorFlag) {
322         LOG.info("Method signature is valid");
323     }
324 }
```

```
325         } catch (Exception e) {
326             LOG.log(Level.SEVERE,
327                     " There is an error in validating the " + "method structure. Please
328                     manually verify that the "
329                     + "Method signature is same as the skeleton before uploading");
330     }
331 }
332 }
333
334 //StudentAdmission
335
336 * Don't change this code
337 */
338 package com.cts.unoadm.vo;
339
340 import java.util.Date;
341
342 public class StudentAdmission {
343     String admissionId;
344     String studentCode;
345     Date dateOfCounseling;
346     String departmentName;
347     Date dateOfAdmission;
348     String preferCollegeHostel;
349     String firstGraduate;
350     String managerApproval;
351     double admissionFee;
352     double tuitionFee;
353     double hostelFee;
354     double totalCollegeFee;
355     String finalStatusOfAdmission;
356
357     public StudentAdmission() {
358         super();
359     }
360
361     public StudentAdmission(String admissionId, String studentCode, Date dateOfCounseling,
362                             String departmentName,
363                             Date dateOfAdmission, String preferCollegeHostel, String firstGraduate, String
364                             managerApproval,
365                             double admissionFee, double tuitionFee, double hostelFee, double totalCollegeFee,
366                             String finalStatusOfAdmission) {
367         super();
368         this.admissionId = admissionId;
369         this.studentCode = studentCode;
370         this.dateOfCounseling = dateOfCounseling;
371         this.departmentName = departmentName;
372         this.dateOfAdmission = dateOfAdmission;
373         this.preferCollegeHostel = preferCollegeHostel;
374         this.firstGraduate = firstGraduate;
375         this.managerApproval = managerApproval;
376         this.admissionFee = admissionFee;
377         this.tuitionFee = tuitionFee;
378         this.hostelFee = hostelFee;
379         this.totalCollegeFee = totalCollegeFee;
380         this.finalStatusOfAdmission = finalStatusOfAdmission;
```

```
379     }
380
381     public String getAdmissionId() {
382         return admissionId;
383     }
384
385     public void setAdmissionId(String admissionId) {
386         this.admissionId = admissionId;
387     }
388
389     public String getStudentCode() {
390         return studentCode;
391     }
392
393     public void setStudentCode(String studentCode) {
394         this.studentCode = studentCode;
395     }
396
397     public Date getDateOfCounseling() {
398         return dateOfCounseling;
399     }
400
401     public void setDateOfCounseling(Date dateOfCounseling) {
402         this.dateOfCounseling = dateOfCounseling;
403     }
404
405     public String getDepartmentName() {
406         return departmentName;
407     }
408
409     public void setDepartmentName(String departmentName) {
410         this.departmentName = departmentName;
411     }
412
413     public Date getDateOfAdmission() {
414         return dateOfAdmission;
415     }
416
417     public void setDateOfAdmission(Date dateOfAdmission) {
418         this.dateOfAdmission = dateOfAdmission;
419     }
420
421     public String getPreferCollegeHostel() {
422         return preferCollegeHostel;
423     }
424
425     public void setPreferCollegeHostel(String preferCollegeHostel) {
426         this.preferCollegeHostel = preferCollegeHostel;
427     }
428
429     public String getFirstGraduate() {
430         return firstGraduate;
431     }
432
433     public void setFirstGraduate(String firstGraduate) {
434         this.firstGraduate = firstGraduate;
435     }
```

```
436
437     public String getManagerApproval() {
438         return managerApproval;
439     }
440
441     public void setManagerApproval(String managerApproval) {
442         this.managerApproval = managerApproval;
443     }
444
445     public double getAdmissionFee() {
446         return admissionFee;
447     }
448
449     public void setAdmissionFee(double admissionFee) {
450         this.admissionFee = admissionFee;
451     }
452
453     public double getTuitionFee() {
454         return tuitionFee;
455     }
456
457     public void setTuitionFee(double tuitionFee) {
458         this.tuitionFee = tuitionFee;
459     }
460
461     public double getHostelFee() {
462         return hostelFee;
463     }
464
465     public void setHostelFee(double hostelFee) {
466         this.hostelFee = hostelFee;
467     }
468
469     public double getTotalCollegeFee() {
470         return totalCollegeFee;
471     }
472
473     public void setTotalCollegeFee(double totalCollegeFee) {
474         this.totalCollegeFee = totalCollegeFee;
475     }
476
477     public String getFinalStatusOfAdmission() {
478         return finalStatusOfAdmission;
479     }
480
481     public void setFinalStatusOfAdmission(String finalStatusOfAdmission) {
482         this.finalStatusOfAdmission = finalStatusOfAdmission;
483     }
484
485     @Override
486     public String toString() {
487         return "Student Admission Details: [admissionId=" + admissionId + ", studentCode=" +
488             studentCode + ", dateOfCounseling=" +
489                 + dateOfCounseling + ", departmentName=" + departmentName + ",
490                 dateOfAdmission=" + dateOfAdmission + ", preferCollegeHostel=" +
491                     + preferCollegeHostel + ", firstGraduate=" + firstGraduate + ",
492                     managerApproval=" + managerApproval
```

```
490             + ", admissionFee=" + admissionFee + ", tuitionFee=" + tuitionFee + ",  
491             hostelFee=" + hostelFee + ", totalCollegeFee=" + totalCollegeFee  
492         + ", finalStatusOfAdmission=" + finalStatusOfAdmission + "]";  
493     }  
494 }  
495  
496 //StudentAdmissionDAO  
497  
498 package com.cts.unoadm.dao;  
499  
500 import java.util.ArrayList;  
501 import java.util.List;  
502  
503 import com.cts.unoadm.exception.StudentAdmissionException;  
504 import com.cts.unoadm.vo.StudentAdmission;  
505 import com.cts.unoadm.util.*;  
506 import java.sql.*;  
507  
508 /*  
509 CREATE DATABASE CTSUNO  
510  
511 CREATE TABLE UNO_ADMISSION(  
512     ADMISSION_ID VARCHAR(50) PRIMARY KEY,  
513     STUDENT_CODE VARCHAR(50)NOT NULL,  
514     DATE_OF_COUNSELING DATE NOT NULL,  
515     DEPARTMENT_NAME VARCHAR(20),  
516     DATE_OF_ADMISSION DATE NOT NULL,  
517     PREFER_COLLEGE_HOSTEL VARCHAR(20),  
518     FIRST_GRADUATE VARCHAR(20),  
519     MANAGER_APPROVAL VARCHAR(10),  
520     ADMISSION_FEE DOUBLE,  
521     TUITION_FEE DOUBLE,  
522     HOSTEL_FEE DOUBLE,  
523     TOTAL_COLLEGE_FEE DOUBLE,  
524     FINAL_STATUS_OF_ADMISSION VARCHAR(20)  
525 );  
526 */  
527 public class StudentAdmissionDAO {  
528  
529     public boolean addStudentAdmissionDetails(List<StudentAdmission> stdAdmissions) throws  
StudentAdmissionException {  
530         boolean recordsAdded = false;  
531         Connection con = DBConnectionManager.getInstance().getConnection();  
532         PreparedStatement ps = null;  
533         try{  
534             //inserting values of list stdAdmissions into database  
535             String query = "insert into UNO_ADMISSION  
                (ADMISSION_ID,STUDENT_CODE,DATE_OF_COUNSELING,DEPARTMENT_NAME,DATE_OF_ADMISSION,PREFER_COLLEGE  
                _HOSTEL,FIRST_GRADUATE,MANAGER_APPROVAL,ADMISSION_FEE,TUITION_FEE,HOSTEL_FEE,TOTAL_COLLEGE_FEE  
                ,FINAL_STATUS_OF_ADMISSION) values(?,?,?,?,?,?,?,?,?,?,?,?,?,?)";  
536             for(StudentAdmission e:stdAdmissions) {  
537                 ps = con.prepareStatement(query);  
538                 ps.setString(1,e.getAdmissionId());  
539                 ps.setString(2,e.getStudentCode());  
540                 ps.setDate(3,ApplicationUtil.convertUtilToSqlDate(e.getDateOfCounseling()));  
541                 ps.setString(4,e.getDepartmentName());  
542             }  
543         } catch (SQLException e) {  
544             e.printStackTrace();  
545         }  
546         if(recordsAdded){  
547             return true;  
548         }  
549     }  
550 }
```

```
542         ps.setDate(5,ApplicationUtil.convertUtilToSqlDate(e.getDateOfAdmission()));
543         ps.setString(6,e.getPreferCollegeHostel());
544         ps.setString(7,e.getFirstGraduate());
545         ps.setString(8,e.getManagerApproval());
546         ps.setDouble(9,e.getAdmissionFee());
547         ps.setDouble(10,e.getTuitionFee());
548         ps.setDouble(11,e.getHostelFee());
549         ps.setDouble(12,e.getTotalCollegeFee());
550         ps.setString(13,e.getFinalStatusOfAdmission());
551         int i = ps.executeUpdate();
552         if(i>0)
553         {
554             recordsAdded = true;
555         }
556         else
557         {
558             recordsAdded = false;
559         }
560     }
561
562
563 }
564 catch(SQLException e)
565 {
566     try{
567         con.rollback();
568     }catch(Exception e1){
569         e.printStackTrace();
570     }
571 }
572 catch(Exception e) {
573     e.printStackTrace();
574     //throw new StudentAdmissionException("Database Value Insertion Failed",
e.getCause());
575 }
576 finally{
577     try{
578         ps.close();
579         con.close();
580     }catch(Exception e) {
581         e.printStackTrace();
582         //throw new StudentAdmissionException("Database Value Insertion Failed",
e.getCause());
583     }
584 }
585 //code here
586
587     return recordsAdded;
588 }
589
590 public List<StudentAdmission> getAllStudentAdmissionDetails() throws
StudentAdmissionException {
591
592     List<StudentAdmission> stdAdmissions = new ArrayList<StudentAdmission>();
593
594     //code here
595     //Retrieval of all records from database
```

```
596     String query = "select * from UNO_ADMISSION";
597     try(Connection con = DBConnectionManager.getInstance().getConnection());
598     Statement st = con.createStatement();
599     ResultSet rs = st.executeQuery(query);
600     while(rs.next())
601     {
602         //storing retrieved records in object
603         StudentAdmission obj = new StudentAdmission();
604         obj.setAdmissionId(rs.getString(1));
605         obj.setStudentCode(rs.getString(2));
606         obj.setDateOfCounseling(new java.util.Date(rs.getDate(3).getTime()));
607         obj.setDepartmentName(rs.getString(4));
608         obj.setDateOfAdmission(new java.util.Date(rs.getDate(5).getTime()));
609         obj.setPreferCollegeHostel(rs.getString(6));
610         obj.setFirstGraduate(rs.getString(7));
611         obj.setManagerApproval(rs.getString(8));
612         obj.setAdmissionFee(rs.getDouble(9));
613         obj.setTuitionFee(rs.getDouble(10));
614         obj.setHostelFee(rs.getDouble(11));
615         obj.setTotalCollegeFee(rs.getDouble(12));
616         obj.setFinalStatusOfAdmission(rs.getString(13));
617         //adding StudentAdmission object into arraylist
618         stdAdmissions.add(obj);
619     }
620
621     }catch(Exception e)
622     {
623         e.printStackTrace();
624         //throw new StudentAdmissionException("Database Value Retrieval Failed",
e.getCause());
625     }
626     return stdAdmissions;
627
628 }
629 }
630
631 //StudentAdmissionException
632
633 package com.cts.unoadm.exception;
634
635 public class StudentAdmissionException extends Exception {
636
637     private static final long serialVersionUID = -1105431869622052445L;
638
639     /**
640      * @param message
641      * @param cause
642      */
643     public StudentAdmissionException(String message, Throwable cause) {
644         super(message, cause);
645     }
646 }
647
648 //StudentAdmissionService
649
650 package com.cts.unoadm.service;
651
```

```
652 import java.util.ArrayList;
653 import java.util.List;
654
655 import com.cts.unoadm.exception.StudentAdmissionException;
656 import com.cts.unoadm.vo.StudentAdmission;
657 import com.cts.unoadm.util.*;
658 import com.cts.unoadm.dao.*;
659 public class StudentAdmissionService {
660
661     /**
662      * @param empReimburseRecords
663      * @return List<StudentAdmission>
664      */
665     public static List<StudentAdmission> buildStudentAdmissionsList(List<String>
studentAdmissionRecords) {
666         List<StudentAdmission> studentAdmissionList = new ArrayList<StudentAdmission>();
667
668         //Code here
669         //storing each line into List of StudentAdmission objects
670         for(String e:studentAdmissionRecords) {
671             String res[] = e.split(",");
672             String admissionId = res[0];
673             String studentCode = res[1];
674             String dateOfCounseling = res[2];
675             String departmentName = res[3];
676             String dateOfAdmission = res[4];
677             String preferCollegeHostel = res[5];
678             String firstGraduate = res[6];
679             String managerApproval = res[7];
680             StudentAdmission obj = new StudentAdmission();
681             obj.setAdmissionId(admissionId);
682             obj.setStudentCode(studentCode);
683             //converting String to java.util.Date
684             obj.setDateOfCounseling(ApplicationUtil.convertStringToDate(dateOfCounseling));
685             obj.setDepartmentName(departmentName);
686             //converting String to java.util.Date
687             obj.setDateOfAdmission(ApplicationUtil.convertStringToDate(dateOfAdmission));
688             obj.setPreferCollegeHostel(preferCollegeHostel);
689             obj.setFirstGraduate(firstGraduate);
690             obj.setManagerApproval(managerApproval);
691             double[] studentAdmissionCosts = calculateTotalCollegeFee
(prefCollegeHostel,firstGraduate,departmentName);
692             obj.setAdmissionFee(studentAdmissionCosts[0]);
693             obj.setTuitionFee(studentAdmissionCosts[1]);
694             obj.setHostelFee(studentAdmissionCosts[2]);
695             obj.setTotalCollegeFee(studentAdmissionCosts[3]);
696             obj.setFinalStatusOfAdmission("AdmissionSuccessfull");
697
698             studentAdmissionList.add(obj);
699         }
700     }
701
702     return studentAdmissionList;
703 }
704
705
706     public boolean addStudentAdmissionDetails(String inputFeed) throws
```

```
    StudentAdmissionException {
707
708     //Code here
709     List<StudentAdmission> studentAdmissionList =
710     StudentAdmissionService.buildStudentAdmissionsList(ApplicationUtil.readFile(inputFeed));
711     StudentAdmissionDAO stdDao = new StudentAdmissionDAO();
712     return stdDao.addStudentAdmissionDetails(studentAdmissionList);
713 }
714
715     public static double[] calculateTotalCollegeFee(String preferCollegeHostel, String
716 firstGraduate, String departmentName) {
717
718     double[] studentAdmissionCosts = new double[4];
719
720     //Code here..
721     if("YES".equals(preferCollegeHostel))
722     {
723         studentAdmissionCosts[2]=75000;
724     }
725     else{
726         studentAdmissionCosts[2]=0;
727     }
728     if("EEE".equals(departmentName)) {
729
730         studentAdmissionCosts[0]=30000;
731         studentAdmissionCosts[1]=45000;
732     }
733     else if("ECE".equals(departmentName)) {
734
735         studentAdmissionCosts[0]=30000;
736         studentAdmissionCosts[1]=50000;
737     }
738     else if("CSE".equals(departmentName)) {
739
740         studentAdmissionCosts[0]=30000;
741         studentAdmissionCosts[1]=45000;
742     }
743     else if("MECH".equals(departmentName)) {
744
745         studentAdmissionCosts[0]=30000;
746         studentAdmissionCosts[1]=55000;
747     }
748     else if("CIVIL".equals(departmentName)) {
749
750         studentAdmissionCosts[0]=30000;
751         studentAdmissionCosts[1]=50000;
752     }
753     else if("IT".equals(departmentName)) {
754
755         studentAdmissionCosts[0]=30000;
756         studentAdmissionCosts[1]=45000;
757     }
758     //for first graduate discount is there
759     if("YES".equals(firstGraduate)) {
760         studentAdmissionCosts[3]=studentAdmissionCosts[0]+studentAdmissionCosts
761 [1]+studentAdmissionCosts[2]-20000;
762     }
763     else{
```

```
760         studentAdmissionCosts[3]=studentAdmissionCosts[0]+studentAdmissionCosts  
761         [1]+studentAdmissionCosts[2];  
762     }  
763     return studentAdmissionCosts;  
764 }  
765 public boolean searchStudentAdmission(String admissionId) throws StudentAdmissionException  
{  
766     boolean status = false;  
767  
768     //Code here..  
769     StudentAdmissionDAO stdDao = new StudentAdmissionDAO();  
770     List<StudentAdmission> stdAdmissions = stdDao.getAllStudentAdmissionDetails();  
771     for(StudentAdmission e:stdAdmissions) {  
772         if(e.getAdmissionId().equals(admissionId)) {  
773             status = true;  
774             System.out.println(e);  
775             break;  
776         }  
777     }  
778     return status;  
779 }  
780 }  
781 }
```