

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        System.out.println("Enter the amount to be withdrawn:");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();
}
}
```

StudentAdmissionDAO.java

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

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

import com.cts.unoadm.exception.StudentAdmissionException;
import com.cts.unoadm.vo.StudentAdmission;
import com.cts.unoadm.util.ApplicationUtil;
import com.cts.unoadm.util.DBConnectionManager;

public class StudentAdmissionDAO {

    @SuppressWarnings("finally")
    public boolean addStudentAdmissionDetails(List<StudentAdmission>
stdAdmissions) throws StudentAdmissionException {
        boolean recordsAdded = false;
        //code here
    }
}
```

```
try(Connection con = DBConnectionManager.getInstance().getConnection()) {  
  
    for(StudentAdmission stdAdmObj:stdAdmissions) {  
  
        String sql = "INSERT INTO students  
VALUES(?,?,?,?,?,?,?,?,?,?);";  
  
        PreparedStatement prepState = con.prepareStatement(sql);  
  
        prepState.setString(1, stdAdmObj.getAdmissionId());  
  
        prepState.setString(2, stdAdmObj.getStudentCode());  
  
        prepState.setDate(3,  
ApplicationUtil.convertUtilToSqlDate(stdAdmObj.getDateOfCounseling()));  
  
        prepState.setString(4, stdAdmObj.getDepartmentName());  
  
        prepState.setDate(5,  
ApplicationUtil.convertUtilToSqlDate(stdAdmObj.getDateOfAdmission()));  
  
        prepState.setString(6, stdAdmObj.getPreferCollegeHostel());  
  
        prepState.setString(7, stdAdmObj.getFirstGraduate());  
  
        prepState.setString(8, stdAdmObj.getManagerApproval());  
  
        prepState.setFloat(9, (float)stdAdmObj.getAdmissionFee());  
  
        prepState.setFloat(10, (float)stdAdmObj.getTuitionFee());  
  
        prepState.setDouble(11, (double)stdAdmObj.getHostelFee());  
  
        prepState.setFloat(12, (float)stdAdmObj.getTotalCollegeFee());  
  
        prepState.setString(13, stdAdmObj.getFinalStatusOfAdmission());  
  
        prepState.execute();  
  
    }  
  
    recordsAdded= true;
```

```
        } catch(Exception e) {  
            System.out.println(e.getMessage());  
  
            throw new StudentAdmissionException(e.getMessage(), e.getCause());  
        } finally {  
  
            return recordsAdded;  
        }  
    }  
  
    @SuppressWarnings("finally")  
    public List<StudentAdmission> getAllStudentAdmissionDetails() throws  
    StudentAdmissionException {  
  
        List<StudentAdmission> stdAdmissions = new ArrayList<StudentAdmission>();  
  
        //code here  
        try(Connection con = DBConnectionManager.getInstance().getConnection()) {  
            String sql = "SELECT * FROM students";  
            PreparedStatement prepState = con.prepareStatement(sql);  
            ResultSet resSet = prepState.executeQuery();  
            while(resSet.next()) {  
                String admissionId = resSet.getString(1);  
            }  
        } catch(Exception e) {  
            System.out.println(e.getMessage());  
            throw new StudentAdmissionException(e.getMessage(), e.getCause());  
        }  
    }  
}
```

```
String studentCode = resSet.getString(2);

        Date dateOfCounseling =
ApplicationUtil.convertStringToDate(resSet.getString(3));

        String departmentName = resSet.getString(4);

        Date dateOfAdmission =
ApplicationUtil.convertStringToDate(resSet.getString(5));

        String preferCollegeHostel = resSet.getString(6);

        String firstGraduate = resSet.getString(7);

        String managerApproval = resSet.getString(8);

        double admissionFee = resSet.getDouble(9);

        double tuitionFee = resSet.getDouble(10);

        double hostelFee = resSet.getDouble(11);

        double totalCollegeFee = resSet.getDouble(12);

        String finalStatusOfAdmission = resSet.getString(13);

        StudentAdmission stdAdmObj = new StudentAdmission()

                admissionId,
                studentCode,
                dateOfCounseling,
                departmentName,
                dateOfAdmission,
                preferCollegeHostel,
                firstGraduate,
                managerApproval,
                admissionFee,
                tuitionFee,
```

```
        hostelFee,  
        totalCollegeFee,  
        finalStatusOfAdmission  
    );  
  
    stdAdmissions.add(stdAdmObj);  
  
}  
  
resSet.close();  
  
} catch(SQLException e) {  
    throw new StudentAdmissionException(e.getMessage(), e.getCause());  
  
} finally {  
  
    return stdAdmissions;  
  
}  
  
}
```

StudentAdmissionException.java

```
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.java

```
package com.cts.unoadm.main;

//import java.io.BufferedReader;
//import java.io.File;
//import java.io.FileReader;
//import java.io.BufferedReader;
//import java.io.File;
//import java.io.FileNotFoundException;
//import java.io.FileReader;
import java.io.IOException;
import java.util.Scanner;

import com.cts.unoadm.skeletonvalidator.SkeletonValidator;
import com.cts.unoadm.service.StudentAdmissionService;
import com.cts.unoadm.exception.StudentAdmissionException;

public class MainApp {
```

```
public static void main(String[] args) throws IOException {  
    //Don't delete this code  
    //Skeletonvalidaton starts  
    new SkeletonValidator();  
    //Skeletonvalidation ends  
  
    //Write your code here..  
    @SuppressWarnings("resource")  
    Scanner sc = new Scanner(System.in);  
    StudentAdmissionService stdAdmService = new  
    StudentAdmissionService();  
    try {  
        //          File file = new File("inputFeed.txt");  
        //  
        //          BufferedReader br = new BufferedReader(new  
        FileReader(file));  
        //  
        //          String st;  
        //          while ((st = br.readLine()) != null)  
        //          System.out.println(st);  
        //          br.close();
```

```
if(stdAdmService.addStudentAdmissionDetails("inputFeed.txt"))

{

    System.out.println("Data has been inserted into
database");

} else {

    System.out.println("Database insertion failed");


}

System.out.print("Enter a admission Id to search status - ");

String admissionId = sc.nextLine();

if(stdAdmService.searchStudentAdmission(admissionId)) {

    System.out.println("Student admission found");

} else {

    System.out.println("Can't be found on database");

}

} catch(StudentAdmissionException e) {

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

}

}
```

```
}
```

StudentAdmissionService.java

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

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

import com.cts.unoadm.exception.StudentAdmissionException;
import com.cts.unoadm.vo.StudentAdmission;
import com.cts.unoadm.util.ApplicationUtil;
import com.cts.unoadm.dao.StudentAdmissionDAO;

public class StudentAdmissionService {

    /**
     * @return List<StudentAdmission>
     */
}
```

```
public static List<StudentAdmission>
buildStudentAdmissionsList(List<String> studentAdmissionRecords) {

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

    //Code here

    for(String line:studentAdmissionRecords) {

        String[] words = line.split(",");
        String admissionId = words[0].trim();
        String studentCode = words[1].trim();
        Date dateOfCounseling =
ApplicationUtil.convertStringToDate(words[2].trim());
        String departmentName = words[3].trim();
        Date dateOfAdmission =
ApplicationUtil.convertStringToDate(words[4].trim());
        String preferCollegeHostel = words[5].trim();
        String firstGraduate = words[6].trim();
        String managerApproval = words[7].trim();
        double[] fees = calculateTotalCollegeFee(preferCollegeHostel,
firstGraduate, departmentName);
        double admissionFee = fees[0];
        double tuitionFee = fees[1];
        double hostelFee = fees[2];
    }
}
```

```
        double totalCollegeFee = fees[3];  
  
        String finalStatusOfAdmission = "AdmissionSuccessfull";  
        StudentAdmission stdObj = new StudentAdmission(  
            admissionId,  
            studentCode,  
            dateOfCounseling,  
            departmentName,  
            dateOfAdmission,  
            preferCollegeHostel,  
            firstGraduate,  
            managerApproval,  
            admissionFee,  
            tuitionFee,  
            hostelFee,  
            totalCollegeFee,  
            finalStatusOfAdmission  
        );  
        studentAdmissionList.add(stdObj);  
    }  
    return studentAdmissionList;  
}
```

```
public boolean addStudentAdmissionDetails(String inputFeed) throws
StudentAdmissionException {

    //Code here

    List<String> parsedRecords = ApplicationUtil.readFile(inputFeed);

    List<StudentAdmission> studentAdmissionRecords =
StudentAdmissionService.buildStudentAdmissionsList(parsedRecords);

    StudentAdmissionDAO s=new StudentAdmissionDAO();

    return s.addStudentAdmissionDetails(studentAdmissionRecords);

}

//    return false;

//}
```

```
public static double[] calculateTotalCollegeFee(String preferCollegeHostel,
String firstGraduate, String departmentName) {

    double[] studentAdmissionCosts = new double[4];
```

```
//Code here..  
  
studentAdmissionCosts[0] = 30000d;  
  
studentAdmissionCosts[1] = 0d;  
  
studentAdmissionCosts[2] = 0d;  
  
studentAdmissionCosts[3] = 0d;  
  
if(departmentName.equalsIgnoreCase("EEE") ||  
departmentName.equalsIgnoreCase("CSE") ||  
departmentName.equalsIgnoreCase("IT")) {  
  
    studentAdmissionCosts[1] = 45000d;  
  
} else if(departmentName.equalsIgnoreCase("ECE") ||  
departmentName.equalsIgnoreCase("CIVIL")) {  
  
    studentAdmissionCosts[1] = 50000d;  
  
} else if(departmentName.equalsIgnoreCase("MECH")) {  
  
    studentAdmissionCosts[1] = 55000d;  
  
}  
  
if(preferCollegeHostel.equalsIgnoreCase("YES")) {  
  
    studentAdmissionCosts[2] = 75000d;  
  
}  
  
studentAdmissionCosts[3] = studentAdmissionCosts[0] +  
studentAdmissionCosts[1] + studentAdmissionCosts[2];
```

```
if(firstGraduate.equalsIgnoreCase("YES")) {  
    studentAdmissionCosts[3] -= 20000d;  
}  
  
return studentAdmissionCosts;  
}  
  
  
public boolean searchStudentAdmission(String admissionId) throws  
StudentAdmissionException {  
    boolean status = false;  
  
  
    //Code here..  
  
    List<StudentAdmission> fetchedAdmissions = new  
    StudentAdmissionDAO().getAllStudentAdmissionDetails();  
  
  
    for(StudentAdmission stdAdm:fetchedAdmissions) {  
        if(stdAdm.getAdmissionId().equalsIgnoreCase(admissionId)) {  
            status = true;  
            System.out.println(stdAdm.toString());  
        }  
    }  
}
```

```
        return status;  
    }  
}
```

SkeletonValidator.java

```
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.unoadm.util.DBConnectionManager");  
        validateClassName("com.cts.unoadm.util.ApplicationUtil");  
    }  
}
```

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

    validateClassName("com.cts.unoadm.dao.StudentAdmissionDAO");

    validateClassName("com.cts.unoadm.vo.StudentAdmission");

    validateClassName("com.cts.unoadm.exception.StudentAdmissionException");

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.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");

}

}

}
```

ApplocationUtil.java

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

import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.stream.Stream;
import java.util.concurrent.TimeUnit;
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..
    try(Stream<String> lines = Files.lines(Paths.get(fileName))) {
        lines.forEach((line)->{
            String[] words = line.split(",");
            if(ApplicationUtil.checkIfValidAdmission(
                ApplicationUtil.convertStringToDate(words[2].trim()),
                ApplicationUtil.convertStringToDate(words[4].trim()),
                words[7].trim()
            ))
            studentAdmissionList.add(line.trim());
        });
    } catch(IOException e) {
```

```
        throw new StudentAdmissionException(e.getMessage(),  
e.getCause());  
    }  
  
}
```

```
return studentAdmissionList;
```

```
}
```

```
/**
```

```
* @param util
```

```
*      Date
```

```
* @return sql Date
```

```
*/
```

```
public static java.sql.Date convertUtilToSqlDate(java.util.Date uDate) {
```

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

```
return new java.sql.Date(uDate.getTime());
```

```
//return sDate;  
}  
  
/**  
 * @param inDate  
 * @return Date  
 */  
public static Date convertStringToDate(String inDate) {  
  
    //Code here..  
  
    SimpleDateFormat sDf = new SimpleDateFormat("yyyy-MM-dd");  
  
    Date date = null;  
  
    try {  
        date = sDf.parse(inDate);  
    } catch(ParseException e) {  
        e.printStackTrace();  
    }  
    return date;//TODO change this return value  
}
```

```
//return new Date();//TODO change this return value
//}

public static boolean checkIfValidAdmission(Date dtOfCounseling, Date
dtOfAdmission, String manager) {

    boolean admissionValidity = false;

    //Code here..

    long counselingMillis = dtOfCounseling.getTime();
    long admissionMillis = dtOfAdmission.getTime();

    long days = TimeUnit.DAYS.convert(Math.abs(admissionMillis -
counselingMillis), TimeUnit.MILLISECONDS);

    if(days <= 10 && manager.equalsIgnoreCase("Approved")) {

        admissionValidity = true;
    }

    return admissionValidity;
}
```

DBConnectionManager.java

```
/*
 * 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));
            DBConnectionManager.connection =
                DriverManager.getConnection(props.getProperty(URL),
                props.getProperty(USER_NAME),
                props.getProperty(PASSWORD));
            //Class.forName(com.mysql.cj.jdbc.Driver);
        } catch (SQLException e) {
            throw new StudentAdmissionException("Error connecting to database");
        }
    }

    public static Connection getConnection() throws StudentAdmissionException {
        if (connection == null) {
            synchronized (DBConnectionManager.class) {
                if (connection == null) {
                    connection = new DBConnectionManager();
                }
            }
        }
        return connection;
    }
}
```

```

        //DBConnectionManager.connection =
DriverManager.getConnection(URL,USER_NAME,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 new DBConnectionManager();

    //return null;
}

/**
 * @throws StudentAdmissionException
 */
private void loadProperties() throws StudentAdmissionException {
    FileInputStream inputStream = null;
    try {
        inputStream = new FileInputStream(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.java

```

/*
 * 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 + "]";
    }

}

```

Database.properties

```

#IF NEEDED, YOU CAN MODIFY THIS PROPERTY FILE

#ENSURE YOU ARE NOT CHANGING THE NAME OF THE PROPERTY

#YOU CAN CHANGE THE VALUE OF THE PROPERTY

#LOAD THE DETAILS OF DRIVER CLASS, URL, USERNAME AND PASSWORD using this properties
file only.

#Do not hard code the values

```

```

drivername=com.mysql.cj.jdbc.Driver

url=jdbc:mysql://localhost:3306/uno_admission

username=root

password=

```

inputFeed.txt

A001,S001,2020-01-15,EEE,2020-01-25,YES,YES,Approved

A002,S002,2020-02-04,MECH,2020-02-12,N0,YES,Approved

A003,S003,2020-04-21,CSE,2020-05-27,YES,NO,Approved

A004,S004,2020-07-16,IT,2020-07-24,NO,NO,Approved

A005,S005,2020-08-10,ECE,2020-08-11,YES,YES,Approved

A006,S006,2020-09-01,EEE,2020-09-10,YES,NO,Pending

A007,S007,2020-10-19,CIVIL,2020-10-28,NO,YES,Approved

MySQL Queries ::

--

-- Table structure for table `students`

--

DROP TABLE IF EXISTS students;

/*!40101 SET @saved_cs_client = @@character_set_client */;

/*!40101 SET character_set_client = utf8 */;

CREATE TABLE students (

admission_id varchar(4) NOT NULL,

student_code varchar(4) NOT NULL,

date_of_counseling date DEFAULT NULL,

department_name varchar(15) NOT NULL,

date_of_admission date DEFAULT NULL,

prefer_college_hostel varchar(20) NOT NULL,

first_graduate varchar(20) NOT NULL,

manager_approval varchar(15) NOT NULL,

```
admission_fee float(11,2) NOT NULL,  
tuition_fee float(11,2) NOT NULL,  
hostel_fee float(11,2) NOT NULL,  
total_college_fee float(11,2) DEFAULT NULL,  
final_status_of_admission varchar(25) NOT NULL,  
PRIMARY KEY (admission_id),  
UNIQUE KEY student_code (student_code)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;  
/*!40101 SET character_set_client = @saved_cs_client */;  
  
select * from students;
```

1. AirVoice - Registration

Grade settings: Maximum grade: 100

Disable external file upload, paste and drop external content: Yes

Run: Yes **Evaluate:** Yes

Automatic grade: Yes **Maximum execution time:** 16 s

SmartBuy is a leading mobile shop in the town. After buying a product, the customer needs to provide a few personal details for the invoice to be generated.

You being their software consultant have been approached to develop software to retrieve the personal details of the customers, which will help them to generate the invoice faster.

Component Specification: Customer

Type(Class)	Attributes	Methods	Responsibilities
Customer	String customerName long contactNumber String emailId int age	Include the getters and setters method for all the attributes.	

In the **Main** class, create an object for the Customer class.

Get the details as shown in the sample input and assign the value for its attributes using the setters.

Display the details as shown in the sample output using the getters method.

All classes and methods should be public, Attributes should be private.

Note:

In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the rest of the text represents the output.

Ensure to follow the object oriented specifications provided in the question.

Ensure to provide the names for classes, attributes and methods as specified in the question.

Adhere to the code template, if provided.

Sample Input 1:

Enter the Name:

john

Enter the ContactNumber:

9874561230

Enter the EmailId:

john@gmail.com

Enter the Age:

32

Sample Output 1:

Name:john

ContactNumber:9874561230

EmailId:john@gmail.com

Age:32

[Automatic evaluation\[+\]](#)

Customer.java

```
1 public class Customer {  
2     private String customerName;  
3  
4     private long contactNumber;  
5  
6     private String emailId;  
7  
8     private int age;  
9  
10    public String getCustomerName() {  
11        return customerName;  
12    }  
13  
14    public void setCustomerName(String customerName) {  
15        this.customerName = customerName;  
16    }  
17  
18    public long getContactNumber() {  
19        return contactNumber;  
20    }  
21  
22    public void setContactNumber(long contactNumber) {  
23        this.contactNumber = contactNumber;  
24    }  
25  
26    public String getEmailId() {  
27        return emailId;  
28    }  
29  
30    public void setEmailId(String emailId) {  
31        this.emailId = emailId;  
32    }
```

```
33
34     public int getAge() {
35         return age;
36     }
37
38     public void setAge(int age){
39         this.age = age;
40     }
41
42
43
44 }
```

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
8         //Fill the code
9         Customer c=new Customer();
10        System.out.println("Enter the Name:");
11        String name=(sc.nextLine());
12        System.out.println("Enter the ContactNumber:");
13        long no=sc.nextLong();
14        sc.nextLine();
15        System.out.println("Enter the EmailId:");
16        String mail=sc.nextLine();
17
18        System.out.println("Enter the Age:");
19        int age=sc.nextInt();
20        c.setCustomerName(name);
21        c.setContactNumber(no);
22        c.setEmailId(mail);
23        c.setAge(age);
24        System.out.println("Name:"+c.getCustomerName());
25        System.out.println("ContactNumber:"+c.getContactNumber());
26        System.out.println("EmailId:"+c.getEmailId());
27        System.out.println("Age:"+c.getAge());
28
29
30
31    }
32
33 }
```

2. Grade

Reviewed on Friday, 10 December 2021, 6:14 PM by Automatic grade

Grade 100 / 100

Assessment report

[\[+\]Grading and Feedback](#)

3. ZeeZee bank

Grade settings: Maximum grade: 100

Disable external file upload, paste and drop external content: Yes

Run: Yes **Evaluate:** Yes

Automatic grade: Yes **Maximum execution time:** 16 s

ZeeZee is a leading private sector bank. In the last Annual meeting, they decided to give their customer a 24/7 banking facility. As an initiative, the bank outlined to develop a stand-alone device that would offer deposit and withdrawal of money to the customers anytime.

You being their software consultant have been approached to develop software to implement the functionality of deposit and withdrawal anytime.

Component Specification: Account

Type(Class)	Attributes	Methods	Responsibilities
Account	long accountNumber double balanceAmount	Include the getters and setters method for all the attributes. Include a parametrized constructor of two arguments in the order – accountNumber,balanceAmount to initialize the values for the account object	

Requirement 1: Being able to deposit money into an account anytime

As per this requirement, the customer should be able to deposit money into his account at any time and the deposited amount should reflect in his account balance.

Component Specification: Account

Component Name	Type(Class)	Methods	Responsibilities
Deposit amount to an account	Account	public void deposit(double depositAmt)	This method takes the amount to be deposited as an argument This method should perform the deposit, by adding the deposited amount to the balanceAmount

Requirement 2: Being able to withdraw money from the account anytime

As per this requirement, the customer should be able to withdraw money from his account anytime he wants. The amount to be withdrawn should be less than or equal to the balance in the account. After the withdrawal, the account should reflect the balance amount

Component Specification: Account

Component Name	Type(Class)	Methods	Responsibilities
Withdraw amount from an account	Account	public boolean withdraw(double withdrawAmt)	This method should take the amount to be withdrawn as an argument. This method should check the balanceAmount and deduct the withdraw amount from the balanceAmount and return true. If there is insufficient balance then return false.

In the **Main** class, Get the details as shown in the sample input.

Create an object for the Account class and invoke the deposit method to deposit the amount and withdraw method to withdraw the amount from the account.

All classes and methods should be public, Attributes should be private.

Note:

Balance amount should be displayed corrected to 2 decimal places.

Order of the transactions to be performed (Display,Deposit,Withdraw).

If the balance amount is insufficient then display the message as shown in the Sample Input / Output.

In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the rest of the text represents the output.

Ensure to follow the object-oriented specifications provided in the question.

Ensure to provide the names for classes, attributes, and methods as specified in the question.

Adhere to the code template, if provided.

Sample Input/Output 1:

Enter the account number:

1234567890

Enter the available amount in the account:

15000

Enter the amount to be deposited:

1500

Available balance is:16500.00

Enter the amount to be withdrawn:

500

Available balance is:16000.00

Sample Input/Output 2:

Enter the account number:

1234567890

Enter the available amount in the account:

15000

Enter the amount to be deposited:

1500

Available balance is:16500.00

Enter the amount to be withdrawn:

18500

Insufficient balance

Available balance is:16500.00

Automatic evaluation [\[+\]](#)

Main.java

```
1 import java.util.Scanner;
2 import java.text.DecimalFormat;
3
4 public class Main{
5
6     public static void main (String[] args) {
7         Scanner sc=new Scanner(System.in);
8         DecimalFormat decimalFormat=new DecimalFormat("0.00");
9         System.out.println("Enter the account number:");
```

```

10    long accountNumber= sc.nextLong();
11    System.out.println("Enter the available amount in the account:");
12    double balanceAmount= sc.nextDouble();
13    Account account=new Account(accountNumber,balanceAmount);
14    System.out.println("Enter the amount to be deposited:");
15    double depositAmount=sc.nextDouble();
16    account.deposit(depositAmount);
17    double availableBalance=account.getBalanceAmount();
18    System.out.println("Available balance is:"+decimalFormat.format(availableBalance));
19    System.out.println("Enter the amount to be withdrawn:");
20    double withdrawAmount= sc.nextDouble();
21    boolean isWithdrawn=account.withdraw(withdrawAmount);
22    availableBalance=account.getBalanceAmount();
23    if(isWithdrawn){
24        System.out.println("Insufficient balance");
25    }
26    System.out.println("Available balance is:"+decimalFormat.format(availableBalance));
27
28    //Fill the code
29}
30}

```

Account.java

```

1
2 public class Account {
3     private long accountNumber;
4     private double balanceAmount;
5     public Account(long accountNumber,double balanceAmount){
6         this.accountNumber=accountNumber;
7         this.balanceAmount=balanceAmount;
8     }
9     public long getAccountNumber(){
10        return accountNumber;
11    }
12    public void setAccountNumber(long accountNumber){
13        this.accountNumber=accountNumber;
14    }
15    public double getBalanceAmount(){
16        return balanceAmount;
17    }
18    public void setBalanceAmount(double balanceAmount){
19        this.balanceAmount=balanceAmount;
20    }
21    public void deposit(double depositAmount){
22        balanceAmount+=depositAmount;
23    }
24    public boolean withdraw(double withdrawAmount){
25        if(withdrawAmount<=balanceAmount){
26            balanceAmount-=withdrawAmount;
27            return true;
28        }
29        return false;
30    }
31}

```

Grade

Reviewed on Thursday, 27 May 2021, 3:28 AM by Automatic grade

Grade 100 / 100

Assessment report

[\[+\]Grading and Feedback](#)

3.Call Details

Grade settings: Maximum grade: 100

Disable external file upload, paste and drop external content: Yes

Run: Yes **Evaluate:** Yes

Automatic grade: Yes **Maximum execution time:** 16 s

AirCarrier is a leading mobile network provider. They maintain a record of all the calls made by their postpaid customers. The details are stored in a particular format

[callId:calledNumber:noOfMinutes] . At the end of every month, the network provider wants to extract the information from the file and populate it to the Call object for calculating the bill.

You being their software consultant have been approached to develop software to implement the functionality of extracting the data from the given format.

Component Specification: Call

Type(Class)	Attributes	Methods	Responsibilities
Call	int callId long calledNumber float duration	Include the getters and setters method for all the attributes.	

Requirement 1: Extracting the data from the callDetails

This requirement is responsible for extracting the customer's callId, calledNumber and duration from the callDetails. After the extraction set the callId, calledNumber and duration to the call object.

Component Specification: Call

Component Name	Type(Class)	Methods	Responsibilities
Extraction from file	Call	public void parseData(String callDetails)	This method takes the callDetails as an argument This method should perform the extraction process, to set the callId, calledNumber and duration of the call object.

In the **Main** class, Get the details as shown in the sample input.

Create an object for the Call and invoke the **parseData** method to set the callId, calledNumber and duration for each customer.

Invoke the corresponding getters to display the call details as shown in the Sample Output

All classes and methods should be public, Attributes should be private.

Note:

In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the rest of the text represents the output.

Ensure to follow the object-oriented specifications provided in the question.

Ensure to provide the names for classes, attributes and methods as specified in the question.

Adhere to the code template, if provided.

Sample Input 1:

Enter the call details:

102:6547891230:2.15

Sample Output 1:

Call id:102

Called number:6547891230

Duration:2.15

Automatic evaluation[+]

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         System.out.println("Enter the call details:");
8         String a=sc.nextLine();
9         Call obj=new Call();
10        obj.parseData(a);
11        System.out.println("Call id:"+obj.getCallId());
12        System.out.println("Called number:"+obj.getCalledNumber());
13        System.out.println("Duration:"+obj.getDuration());
14        //Fill the code
15
16    }
17 }
```

Call.java

```
1  public class Call {  
2      private int callId;  
3      private long calledNumber;  
4      private float duration;  
5      public Call(){  
6      }  
7      public int getCallId(){  
8          return callId;  
9      }  
10     public long getCalledNumber(){  
11         return calledNumber;  
12     }  
13     public float getDuration(){  
14         return duration;  
15     }  
16     public void setCallId(int callId){  
17         this.callId=callId;  
18     }  
19     public void setCalledNumber(long calledNumber){  
20         this.calledNumber=calledNumber;  
21     }  
22     public void setDuration(float duration){  
23         this.duration=duration;  
24     }  
25     public void parseData(String calld){  
26         callId=Integer.parseInt(calld.split(":")[0]);  
27         setCallId(callId);  
28         calledNumber=Long.parseLong(calld.split(":")[1]);  
29         setCalledNumber(calledNumber);  
30         duration=Float.parseFloat(calld.split(":")[2]);  
31         setDuration(duration);  
32     }  
33 }  
34 }
```

Grade

Reviewed on Tuesday, 4 May 2021, 4:58 AM by Automatic grade

Grade 100 / 100

Assessment report

[\[+\]Grading and Feedback](#)

4. Pair of Two digits

Grade settings: Maximum grade: 100

Disable external file upload, paste and drop external content: Yes

Run: Yes **Evaluate:** Yes

Automatic grade: Yes **Maximum execution time:** 32 s

Jerold teacher assigned a task to him. The task is to find the pair of two-digit numbers.

The pair is found by checking whether the product of the numbers is same as the product of the reversed numbers. If it is same, then print "Correct pair found". If not print, "Correct pair not found".

Write a Java program to find the correct pair of two-digit numbers.

Assume both the inputs are 2-digit values.

Note:

In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the remaining text represents the output.

Adhere to the code template, if provided.

Hint: [13*62=31*26]

Sample Input 1:

13

62

Sample Output 1:

13 and 62 are correct pair

Sample Input 2:

10

56

Sample Output 2:

10 and 56 are not correct pair

Automatic evaluation[+]

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
8         int a=sc.nextInt();
9         int b=sc.nextInt();
10        if(a>99||a<10||b>99||b<10){
11            System.out.println("No");
12
13        }
14        Main obj=new Main();
15        int ra=obj.rvs(a);
16        int rb=obj.rvs(b);
17        if(a*b==ra*rb){
18            System.out.println(a+" and "+b+" are correct pair");
19        }
}
```

```
20     else{
21         System.out.println(a+" and "+b+" are not correct pair");
22     }
23 }
24 int rvs(int num){
25     int r,rnum=0;
26     while(num>0)
27     {
28         r=num%10;
29         rnum=rnum*10+r;
30         num/=10;
31     }
32     return(rnum);
33 }
34 }
35
36
37
38
39
40
```

Grade

Reviewed on Thursday, 27 May 2021, 3:38 AM by Automatic grade

Grade 100 / 100

Assessment report

TEST CASE PASSED

[\[+\]Grading and Feedback](#)

-----End-----

Group-2

1. Find MemberShip Category Count

Grade settings: Maximum grade: 100

Disable external file upload, paste and drop external content: Yes

Run: Yes **Evaluate:** Yes

Automatic grade: Yes **Maximum execution time:** 32 s

Find Membership Category Count

ZEE Shopping mall wanted to know how many Members are available for each of their Membership category. The member ship category is of three types (Gold, Silver and Platinum).

You being their software consultant have been approached by them to develop an application which can be used for managing their business. You need to implement a java program using thread to find out the count of members in each membership category. Membership details should be obtained from the user in the console.

Component Specification: Member (Model Class)

Type(Class)	Attributes	Methods	Responsibilities
Member	String memberId String memberName String category	Include getters and setter method for all the attributes. Include a three argument constructor in the given order – memberId, memberName and category.	Set the values for all the attributes via constructor.

Note: The class and methods should be declared as public and all the attributes should be declared as private.

Requirement 1: Count the number of members

Count the number of members available in the memberList based on the membership category to be searched and set the value to count attribute.

Component Specification: ZEEShop (Thread Class)

Component Name	Type(Class)	Attributes	Methods	Responsibilities
	ZEEShop	String memberCategory int count List<Member>memberList	Include getters and setter method for all the attributes.	Set the values for all the attributes via constructor.

			Include a two argument constructor with arguments – memberCategory and memberList.	
Count the number of members	ZEEShop		void run()	Count the number of members based on the Membership category and set the value to the countattribute.

Note: The class and methods should be declared as public and all the attributes should be declared as private.

Create a class called **Main** with the main method and get the inputs like **number of members**, **member details**, **number of times Membership category needs to be searched** and **Membership category to be searched** from the user.

The member details will be in the form of String in the following format **memberId:memberName:category**.

Parse the member details and set the values for all attributes in **Member** class using **constructor**.

Invoke the ZEEShop thread class for each memberCategory and count the number of members in that category and display the count as shown in the sample input and output.

Assumption: The **memberCategory** is **case –sensitive** and will be of only three values – **Platinum or Gold or Silver**.

Note:

In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the remaining text represents the output.

Ensure to follow object oriented specifications provided in the question description.
Ensure to provide the names for classes, attributes and methods as specified in the question description.

Adhere to the code template, if provided.

Sample Input / Output 1:

Enter the number of Members:

5

Enter the Member Details:

101:Tom:Gold

Enter the Member Details:

102:Sam:Gold

Enter the Member Details:

103:John:Silver

Enter the Member Details:

104:Rose:Platinum

Enter the Member Details:

105:Tint:Silver

Enter the number of times Membership category needs to be searched:

4

Enter the Category

Gold

Enter the Category

Silver

Enter the Category

Platinum

Enter the Category

Gold

Gold:2

Silver:2

Platinum:1

Gold:2

Automatic evaluation [[+](#)]

Member.java

```
1  
2 public class Member {  
3
```

```

4   private String memberId;
5   private String memberName;
6   private String category;
7
8   public String getMemberId() {
9       return memberId;
10  }
11  public void setMemberId(String memberId) {
12      this.memberId = memberId;
13  }
14  public String getMemberName() {
15      return memberName;
16  }
17  public void setMemberName(String memberName) {
18      this.memberName = memberName;
19  }
20  public String getCategory() {
21      return category;
22  }
23  public void setCategory(String category) {
24      this.category = category;
25  }
26
27  public Member(String memberId, String memberName, String category) {
28      super();
29      this.memberId = memberId;
30      this.memberName = memberName;
31      this.category = category;
32  }
33
34
35 }
```

Main.java

```

1 import java.util.*;
2 public class Main {
3
4     public static void main(String args[]){
5         // Fill the code here
6         List<Member> memberList = new ArrayList<Member>();
7         Scanner scan = new Scanner(System.in);
8         System.out.println("Enter the no of Members");
9         int memberCount = scan.nextInt();
10        String templp;
11        while(memberCount>0){
12            System.out.println("Enter the member details");
13            templp = scan.next();
14            String tempArr[] = templp.split(":");
15            memberList.add(new Member(tempArr[0],tempArr[1],tempArr[2]));
16            memberCount--;
17        }
18        System.out.println("Enter the number of times Membership category needs to be searched");
19        int noOfTimes = scan.nextInt();
20        String[] tempArr = new String[noOfTimes];
21        for(int index=0;index<noOfTimes;index++){
22            System.out.println("Enter the category");
23            tempArr[index] = scan.next();
24        }
25        int countArr[] = new int [noOfTimes];
26        for(int i=0; i<noOfTimes;i++){
27            ZEEShop thread = new ZEEShop(tempArr[i],memberList);
28            thread.run();
29            /*try{
30                thread.join();
31            }catch(InterruptedException e){
32            }*/
33    }
```

```

34     countArr[i] = thread.getCount();
35 }
36 for(int i=0;i<noOfTimes;i++){
37     System.out.println(tempArr[i]+ ":"+countArr[i]);
38 }
39 scan.close();
40 /*List<ZEEShop> zList = new ArrayList<ZEEShop>()
41 for(int i = 0;i<count;i++){
42     ZEEShop zs = new ZEEShop(category , memList);
43     zList.add(zs);
44 }
45 for(ZEEShop z: zeelist){
46     z.start();
47     try{
48         z.join();
49     }catch(Exception e){
50         e.printStackTrace();
51     }
52 }*/
53 }
54 }
55

```

ZEEShop.java

```

1 import java.util.*;
2 public class ZEEShop extends Thread {
3     // Fill the code here
4     private String memberCategory;
5     private int count;
6     private List<Member> memberList;
7     public ZEEShop(String memberCategory, List memberList){
8         super();
9         this.memberCategory = memberCategory;
10        this.memberList = memberList;
11    }
12    public int getCount(){
13        return count;
14    }
15    public String getMemberCategory(){
16        return memberCategory;
17    }
18    public List<Member> getMemberList(){
19        return memberList;
20    }
21    public void setMemberCategory(String memberCategory){
22        this.memberCategory = memberCategory;
23    }
24    public void setMemberList(List<Member> memberList){
25        this.memberList = memberList;
26    }
27    public void setCount(int count){
28        this.count = count;
29    }
30    public void run(){
31
32        synchronized(this)
33        {
34            for(Member m : memberList){
35                if(m.getCategory().equals(memberCategory))
36                    count++;
37            }
38
39        }
40    }
41 }
42

```

Grade

Reviewed on Friday, 7 January 2022, 7:25 PM by Automatic grade

Grade 100 / 100

Assessment report

[\[+\]Grading and Feedback](#)

2. Grade Calculation

Grade settings: Maximum grade: 100

Disable external file upload, paste and drop external content: Yes

Run: Yes **Evaluate:** Yes

Automatic grade: Yes **Maximum execution time:** 32 s

Grade Calculation

Rita is working as a science teacher in an International school. She is the Class Teacher of class V and was busy in calculating the grade for each student in her class, based on his/her total marks obtained in SA1 assessment.

Since she found it very difficult to calculate the grade, she approached you to develop an application which can be used for completing her task faster. You need to implement a java program using thread to calculate the grade for each student. Student details should be obtained from the user in the console.

Requirement 1: Calculate the grade for each student.

Calculate the grade based on total marks (sum of all marks) as shown below obtained by each student and set the same in result attribute for respective student.

Total Marks	Grade
400 to 500	A
300 to 399	B
200 to 299	C
Less than 200	E

Assumption: Each student will have only five subjects and marks of each subject will be greater than or equal to 0 and lesser than or equal to 100. Hence the maximum Total marks obtained by each student will be 500. And the minimum Total marks obtained by each student will be 0.

Component Specification: GradeCalculator (Thread Class)

Component Name	Type(Class)	Attributes	Methods	Responsibilities
	GradeCalculator	String studName	Include getters and setter method for all	Set the values for all the attributes via constructor.

		char result int[] marks	the attributes. Include a two argument constructor in the given order – studName and marks.	
calculate the grade for each student	GradeCalculator		public void run()	Calculate the grade based on total marks and set the same to result attribute.

Note: The class and methods should be declared as public and all the attributes should be declared as private.

Create a class called **Main** with the main method and get the inputs like **number of threads** and **Student details** from the user.

The student details will be in the form of String in the following format **studName:mark1:mark2:mark3:mark4:mark5**.

Parse the student details and set the values of studName and marks attributes in **GradeCalculator** thread class using **constructor**.

Invoke the **GradeCalculator** thread class to calculate the grade based on total marks and set the same to result attribute.

Display the Student name and Grade obtained by each student as shown in the sample input and output.

Note:

In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the remaining text represents the output.

Ensure to follow object oriented specifications provided in the question description.
Ensure to provide the names for classes, attributes and methods as specified in the question description.

Adhere to the code template, if provided.

Sample Input / Output 1:

Enter the number of Threads:

4

Enter the String:

Jeba:100:80:90:40:55

Enter the String

David:10:8:9:40:5

Enter the String

Adam:90:80:90:50:75

Enter the String

Rohit:99:99:99:99:99

Jeba:B

David:E

Adam:B

Rohit:A

Automatic evaluation[+]

Main.java

```
1 import java.util.Scanner;
2 import java.io.BufferedReader;
3 import java.io.InputStreamReader;
4 public class Main {
5     public static void main(String[] args) throws Exception {
6         BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
7         System.out.println("Enter the number of Threads");
8         int th=Integer.parseInt(br.readLine());
9         GradeCalculator obj=null;
10        String str="";
11        String[] details=new String[th];
12        for(int i=0;i<th;i++)
13        {
14            System.out.println("Enter the String");
15            str=br.readLine();
16            details[i]=str;
17        }
18        for(int i=0;i<th;i++)
19        {
20            String sp[]=details[i].split(":");
21            int k=0;
22            int arr[]=new int[sp.length];
23            for(int j=1;j<sp.length;j++)
24                arr[k++]=Integer.parseInt(sp[j]);
25            obj=new GradeCalculator(sp[0],arr);
26            obj.start();
27            try{
28                Thread.sleep(1000);
29            }
30            catch(Exception e)
31            {
32                System.out.println(e);
33            }
34        }
35        //Fill your code here
36
37    }
38}
```

```

39 }
GradeCalculator.java
1
2 public class GradeCalculator extends Thread{
3   private String studName;
4   private char result;
5   private int[] marks;
6   public String getStudName()
7 {
8   return studName;
9 }
10 public void setStudName()
11 {
12   this.studName=studName;
13 }
14 public char getResult()
15 {
16   return result;
17 }
18 public void setResult(char result)
19 {
20   this.result=result;
21 }
22 public int[] getMarks()
23 {
24   return marks;
25 }
26 public void setMarks(int[] marks)
27 {
28   this.marks=marks;
29 }
30 public GradeCalculator(String studName,int[] marks)
31 {
32   this.studName=studName;
33   this.marks=marks;
34 }
35 public void run()
36 {
37   int sum=0;
38   int[] score=getMarks();
39   for(int i=0;i<score.length;i++)
40     sum=sum+score[i];
41   if(400<=sum)&&(sum<=500)
42     System.out.println(getStudName()+":"+'A');
43   if(300<=sum)&&(sum<=399)
44     System.out.println(getStudName()+":"+'B');
45   if(200<=sum)&&(sum<=299)
46     System.out.println(getStudName()+":"+'C');
47   if(sum<200)
48     System.out.println(getStudName()+":"+'E');
49 }
50 }

```

Grade

Reviewed on Friday, 7 January 2022, 7:24 PM by Automatic grade

Grade 100 / 100

Assessment report

[\[+\]Grading and Feedback](#)

3. Query Data Set

Grade settings: Maximum grade: 100
Disable external file upload, paste and drop external content: Yes
Run: Yes **Evaluate:** Yes
Automatic grade: Yes **Maximum execution time:** 32 s

Query Data Set

Jackson is pursuing his Bachelor's degree in TekSpec University, Alaska. His professor has given him a weekend assignment to develop an application in java using inner class concept.

You being his best friend, help him in completing his weekend assignment. You need to implement a java program using inner class concept to display the details available in primaryDataSet, secondaryDataSet and Query.

Requirement 1: Display the details in primaryDataSet, secondaryDataSet and Query

Component Specification: Query (Model Class)

Component Name	Type(Class)	Attributes	Methods	Responsibilities
	Query	String queryId String queryCategory DataSet primaryDataSet DataSet secondaryDataSet	Include getters and setter method for all the attributes.	
	DataSet	String theatreId String theatreName String location int noOfScreen double ticketCost	Include getters and setter method for all the attributes.	The DataSet class must be available only to Query class. So the DataSet class should come as a Inner class in Query class.
Display the details	Query	toString()		Override the toString method to produce the output as specified in the sample output

Note: The class and methods should be declared as public and all the attributes should be declared as private.

Create a class called **TestApplication** with the main method and get the inputs for primary data set and secondary data set like **theatreId**, **theatreName**, **location**, **noOfScreen** and **ticketCost**, and details of Query like **queryId** and **queryCategory** from the user.

Display the details of primary data set, secondary data set and Query as shown in the sample input and output.

Note:

In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the remaining text represents the output.

Ensure to follow object oriented specifications provided in the question description.

Ensure to provide the names for classes, attributes and methods as specified in the question description.

Adhere to the code template, if provided.

Sample Input / Output 1:

Enter the Details for primary data set

Enter the theatre id

PNR6001

Enter the theatre name

KV cinemas

Enter the location

Chennai

Enter the no of screens

8

Enter the ticket cost

120

Enter the Details for secondary data set

Enter the theatre id

RNV5001

Enter the theatre name

Inoxe

Enter the location

Bangalore

Enter the no of screens

5

Enter the ticket cost

150

Enter the query id

Q510

Enter the query category

DML

Primary data set

Theatre id : PNR6001

Theatre name : KV cinemas

Location : Chennai

No of Screen : 8

Ticket Cost : 120

Secondary data set

Theatre id : RNV5001

Theatre name : Inoxe

Location : Bangalore

No of Screen : 5

Ticket Cost : 150

Query id : Q510

Query category : DML

Automatic evaluation [\[+\]](#)

Query.java

```
1  
2 //Write the required business logic as expected in the question description  
3  
4 public class Query {  
5     private String queryId;  
6     private String queryCategory;  
7     private DataSet primaryDataSet;  
8     private DataSet secondaryDataSet;  
9 }
```

```

10  @Override
11  public String toString()
12  {
13      String g="";
14      g+=("Primary data set"+\n);
15      g+=("Theatre id :" +primaryDataSet.getTheatreId()\n);
16      g+=("Theatre name :" +primaryDataSet.getTheatreName()\n);
17      g+=("Location :" +primaryDataSet.getLocation()\n);
18      g+=("No of Screen :" +primaryDataSet.getNoOfScreen()\n);
19      g+=("Ticket Cost :" +primaryDataSet.getTicketCost()\n);
20
21      g+=("Secondary data set"+\n);
22      g+=("Theatre id :" +secondaryDataSet.getTheatreId()\n);
23      g+=("Theatre name :" +secondaryDataSet.getTheatreName()\n);
24      g+=("Location :" +secondaryDataSet.getLocation()\n);
25      g+=("No of Screen :" +secondaryDataSet.getNoOfScreen()\n);
26      g+=("Ticket Cost :" +secondaryDataSet.getTicketCost()\n);
27      g+=("Query id : "+queryId+\n);
28      g+=("Query category : "+queryCategory+\n);
29
30      return g;
31  }
32  public class DataSet{
33      private String theatreId;
34      private String theatreName;
35      private String location;
36      private int noOfScreen;
37      private double ticketCost;
38
39      public double getTicketCost()
40      {
41          return ticketCost;
42      }
43      public void setTicketCost(double a)
44      {
45          ticketCost=a;
46      }
47
48      public int getNoOfScreen()
49      {
50          return noOfScreen;
51      }
52      public void setNoOfScreen(int a)
53      {
54          noOfScreen=a;
55      }
56      public String getLocation()
57      {
58          return location;
59      }
60      public void setLocation(String a)
61      {
62          location=a;
63      }
64      public String getTheatreName ()
65      {
66          return theatreName;
67      }
68      public void setTheatreName(String a)
69      {
70          theatreName=a;
71      }
72
73      public String getTheatreId()
74      {
75          return theatreId;
76      }

```

```

77     public void setTheatreId(String a)
78     {
79         theatreId=a;
80     }
81 }
82 public void setSecondaryDataSet(DataSet pD)
83 {
84     this.secondaryDataSet=pD;
85 }
86 public DataSet getSecondaryDataSet()
87 {
88     return this.secondaryDataSet;
89 }
90 public void setPrimaryDataSet(DataSet pD)
91 {
92     this.primaryDataSet=pD;
93 }
94 public DataSet getPrimaryDataSet()
95 {
96     return this.primaryDataSet;
97 }
98 public void setQueryId (String queryId)
99 {
100    this.queryId=queryId;
101 }
102 public void setQueryCategory(String queryCategory)
103 {
104    this.queryCategory=queryCategory;
105 }
106 public String getQueryId()
107 {
108     return this.queryId;
109 }
110 public String getQueryCategory()
111 {
112     return this.queryCategory;
113 }
114
115 }

```

TestApplication.java

```

1 import java.util.*;
2 public class TestApplication {
3     //Write the required business logic as expected in the question description
4     public static void main (String[] args) {
5         Scanner sc= new Scanner (System.in);
6         Query q= new Query();
7         Query.DataSet pd= q.new DataSet();
8         Query.DataSet sd= q.new DataSet();
9         System.out.println("Enter the Details for primary data set");
10        System.out.println("Enter the theatre id");
11        pd.setTheatreId(sc.nextLine());
12        System.out.println("Enter the theatre name");
13        pd.setTheatreName(sc.nextLine());
14        System.out.println("Enter the location");
15        pd.setLocation(sc.nextLine());
16        System.out.println("Enter the no of screens");
17        pd.setNoOfScreen(sc.nextInt());
18        System.out.println("Enter the ticket cost");
19        pd.setTicketCost(sc.nextDouble());
20        System.out.println("Enter the Details for secondary data set");
21        System.out.println("Enter the theatre id");
22
23        String id2=sc.next();
24        //System.out.println(id2);
25        sd.setTheatreId(id2);
26        System.out.println("Enter the theatre name");

```

```

27     sc.nextLine();
28     sd.setTheatreName(sc.nextLine());
29     System.out.println("Enter the location");
30     String gll=sc.nextLine();
31     sd.setLocation(gll);
32     System.out.println("Enter the no of screens");
33
34 //System.out.println(gll);
35 //String pp=sc.nextLine();
36 //System.out.println(pp);
37
38     sd.setNoOfScreen(sc.nextInt());
39     System.out.println("Enter the ticket cost");
40     sd.setTicketCost(sc.nextDouble());
41     sc.nextLine();
42     System.out.println("Enter the query id");
43     q.setQueryId(sc.nextLine());
44     System.out.println("Enter the query category");
45     q.setQueryCategory(sc.nextLine());
46
47     q.setSecondaryDataSet(sd);
48     q.setPrimaryDataSet(pd);
49     System.out.println(q.toString());
50 }
51 }
```

Grade

Reviewed on Friday, 17 December 2021, 6:59 PM by Automatic grade

Grade 100 / 100

Assessment report

[\[+\]Grading and Feedback](#)

4. Retrieve Flights Based on Source and Destination

Grade settings: Maximum grade: 100

Disable external file upload, paste and drop external content: Yes

Run: Yes **Evaluate:** Yes

Automatic grade: Yes **Maximum execution time:** 32 s

Retrieve Flights Based on Source and Destination

Zaro Flight System wants to automate the process in their organization. The flight details are available in the database, the customer should have the facility to view flights which are from a particular source to destination.

You being their software consultant have been approached by them to develop an application which can be used for managing their business. You need to implement a java program to view all the flight based on source and destination.

Component Specification: Flight (Model Class)

Type	Class	Attribute	Methods	Responsibilitie
------	-------	-----------	---------	-----------------

Attributes		Operations
Flight	int flightId String source String destination int noOfSeats double flightFare	Include getters and setter method for all the attributes. Include a five argument constructor in the given order – flightId, source, destination, noOfSeats and flightFare.

Note: The class and methods should be declared as public and all the attributes should be declared as private.

Requirement 1: Retrieve all the flights with the given source and destination

The customer should have the facility to view flights which are from a particular source to destination. Hence the system should fetch all the flight details for the given source and destination from the database. Those flight details should be added to a ArrayList and return the same.

Component Specification: FlightManagementSystem

Component Name	Type(Class)	Attributes	Methods	Responsibilities
Retrieve all the flights with the given source and destination	FlightManagement System		public ArrayList<Flight> viewFlightBySourceDestination(String source, String destination)	This method should accept a Source and a destination as parameter and retrieve all the flights with the given source and destination from the database. Return these details as ArrayList<Flight>.

Note: The class and methods should be declared as public and all the attributes should be declared as private.

The **flight** table is already created at the backend. The structure of flight table is:

Column Name	Datatype
flightId	int
source	varchar2(30)
destination	varchar2(30)
noofseats	int
flightfare	number(8,2)

Sample records available in **flight** table are:

Flightid	Source	Destination	Noofseats	Flightfare
18221	Malaysia	Singapore	50	5000
18222	Dubai	Kochi	25	50000
18223	Malaysia	Singapore	150	6000
18224	Malaysia	Singapore	100	7000

To connect to the database you are provided with **database.properties** file and **DB.java** file. (**Do not change any values in database.properties file**)

Create a class called **Main** with the main method and get the inputs like **source** and **destination** from the user.

Display the details of flight such as flightId, noofseats and flightfare for all the flights returned as `ArrayList<Flight>` from the method **viewFlightBySourceDestination** in **FlightManagementSystem** class.

If no flight is available in the list, the output should be “**No flights available for the given source and destination**”.

Note:

In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user and the remaining text represents the output.

Ensure to follow object oriented specifications provided in the question description.
Ensure to provide the names for classes, attributes and methods as specified in the question description.

Adhere to the code template, if provided.

Sample Input / Output 1:

Enter the source

Malaysia

Enter the destination

Singapore

Flightid Noofseats Flightfare

18221 50 5000.0

18223 150 6000.0

18224 100 7000.0

Sample Input / Output 2:

Enter the source

Malaysia

Enter the destination

Dubai

No flights available for the given source and destination

Automatic evaluation [+]

Flight.java

```
1  public class Flight {  
2      private int flightId;  
3      private String source;  
4      private String destination;  
5      private int noOfSeats;  
6      private double flightFare;  
7      public int getFlightId() {  
8          return flightId;  
9      }  
10     public void setFlightId(int flightId) {  
11         this.flightId = flightId;  
12     }  
13     public String getSource() {  
14         return source;  
15     }  
16     public void setSource(String source) {  
17         this.source = source;  
18     }  
19     public String getDestination() {  
20         return destination;  
21     }  
22     public void setDestination(String destination) {  
23         this.destination = destination;  
24     }  
25     public int getNoOfSeats() {  
26         return noOfSeats;  
27     }  
28     public void setNoOfSeats(int noOfSeats) {  
29         this.noOfSeats = noOfSeats;  
30     }  
31     public double getFlightFare() {  
32         return flightFare;  
33     }  
34     public void setFlightFare(double flightFare) {  
35         this.flightFare = flightFare;  
36     }  
37     public Flight(int flightId, String source, String destination,  
38                 int noOfSeats, double flightFare) {  
39 }
```

```

41         super();
42         this.flightId = flightId;
43         this.source = source;
44         this.destination = destination;
45         this.noOfSeats = noOfSeats;
46         this.flightFare = flightFare;
47     }
48
49
50
51 }
52

```

FlightManagementSystem.java

```

1 import java.sql.Connection;
2 import java.sql.ResultSet;
3 import java.sql.SQLException;
4 import java.util.ArrayList;
5 import java.util.List;
6 import java.sql.PreparedStatement;
7 public class FlightManagementSystem {
8     public ArrayList <Flight> viewFlightBySourceDestination(String source, String destination){
9         Connection conn = null;
10        ResultSet Rs = null;
11        String sql = "select * from flight where source=? and destination=? order by flightid";
12        ArrayList<Flight> flight = new ArrayList<>();
13        try{
14            conn = DB.getConnection();
15            PreparedStatement ps = conn.prepareStatement(sql);
16
17            ps.setString(1, source);
18            ps.setString(2, destination);
19
20            Rs=ps.executeQuery();
21            while(Rs.next()){
22                Flight F = new Flight(Rs.getInt(1),source,destination,Rs.getInt(4),Rs.getInt(5));
23                flight.add(F);
24            }
25        }catch(ClassNotFoundException e){
26            e.printStackTrace();
27        }catch(SQLException e){
28            e.printStackTrace();
29        }
30
31        return flight;
32    }
33 }

```

Main.java

```

1 import java.util.Comparator;
2 import java.util.Scanner;
3 import java.util.ArrayList;
4
5
6 public class Main{
7     public static void main(String[] args){
8         Scanner sc=new Scanner(System.in);
9         // fill your code here
10        System.out.println("Enter the source");
11        String source=sc.nextLine();
12        System.out.println("Enter the destination");
13        String destination=sc.nextLine();
14        ArrayList<Flight> flight = new
FlightManagementSystem().viewFlightBySourceDestination(source,destination);
15        if(flight.isEmpty())
16        {
17            System.out.println("No flights available for the given source and destination");

```

```

18
19     }
20     else
21     {
22         System.out.println("Flightid Noofseats Flightfare");
23         for(Flight f : flight)
24         {
25             System.out.println(f.getFlightId()+" "+f.getNoOfSeats()+" "+f.getFlightFare());
26         }
27     }
28
29
30 }
31}

```

DB.java

```

1 import java.io.FileInputStream;
2 import java.io.IOException;
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.SQLException;
6 import java.util.Properties;
7
8 public class DB {
9
10    private static Connection con = null;
11    private static Properties props = new Properties();
12
13
14 //ENSURE YOU DON'T CHANGE THE BELOW CODE WHEN YOU SUBMIT
15    public static Connection getConnection() throws ClassNotFoundException, SQLException {
16        try{
17
18            FileInputStream fis = null;
19            fis = new FileInputStream("database.properties");
20            props.load(fis);
21
22            // load the Driver Class
23            Class.forName(props.getProperty("DB_DRIVER_CLASS"));
24
25            // create the connection now
26            con =
27        DriverManager.getConnection(props.getProperty("DB_URL"),props.getProperty("DB_USERNAME"),props.getProperty("DB_PASSWORD"));
28        }
29        catch(IOException e){
30            e.printStackTrace();
31        }
32        return con;
33    }
34}

```

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 IN DB.java using this
properties file only.
5 #Do not hard code the values in DB.java.
6
7 DB_DRIVER_CLASS=com.mysql.jdbc.Driver
8 DB_URL=jdbc:mysql://localhost:3306/${sys:DB_USERNAME}
9 DB_USERNAME=${sys:DB_USERNAME}
10 DB_PASSWORD=${sys:DB_USERNAME}
11

```

Grade

Reviewed on Wednesday, 12 May 2021, 6:31 AM by Automatic grade

Grade 100 / 100

Assessment report

[\[+\]Grading and Feedback](#)

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 lineItem
 * @return boolean
 */
public boolean isBulkOrder(OrderLineItem lineItem) {
    boolean result=false;

    if(lineItem.quantity>5){
        result = true;
    }
    else if(lineItem.quantity<=5 && lineItem.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;  
        }  
        balanceAmount=total-(float)withdrawAmt;  
        return true;  
    }  
}
```

```
 }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();
        }
    }
}
```

```

        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");
}

}

}
```

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.

```
public class UserMainCode {  
    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=84882;
        System.out.println(SumOfOddsAndEvens(n));
    }
}

```

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

```
import java.util.*;  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        Scanner s=new Scanner(System.in);  
  
        String pan=s.next();  
  
        int b=panNumberValidation(pan);  
  
        if(b==1)  
  
            System.out.println("valid Pancard Number");  
  
        else  
  
            System.out.println("not a valid credential");  
  
    }  
  
    public static int panNumberValidation(String input) {  
  
        int b=0;  
  
        if(input.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

```
public class UserMainCode
```

```
{
```

```
    public static int display(int number){
```

```
        int n1=0,n2=0;
```

```
        while(number!=0)
```

```
{
```

```
        n1=number%10;
```

```
        if((n1%2)==0)
```

```
n2+=n1*n1;  
  
number/=10;  
  
}  
  
return n2;  
  
}  
  
}
```

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

```

import java.util.Scanner;

public class Middle {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        String s=sc.nextLine();
        StringBuffer sb=new StringBuffer();
        if(s.length()%2==0)
        {
            sb.append(s.substring(s.length()/2-1,s.length()/2+1));
            //System.out.println(sb.toString());
        }
        System.out.println(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

```
import java.util.Scanner;
public class Main {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);
        String s=sc.nextLine();
        int res=UserMainCode.checkCharacter(s);
        if(res==1)
        {
            System.out.println("Valid");
        }
        else
            System.out.println("Invalid");
    }
}
```

```
public class UserMainCode {
    public static int checkCharacter(String s)

    {
        int res=-1;
        if(s.charAt(0)==s.charAt(s.length()-1))
        {
            res=1;
        }
    }
}
```

```
    }
    return res;
}
}
```

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

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

```

public class Main {

    public static void main(String[] args) {

Scanner sc=new Scanner(System.in);
String s=sc.nextLine();
int n=sc.nextInt();
System.out.println(UserMainCode.stringChange(s,n));

}

}

public class UserMainCode {
    public static String stringChange(String s,int n)
    {

        StringBuffer sb=new StringBuffer();
        sb.append(s.substring(0,n));
        sb.append(s.substring(s.length()-n));
        return sb.toString();
    }
}

```

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

```
import java.util.Scanner;
public class Main
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        int a=sc.nextInt();
        System.out.println(UserMainCode.reverse(a));
    }
}
```

```
public class UserMainCode
```

```

{
    public static int reverse(int a)
    {
        String s=String.valueOf(a);
        StringBuffer sb=new StringBuffer(s);
        sb.reverse(); //reverse return type is void
        int res=Integer.parseInt(sb.toString());
        return res;
    }
}

```

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 arrayList, 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

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

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;
public class Qus8Main {

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        String s=sc.nextLine();
        SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
        try {
            Date d=sdf.parse(s);
            System.out.println("Valid date format");
        } catch (ParseException e) {
            System.out.println("Invalid date format");
        }
    }
}
```

```

sdf.setLenient(false);
int res=0;
if(s.matches("[0-9]{2}(/)[0-9]{2}(/)[0-9]{4}"))
{
    try {
        Date d=sdf.parse(s);
        res=1;
    } catch (ParseException e) {
        res=-1;
    }
    System.out.println(res);
}
}
}

```

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

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Scanner;
public class Qus8Main {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);
        String s=sc.nextLine();
        SimpleDateFormat sdf=new SimpleDateFormat("hh:mm a");
        sdf.setLenient(false);
        int res=0;

        try {
            Date d=sdf.parse(s);
            res=1;
        } catch (ParseException e) {
            res=-1;
        }

        System.out.println(res);

    }
}
```

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

```
public class Main {  
  
    public static void main(String[] args) {  
  
        String s1="zzzz";  
  
        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();

    }

}

```

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

```
public class UserMainCode {  
  
    public static int display(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:

cmmitmnt

Sample Input 2:

capacity

Sample Output 2:

Cpcty

```
public class Main {  
  
    public static void main(String[] args) {  
  
        String s1="capacity";
```

```

        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)!='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
```

```
public class useerm{  
  
    public static int display(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 ≥ 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

```
import java.util.Arrays;

public class kape1 {

    public static int display(int []array)

    {

        Arrays.sort(array);

        int n=array[array.length-1]-array[0];

        int b=array.length;

        if(b==1)

        {

            n=array[0];

        }

        return n;

    }

}
```

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
```

17. Generate the series

Given a method taking an odd positive Integer number as input. Write code to evaluate the following series:

$1+3-5+7-9\dots+/-n$.

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

```
public class UserMainCode
{
    public static int generateSeries(int n)
    {
        int i=0,sumo=0,sume=0,sum=1;
        if(n==1)
        {
            sum=n;
            break;
        }

        for(i=3;i<=n;i=i+4)
        {

            sumo=sumo+i;
        }
        for(i=5;i<=n;i=i+4)
        {
            sume=sume+i;
        }
    }
}
```

```
sum+=sumo-sume;  
return sum;  
}  
}
```

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

```

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();
int c=sc.nextInt();
System.out.println(UserMainCode.calculateElectricityBill(s1,s2, c));

}
}

```

```

public class UserMainCode {
    public static int calculateElectricityBill(String s1,String s2,int c)

    {

        int a=Integer.parseInt(s1.substring(5));
        int b=Integer.parseInt(s2.substring(5));
        int res=Math.abs((b-a)*c);
        return res;
    }
}

```

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

```
public class Main {  
  
    public static void main(String[] args) {  
  
        String s1="goodbad";  
  
        getvalues(s1);  
  
    }  
  
    public static void getvalues(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

```
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();

        System.out.println(UserMainCode.concatString(s1,s2));
    }
}
```

```
public class UserMainCode
{
    public static String concatString(String s1, String s2)
    {
```

```

StringBuffer sb=new StringBuffer();
int n=s2.length();
sb.append(s1.substring(s1.length()-n));
sb.append(s2.substring(0));
return sb.toString();
}
}

```

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

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        String s=sc.nextLine();
        //UserMainCode u=new UserMainCode();
        int b=UserMainCode.validateColorCode(s);
        if(b==1)
        {
            System.out.println("Valid color code");

        }
        else
        {
            System.out.println("Invalid color code ");
        }
    }

    public class UserMainCode
    {
        public static int validateColorCode(String a)
        {
            int r=-1;
            if(a.matches("(#[A-F0-9]{6}))")
            {
                r=1;
            }
            return r;
        }
    }
}
```

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

CTS-215

```
public class Main {
```

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

```
        String s1="CTS-2j4";
```

```
        getvalues(s1);
```

```
}
```

```
public static void getvalues(String s1) {  
    if(s1.matches("(CTS)[-]{1}[0-9]{3}"))  
    {  
        System.out.println(1);  
    }  
    else  
        System.out.println(-1);  
}  
}
```

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

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 array. The 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

```
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.checkLargestAmongCorner(a));
}

}
```

```
public class UserMainCode
{
    public static int checkLargestAmongCorner(int a[])
    {
        int max=0;
        int m=a[a.length/2];
        int f=a[0];
        int l=a[a.length-1];
        if(m>f && m>l)
        {
            max=m;
        }
        else if(f>m && f>l)
        {
            max=f;
        }
        else
```

```

        max=1;

    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
```

```

import java.util.Scanner;
public class Main {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        int r=sc.nextInt();

        System.out.println(UserMainCode.calculateNcr(n, r));
    }
}

```

```

}

}

public class UserMainCode
{
    public static int calculateNcr(int n,int r)
    {
        int i,prod=1,prod1=1,prod2=1;
        for(i=1;i<=n;i++)
        {
            prod=prod*i;

        }
        for(i=1;i<=r;i++)
        {
            prod1=prod1*i;
        }
        int diff=n-r;
        for(i=1;i<=diff;i++)
        {
            prod2=prod2*i;
        }
        int dem=prod1*prod2;
        int res=prod/dem;
        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

```
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.display(a,b);  
  
if(u== -1)  
  
System.out.println("No common elements");  
  
else  
  
System.out.println(u);}  
  
}
```

```
public class UserMainCode {  
  
public static int display(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;
```

```
}
```

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

```
public class UserMainCode {  
    public static int validatePassword(String password) {  
        String regEx = "[^0-9|@|_|#](.)*[@|_|#](.)*[^@|_|#]";  
        if(password.length() >= 8 && password.matches(regEx)) {  
            return 1;  
        }  
        return -1;  
    }  
}  
  
if(s.length() >= 8 && s.matches("[^0-9|#|_|@](.)*[#|_|@](.)*[^@|_|#]"))  
{  
    System.out.println("valid");  
}
```

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

```
import java.util.*;  
  
public class Main {  
  
    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// valid id");

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.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
```

```
import java.util.*;
```

```
public class Main
```

```
{
```

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

```
{
```

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

        int n=sc.nextInt();
        sc.nextLine();
```

```
String[] a=new String[n];
for(int i=0;i<n;i++)
    a[i]=sc.nextLine();

int m=sc.nextInt();

System.out.println(UserMainCode.display(a,m));

}

import java.util.*;

public class UserMainCode

{

    public static int display(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

```
import java.text.*;
```

```
import java.util.*;
```

```
public class Main {  
  
    public static void main(String[] args) throws ParseException {  
  
        String s1="2012-03-01";  
  
        String s2="2012-03-16";  
  
        System.out.println(monthsBetweenDates(s1,s2));  
  
    }  
  
    public static int monthsBetweenDates(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

```
public class Main {  
  
    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;

}

}

```

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

```
import java.util.*;

public class Main {

    public static void main(String[] args) {

        String ipAddress="10.230.110.160";

        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,".");

        String s=t.nextToken();

        int a=Integer.parseInt(s);
```

```

        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

```
import java.text.*;  
  
import java.util.*;  
  
public class Main {  
  
    public static int dateDifference(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;  
    }  
  
    public static void main(String[] args) throws ParseException {
```

```

String s1="2012-03-12";
String s2="2012-03-14";
System.out.println(dateDifference(s1,s2));
}

}

```

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

```

import java.util.*;
public class Main {
    public static String extensionString(String s1){
        StringTokenizer t=new StringTokenizer(s1,".");
        String ss=t.nextToken();
        String s2=t.nextToken();
    }
}

```

```

        return s2;

    }

    public static void main(String[] args) {

        String s1="sun.gif";

        System.out.println(extensionString(s1));

    }

}

```

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:

[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

```
import java.util.Arrays;  
  
import java.util.StringTokenizer;  
  
public class PO  
  
{  
  
    public static int display(String s, String s1)  
  
    {  
  
        int c=0, m=0; String t=null;  
  
        char a[] = s.toCharArray();  
  
        char b[] = s1.toCharArray();  
  
        Arrays.sort(a);  
  
        Arrays.sort(b);  
  
        String s2 = new String(a);  
  
        String s3 = new String(b);  
  
        StringTokenizer st = new StringTokenizer(s2);  
  
        StringTokenizer st1 = new StringTokenizer(s3);  
  
        while(st.hasMoreTokens() && st1.hasMoreTokens())  
        {  
            String s4 = st.nextToken();  
            String s5 = st1.nextToken();  
  
            if(s4.equals(s5))  
                c++;  
        }  
  
        System.out.println(c);  
    }  
}
```

```
s1=st1.nextToken();

if(s.length()>s1.length())

{t=s1;

s1=s;

s=t;

}

for(int i=0;i<s.length();i++)

{

for(int j=0;j<s1.length();j++)

{

if(s.charAt(i)==s1.charAt(j))

{

if((s.indexOf(s.charAt(i))==s.lastIndexOf(s.charAt(i)))&&(s1.indexOf(s1.charAt(j))==s1.lastIndexOf(s1.charAt(j))))



{



c++;



}

}

}

return c;

}

}
```

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

```
import java.util.StringTokenizer;

public class Main {

    public static void main(String[] args) {

        String s1="vishal jadiya";

        getvalues(s1);

    }

    public static void getvalues(String s1) {

        StringBuffer sb=new StringBuffer();

        StringTokenizer st=new StringTokenizer(s1, " ");

        String s2=st.nextToken();

    }

}
```

```

String s3=st.nextToken();

sb.append(s3);

sb.append(" , ");

sb.append(s2.substring(0,1).toUpperCase());

System.out.println(sb);

}

}

```

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

```

import java.util.Scanner;
public class Qus8Main {

    public static void main(String[] args) {

```

```

Scanner sc=new Scanner(System.in);
String name=sc.nextLine();
char ch=sc.nextLine().charAt(0);
StringBuffer sb=new StringBuffer(name);
for(int i=0;i<sb.length();i++)
{if(ch==sb.charAt(i))
{
    sb.deleteCharAt(i);
    i--;
}
}
System.out.print(sb.toString());
} }

```

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

```

import java.util.Scanner;

public class Qus8Main {

    public static void main(String[] name)
    {
        Scanner sc=new Scanner(System.in);
        String s=sc.nextLine();
        System.out.println(Qus8.display(s));
    }
}

public class Qus8 {
    public static int display(String name) {

        String s1=name;
        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)
            return 1;
        else
            return 0 ;
        }
    }
}

```

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

```
import java.util.Scanner;

public class Qus8Main {

    public static void main(String[] args) {
        String s1="TRAINER";
        getvalues(s1);
    }
    public static void getvalues(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);
            }
            System.out.println(sb);
        }
        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));
            System.out.println(sb);
        }
    }
}
```

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

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 a `HashMap<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

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:

aaaxxyzAAAx

Sample Output 2:

2

```
public class Main {  
  
    public static void main(String[] args) {  
  
        String input1="aaxxyzAAx";  
  
        System.out.println(consecutiveRepeatingOfChar(input1));  
  
    }  
}
```

```

public static int consecutiveRepeatingOfChar(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 chunk 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

```
import java.util.*;  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        String s1="You are toooo 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);  
  
}  
  
}
```

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

```
public class Main {  
    public static void main(String[] args) {  
        String s1="HelloWorld";  
        getvalues(s1);  
    }  
  
    public static void getvalues(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());

}
}
```

45) Name Shrinking

Write a program that accepts a string as input and converts the first two names into dot-separated initials and prints 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 **getFormatedString** which accepts a string. The return type (String) should return the shrunk 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

```
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.getFormatedString(s));
    }
}
```

```
import java.util.StringTokenizer;
```

```
public class UserMainCode {
    public static String getFormatedString(String s)

    {
        StringTokenizer st=new StringTokenizer(s," ");
        StringBuffer sb=new StringBuffer();
        while(st.hasMoreTokens())
        {
            String a=st.nextToken();
            String b=st.nextToken();
            String c=st.nextToken();
            sb.append(c.substring(0));
            sb.append(" ");
            sb.append(b.substring(0,1));
        }
    }
}
```

```

        sb.append(".");
        sb.append(a.substring(0,1));
        //String ss=sb.toString();
    }
    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)

```

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

```
public class kape {  
    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(kape1.getSum(s2));  
    }  
}
```

```
public class kape1 {  
  
    public static int 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)){  
                    if(c%2!=0)  
                        {  
                            String t=String.valueOf(c);  
                            sum+=Integer.parseInt(t);  
                        }  
                }  
            }  
        return sum;  
    }  
}
```

```
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

```
public class useer{
```

```
public static void main(String[]args)
{
    Scanner sc=new Scanner(System.in);
    int n=sc.nextInt();
    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++;
            }
        }
    }
    if(count>0)
```

```

    {
        System.out.println("Invalid/not unique");

    }

else

{
    System.out.println("valid/unique");

}

}}

```

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

49) Color Code Validation

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

```
import java.util.Scanner;

class Main
{
    public static void main(String[] a)
    {
        Scanner sc=new Scanner(System.in);
        String s=sc.nextLine();
        if(s.matches("(#)[A-Z0-9]{6}"))
        {
            System.out.println("valid");
        }
        else
            System.out.println("invalid");
    }
}
```

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

```
import java.util.*;  
  
public class useerm {  
  
    public static String lengthiestString(String s1,int n){  
  
        StringBuffer sb=new StringBuffer();  
  
        sb.append(s1);  
  
        for(int i=0;i<n;i++)  
  
    }  
}
```

```

        sb.append(s1.substring(s1.length()-n,s1.length()));

// sb.append(s1.substring(s1.length()-n))

}

return sb.toString();

}

public static void main(String[] args) {

    Scanner s=new Scanner(System.in);

    System.out.println("enter the String:");

    String s1=s.nextLine();

    int n=s.nextInt();

    System.out.println("the lengthiest string is:"+lengthiestString(s1,n));

}

}

```

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

```
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.calculateBornDay(s1));
    }
}

import java.text.SimpleDateFormat;
import java.text.ParseException;
import java.util.Date;
public class UserMainCode {
    public static String calculateBornDay(String s1) throws ParseException
    {
        SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");
        SimpleDateFormat sdf1=new SimpleDateFormat("EEEE");
        Date d=sdf.parse(s1);
        String s=sdf1.format(d);
        return s.toUpperCase();
    }
}
```

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

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

```
import java.util.Date;  
  
import java.text.SimpleDateFormat;  
  
public class Usermaincode  
  
{public static boolean display(String s,String s1,int n)  
  
{  
  
boolean b=false;  
  
SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");  
  
try{  
  
Date d=sdf.parse(s);  
  
Date d1=sdf.parse(s1);  
  
int y=d.getYear();  
  
int y1=d1.getYear();  
  
int m=d.getMonth();  
  
int m1=d1.getMonth();  
  
int day=d.getDay();  
  
int day1=d1.getDay();  
  
int age=y1-y;  
  
if(m>m1)  
  
age--;  
  
else if(m==m1)  
  
{if(day<day1)
```

```
age--;
}
if(age==n)
b=true;
else
b=false;
}
catch(Exception e)
{e.printStackTrace();
}
return b;
}
}
```

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

```
public class User {  
    public static String repeatString (String s)  
    {  
        StringBuffer sb=new StringBuffer();  
        for(int i=0;i<s.length();i++)  
        {  
  
            /* char c=s.charAt(i);  
            if(!Character.isAlphabetic(c)) */  
  
            // if(!Character.isAlphabetic(s.charAt(i))  
            && (!charachter.isWhiteSpace(s.CharAt(i))))  
  
            if( (!Character.isAlphabetic(s.charAt(i)))  
                && (!Character.isWhitespace(s.charAt(i)))) )  
  
                sb.append(s.charAt(i));  
        }  
        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

```
public class User {  
    public static String repeatString (String s,int n)  
    {  
        StringBuffer sb=new StringBuffer();  
        if(n>=3)  
        {  
            for(int i=0;i<n;i++)  
            {  
                sb.append(s.substring(0,3)).append(" ");  
            }  
        }  
        else if(n==2)  
        {  
            for(int i=0;i<n;i++)  
                sb.append(s.substring(0,2)).append(" ");  
        }  
        else if(n==1)  
        {  
            for(int i=0;i<n;i++)  
                sb.append(s.substring(0,1)).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
```

```

public class User {
    public static float averageElements(int a[],int n)
    {
        int c=0,sum=0,k=0;
        float avg=0;
        for(int i=2;i<=n;i++)
        {
            c=0;
            for(int j=1;j<=i;j++)
            {
                if(i%j==0)
                    c++;
            }
            if(c==1)
            {
                k++;
                sum=sum+a[i];
            }
        }
        avg=(float)sum/k;
    return avg;
    }
}

```

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
```

```
public class User {  
    public static int getMiddleElement (int a[],int b[],int n)  
    {  
        int sum=0;  
        for(int i=0;i<n;i++)  
        {  
            for(int j=0;j<n;j++)  
            {  
                if(a[i]==b[j])  
                    sum=sum+a[i];  
            }  
        }  
        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
```

```
public class User {  
    public static int getMiddleElement (int a[])  
    {  
        int n=a.length;  
        int mid=n/2;  
        return a[mid];  
    }  
}
```

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

```
public class User {  
    public static String getString(String s)  
    {  
        StringBuffer sb=new StringBuffer();  
        char a=s.charAt(0);  
        char b=s.charAt(1);  
        if(a!='j'&& b!='b')  
            sb.append(s.substring(2));  
        else if(a=='j' && b!='b')  
            sb.append("j").append(s.substring(2));  
        else if(a!='j' && b=='b')  
            sb.append(s.substring(1));  
        else  
            sb.append(s.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

```
public class User {
    public static int getValidDate(String s) throws ParseException
    {
        int res=0;
        // String s1=null, s2=null, s3=null;
        if(s.matches("[0-9]{2}\\.?[1]\\.?[0-9]{2}\\.?[1]\\.?[0-9]{4}"))
        {
            SimpleDateFormat sdf1=new SimpleDateFormat("dd.MM.yyyy");
            sdf1.setLenient(false);
            try
            {
```

```

        Date d1=sdf1.parse(s);
        res=1;
    }
    catch (ParseException e)
    {
        res=-1;
    }
}

else if(s.matches("[0-9]{2}[-]{1}[0-9]{2}[-]{1}[0-9]{4}"))
{

    SimpleDateFormat sdf3=new SimpleDateFormat("dd-MM-
yyyy");
    sdf3.setLenient(false);
    try
    {
        Date d1=sdf3.parse(s);
        res=1;
    }
    catch (ParseException e)
    {
        res=-1;
    }
}
else if(s.matches("[0-9]{2}[/]{1}[0-9]{2}[/]{1}[0-9]{4}"))
{

    SimpleDateFormat sdf3=new
SimpleDateFormat("dd/MM/yyyy");
    sdf3.setLenient(false);
    try
    {
        Date d1=sdf3.parse(s);
        res=1;
    }
    catch (ParseException e)
    {
        res=-1;
    }
}
else
    res=0;

return res;
}
}

```

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 :

```
6  
3  
6  
9  
4  
2  
5
```

Sample Output:

```
5.5
```

```
public class User {  
    public static float getBoundaryAverage(int a[], int n)  
    {  
        int sum=0;  
        float avg=0;  
        Arrays.sort(a);  
        sum=a[0]+a[n-1];  
        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

```
public class User {  
    public static int countVowels(String s) throws ParseException  
    {  
        int count=0;  
        for(int i=0;i<s.length();i++)  
        {  
            if(s.charAt(i)=='a' ||s.charAt(i)=='A' ||  
                s.charAt(i)=='e' ||s.charAt(i)=='E' ||  
                s.charAt(i)=='i' ||s.charAt(i)=='I' ||  
                s.charAt(i)=='o' ||s.charAt(i)=='O' ||  
                s.charAt(i)=='u'  
                ||s.charAt(i)=='U')  
                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

```
public class User {  
    public static String getMonthName(String s) throws ParseException  
{  
    SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");  
    Date d=sdf.parse(s);  
  
    SimpleDateFormat sdf1=new SimpleDateFormat("MMMM");  
    String month=sdf1.format(d);  
  
    return month.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 “**public class User {**
” 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
```

```
import java.util.Scanner;  
public class Main {  
    public static void main(String[] args) {  
  
        Scanner sc=new Scanner(System.in);  
        String s=sc.nextLine();  
        int n1=sc.nextInt();  
        int n2=sc.nextInt();  
        System.out.println(UserMainCode.reverseSubstring(s,n1,n2));  
  
    }  
}  
  
public class UserMainCode {  
    public static String reverseSubstring(String s,int n1,int n2)  
  
    {  
        StringBuffer sb=new StringBuffer(s);  
        sb.reverse();  
        String ss=sb.substring(n1,n1+n2);  
        return ss.toString();  
    }  
}
```

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

```
public class User {  
    public static int stringFinder(String str, String s1, String s2)  
    {  
        int res=0;  
        if(str.contains(s1)&&str.contains(s2))  
        {  
            if(str.indexOf(s1)<str.indexOf(s2))  
                res=1;  
            else  
                res=0;  
        }  
        return res;  
    }  
}
```

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 a 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 1:

Invalid

```
public class User {
    public static int validatePhoneNumber(String s)
    {
        int res=0;
        if(s.matches("[0-9]{3}(-)[0-9]{3}(-)[0-9]{4}"))
            res=1;
        else
            res=-1;
        return res;
    }
}
```

68) 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 correspond to the number of days in the month in the given year.

Sample Input:

2000

1

Sample Output:

29

```
public class User {  
    public static int getNumberOfDays(int year, int month)  
    {  
        GregorianCalendar gc=new GregorianCalendar(year,month,1);  
  
        int days=gc.getActualMaximum(Calendar.DAY_OF_MONTH);  
        return days;  
    }  
}
```

69) 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

```
public class User {  
    public static String validateNumber(String s)  
    {  
        StringTokenizer st=new StringTokenizer(s, " ");  
        StringBuffer sb=new StringBuffer();  
        while(st.hasMoreTokens())  
        {  
            String r=st.nextToken();  
            if(r.equals("is"))  
            {  
                sb.append(r.replace("is", "is not"));  
            }  
            else  
                sb.append(r);  
            sb.append(" ");  
        }  
  
        // sb.deleteCharAt((sb.length()-1));  
  
        return sb.toString();  
    }  
}
```

70) 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

```
public class User {  
    public static String validateNumber(String s)  
    {  
        String res=null;  
        int count=0;  
        for(int i=1;i<s.length();i++)  
        {  
            char c=s.charAt(i);  
            if(Character.isDigit(c))  
                count++;  
            }  
            if(count==s.length()-1)  
            {  
                res=String.valueOf(Math.abs(Integer.parseInt(s)));  
            }  
            else  
                res="-1";  
        return res;  
    }  
}
```

71) 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

```
public class User {  
    public static int countSeven (int n)  
    {  
        int count=0, r=0;  
        while(n>0)  
        {  
            r=n%10;  
            if(r==7)  
                count++;  
            n=n/10;  
        }  
        return count;  
    }  
}
```

72) 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 Output 1:

hixxx

Sample Input 2:

XXxxtest

Sample Output 2:

XXtestxx

```
public class User {  
    public static String getStringUsingNthCharacter (String s)  
    {  
        StringBuffer sb=new StringBuffer();  
        StringBuffer sb1=new StringBuffer();  
        for(int i=0;i<s.length();i++)  
        {  
            if(s.charAt(i)=='x')  
            {  
                sb1.append(s.charAt(i));  
            }  
            else  
            {  
                sb.append(s.charAt(i));  
            }  
        }  
        sb.append(sb1);  
  
        return sb.toString();  
    }  
}
```

73) 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>=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:

HelloWorld
2

Sample Output 1:

HeIWrd

```
public class User {  
    public static String getStringUsingNthCharacter (String s,int n)  
    {  
        StringBuffer sb=new StringBuffer();  
        sb.append(s.charAt(0));  
        for(int i=1;i<s.length();i=i+n)  
            sb.append(s.charAt(i));  
        return sb.toString();  
    }  
}
```

74) 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

```
public class User {  
    public static boolean compareLastDigit (int a,int b)  
    {  
        boolean b1=false;  
        int r1=a%10;  
        int r2=b%10;  
        if(r1==r2)  
            b1=true;  
        return b1;  
    }  
}
```

75) 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

```
public class User {
    public static int getDistinctSum (int a,int b,int c)
    {
        int sum=0;
        if(a==b&&a==c&&b==c)
            sum=0;
        else if(a!=b&&b!=c&&a==c)
            sum=b;
        else if(a==b&&b!=c&&a!=c)
            sum=c;
        else if(a!=b&&b!=c&&a!=c)
            sum=a+b+c;
        return sum;
    }
}
```

```
int sum=0;
if(a==b&&a==c&&b==c)
    sum=0;
else if(a!=b&&a!=c&&b==c)
    sum=a;
else if(a!=b&&b!=c&&a==c)
    sum=b;
else if(a==b&&b!=c&&a!=c)
    sum=c;
else
    sum=a+b+c;
return sum;
```

```
public class Main {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int a=sc.nextInt();
        int b=sc.nextInt();
        int c=sc.nextInt();
int sum=0;
        if(a!=b&&a!=c&&b!=c)
        {
```

```

        sum=a+b+c;
    }

    else if(a==c)

    {
        sum=b;
    }
    else if(a==b)
    {
        sum=c;
    }
    else if(b==c)
    {
        sum=a;
    }
else
    sum=0;
System.out.println(sum);
}
}

```

76) 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

```

public class User {
    public static boolean validateString (String s)
    {
        boolean b=false;
        if(s.charAt(1)=='i'&&s.charAt(2)=='x')
            b=true;
        return b;
    }
}

```

77) 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

```

public class HelloWorld {

    public static void main(String[] args) {
        // String s1="hello world";
        Scanner sc=new Scanner(System.in);
        String s1=sc.nextLine();
        String ss=Hello.display(s1);
        System.out.println(ss);

    }
}

```

```

public class Hello {

    public static String display(String s1) {

        StringTokenizer st=new StringTokenizer(s1, " ");
        StringBuffer sb=new StringBuffer();
        while(st.hasMoreTokens())
        {
            String a=st.nextToken();
            String b=st.nextToken();
            sb.append(b.substring(b.length()-1));
            sb.append(a.substring(1));
            sb.append(" ");
            sb.append(b.substring(0,b.length()-1));
            sb.append(a.substring(0,1));
        }
        return sb.toString();
    }

}

public class WhiteSpaxc {
    public static String validateNumber(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();
    }
}

```

78) 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

```
public class Main {  
    public static void main(String[] args) {  
        Scanner sc=new Scanner(System.in);  
        String s=sc.next();  
        boolean b=User.validateString(s);  
        System.out.println(b);  
    }  
}
```

```
public class User {  
    public static boolean validateString (String s)  
    {  
        boolean b=false;  
        if(s.length()==3)  
        {  
            if(s.matches("[A-Za-z]{3}"))  
                b=true;  
        }  
        return b;  
    }  
}
```

79) 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

```
public class User {  
    public static String repeatString(String s,int n)  
    {  
        StringBuffer sb=new StringBuffer();  
        for(int i=0;i<n;i++)  
        {  
            sb.append(s);  
        }  
        return sb.toString();  
    }  
}
```

80) 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

```
public class User {  
    public static int SumOdd (int n)  
    {  
        int sum=0;  
        for(int i=1;i<=n;i++)  
        {  
            if(i%2!=0)  
                sum=sum+i;  
        }  
        return sum;  
    }  
}
```

81) 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

```
public class Main {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        String[] s=new String[n];
        for(int i=0;i<n;i++)
            s[i]=sc.next();
        System.out.println(User.concatString(s));
    }
}

public class User {
    public static String concatString (String s[])
    {
        StringBuffer sb=new StringBuffer();
        sb.append(s[0]);
        for(int i=1;i<s.length;i++)
        {
            sb.append(",");
            sb.append(s[i]);
        }
        return sb.toString();
    }
}
```

82) 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

```
public class Main {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int a=sc.nextInt();
        int b=sc.nextInt();
        int c=sc.nextInt();
        System.out.println(User.calculateUnique(a,b,c));
    }
}
```

```
public class User {
    public static int calculateUnique(int a,int b,int c)
    {
        int count=0;
        int[] s={a,b,c};
        int[] res=new int[3];
        for(int i=0;i<s.length;i++)
        {
            res[i]=Math.abs(s[i]);
        }
        count=0;
        for(int i=0;i<res.length-1;i++)
        {

            if(res[i]==res[i+1])
            {
                count++;
            }
        }
        return count+1;
    }
}
```

```
public class Main {
```

```
    public static void main(String[] args) {
        int ct1=0;
        Scanner sc=new Scanner(System.in);
```

```

        int a=sc.nextInt();
        int b=sc.nextInt();
        int c=sc.nextInt();

        if(a!=b)
            ct1=ct1+1;

        if(a!=c)
            ct1=ct1+1;

        if(b!=c)
            ct1=ct1+1;

        if((a==b) & (b==c))
            System.out.println("output "+(ct1+1));
        else
            System.out.println("output "+ct1);

    }
}

```

```

public class Main {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);
        int a=sc.nextInt();
        int b=sc.nextInt();
        int c=sc.nextInt();

        int d=0;
        if(a!=b&&a!=c)
        {
            d=3;
        }
        else if(a==b&&a==c)
        {
            d=1;
        }
        else if(a!=b&&a==c)

        {
            d=2;
        }
        else if(a==b&&a!=c)
        {
            d=2;
        }
    }
}

```

```

        System.out.println(d);
    }
}

```

83) 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
```

```

public class Main {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int a=sc.nextInt();
        int b=sc.nextInt();
        char op=sc.next().charAt(0);
        System.out.println(User.calculateUnique(a,b,op));
    }
}

```

```

public class User {
    public static int calculateUnique(int a,int b,char op)
    {
        int res=0;
        switch(op) {
            case '+':

```

```

        res=a+b;
case '-':
    res=Math.abs(a-b);
case '*':
    res=a*b;
case '/':
    res=Math.round(a/b);
case '%':
    res=Math.round(a%b);
}
return res;
}
}

```

84) 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

```

public class User {
    public static boolean scanArray(int s[])
    {
        boolean b=false;

        for(int i=0;i<s.length-1;i++)
        {
            if(s[i]==100&&s[i+1]==100)
            {
                b=true;
                break;
            }
            else
                b=false;
        }

        return b;
    }
}

```

85) 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

```
public class Main {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        int[] s= new int[n];
        for(int i=0;i<n;i++)
            s[i]=sc.nextInt();
        boolean b=User.scanArray (s);
        System.out.println(b);
    }
}
```

```
public class User {
    public static boolean scanArray(int s[])
    {
        boolean b=false;
        if(s.length>4)
        {
            for(int i=0;i<4;i++)
            {
                if(s[i]==9)
                    b=true;
                else
                    b=false;
            }
        }
        else
        {
            for(int i=0;i<s.length;i++)
            {
                if(s[i]==9)
                    b=true;
                else
                    b=false;
            }
        }
        return b;
    }
}
```

```
}
```

86) 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
```

```
public class Main {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
```

```

        String[] s= new String[n];
        for(int i=0;i<n;i++)
            s[i]=sc.next();
        int n1=sc.nextInt();
        System.out.println(User.countWord (s,n1));
    }
}

public class User {
    public static int countWord (String s[],int n1)
    {
        int count=0;
        for(int i=0;i<s.length;i++)
        {
            if(s[i].length()==n1)
                count++;
        }
        return count;
    }
}

```

87) 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 $((x_1-x_2)*(x_1-x_2)+(y_1-y_2)*(y_1-y_2))$

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

```
public class Main {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int x1=sc.nextInt();
        int y1=sc.nextInt();
        int x2=sc.nextInt();
        int y2=sc.nextInt();
        System.out.println(User.findDistance(x1,y1,x2,y2));
    }
}

public class User {
    public static int findDistance(int x1,int y1,int x2,int y2)
    {
        double d=((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
        int res=(int)Math.ceil(Math.sqrt(d));
        return res;
    }
}
```

88) 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

```
public class Main {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
```

```

        String s=sc.nextLine();
        System.out.println(User.countWord(s));
    }
}

public class User {
    public static int countWord(String s)
    {
        StringTokenizer st=new StringTokenizer(s, " ");
        int count =st.countTokens();
        return count;
    }
}

```

89) 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

```

```

public class Main {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int a=sc.nextInt();
        int b=sc.nextInt();
        int c=sc.nextInt();
        System.out.println(User.getSumMaxMin (a,b,c));
    }
}

```

```

public class User {
    public static int getSumMaxMin (int a,int b,int c)
    {
        int sum=0;
        int[] s={a,b,c};
        Arrays.sort(s);
        sum=s[0]+s[2];
        return sum;
    }
}

```

90) 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

```

public class Main {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        System.out.println(User.convertDecimalToBinary(n));
    }
}

public class User {
    public static long convertDecimalToBinary(int n)
    {
        String x= Integer.toBinaryString(n);
        long res= Integer.parseInt(x);
    }
}

```

```

        long res=Integer.parseInt(Integer.toBinaryString(n));

    }

    return res;
}
}

```

91) 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:

llollo

```

public class Main {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        String s1=sc.next();
        int n=sc.nextInt();
        System.out.println(User.returnLastRepeatedCharacters (s1,n));
    }
}

public class User {
    public static String returnLastRepeatedCharacters (String s1,int n)
    {
        StringBuffer sb=new StringBuffer();
        for(int i=0;i<n;i++)

```

```

        sb.append(s1.substring(s1.length()-n));
        return sb.toString();
    }
}

```

92) 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

```

public class Main {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        String s1=sc.next();
        boolean b=User.validateString(s1);
        System.out.println(b);
    }
}

public class User {
    public static boolean validateString(String s1)
    {
        boolean b= false;
        if(!Character.isDigit(s1.charAt(0)))
            b= true;
        else
            b= false;
    }
}

```

```

        return b;
    }
}

```

93) 3String 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

```

public class Main {

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        String s1=sc.next();
        System.out.println(User.fetchUserName (s1));
    }
}

public class User {
    public static String fetchUserName (String s1)
    {
        StringBuffer sb=new StringBuffer();
        for( int i=0;i<s1.length();i=i+2)
            sb.append(s1.charAt(i));
        return sb.toString();
    }
}

```

```
    }
}
```

94) 2 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

Sample Output 1:

admin

```
public class Main {

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        String s1=sc.next();
        System.out.println(User.fetchUserName (s1));
    }
}

public class User {
    public static String fetchUserName (String s1)
    {
        boolean b=false;
        StringTokenizer st=new StringTokenizer(s1, "@");
        String name=st.nextToken();

        return name;
    }
}
```

95) 1 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

```
public class Main {  
  
    public static void main(String[] args) {  
        Scanner sc=new Scanner(System.in);  
        String s1=sc.next();  
        String s2=sc.next();  
        int n=sc.nextInt();  
        boolean b=User.isEqual(s1,s2,n);  
        System.out.println(b);  
    }  
}
```

```

        }

    }

public class User {
    public static boolean isEqual(String s1, String s2, int n)
    {
        boolean b=false;
        String i= s1.toLowerCase();
        String j= s2.toLowerCase();
        if(i.charAt(n-1)==j.charAt(n))
            b=true;
        else
            b=false;

        return b;
    }
}

```

96) 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

```
public class Main {  
    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={2,4,5,1,9,3,8};  
        System.out.println(getDiffArray(n1));  
    }  
}
```

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!

Solution :

```
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(User.printCapitalized(s));
    }
}

import java.util.StringTokenizer;

public class User {
    public static String printCapitalized(String s) {
        StringTokenizer st= new StringTokenizer(s, " ");
        StringBuffer sb= new StringBuffer();
        while(st.hasMoreTokens())
        {
            String s1=st.nextToken();
            String s2=s1.substring(0, 1);
            String s3=s1.substring(1);
            sb.append(s2.toUpperCase());
            sb.append(s3);
            sb.append(" ");
        }
        return sb.toString();
    }
}
```

```

    }
    return sb.toString();
}
}

import java.util.StringTokenizer;
public class UserMainCode {
    public static String changeWord(String s)
    {
        StringTokenizer st=new StringTokenizer(s, " ");
        StringBuffer sb=new StringBuffer();
        while(st.hasMoreTokens())
        {
            String s1=st.nextToken();
            sb.append(s1.substring(0,1).toUpperCase());
            sb.append(s1.substring(1));
            sb.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.]

Solution :

```
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.println(User.findMaxDistance(a));  
    }  
}  
  
  
public class User {  
    public static int findMaxDistance(int[] a) {  
  
        int dif,max=0;  
        int n=a.length;  
        for (int i=0;i<n-1;i++)  
        {  
            dif=Math.abs(a[i]-a[i+1]);  
  
            // if(max<dif)  
  
            if(dif>max)  
            {  
                if(a[i+1]>a[i])  
                    max=i+1;  
                else  
                    max=i;  
            }  
        }  
        return max;  
    }  
}
```

```
}
```

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)

Solution:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int a = sc.nextInt();
        int b = sc.nextInt();
```

```

System.out.println(User.addPalindromes(a,b));
}
}

public class User {
public static int addPalindromes(int a, int b) {

    int temp=0,sum=0,r,sum1=0;

    for (int i=a;i<=b;i++)
    {
        temp=i;
        sum=0;
    while(temp>0){
            r=temp%10;
            sum=sum*10+r;
            temp=temp/10;
        }
    if(i==sum)
        sum1=sum1+i;
    }
    return sum1;
}
}

```

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

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String s = sc.next();
        int res = User.validatePAN(s);
        if (res == 1)
            System.out.println("Valid");
        else
            System.out.println("Invalid");
    }
}

public class User {
    public static int validatePAN(String s) {
        int res = 0;
        if (s.length() == 8) {
            if (s.matches("[A-Z]{3}[0-9]{4}[A-Z]{1}"))
                res = 1;
            else
                res = 2;
        }
        return res;
    }
}
```

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**]

Solutions:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int s = sc.nextInt();

        System.out.println(User.getSumOfNfibos(s));
    }
}

public class User {
    public static int getSumOfNfibos(int s) {

        int a = 0, b = 1, c = 0, d = 1;
        for (int i = 3; i <= s; i++) {
            c = a + b;
            a = b;
            b = c;
            d = d + 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

Solutions:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String s=sc.next();
        int res=User.testVowels (s);
        if(res==1)
            System.out.println("Yes");
        else
            System.out.println("No");
    }
}

public class User {
    public static int testVowels (String s){

        int res,count=0;
        String s1="aeiou";
    }
}
```

```

String s2=s.toLowerCase();
for(int i=0;i<s2.length();i++)
{
    for(int j=0;j<s1.length();j++)
    {
        if(s2.charAt(i)==s1.charAt(j))
        {
            count++;
        }
    }
}

if(count==s1.length())
    res=1;
else
    res=2;
return res;
}
}

```

```

public class User {
public static int testOrderVowels(String s1) {

StringBuffer sb = new StringBuffer();
int res = 0;
for (int i = 0; i < s1.length(); i++) {
if (s1.charAt(i) == 'a' || s1.charAt(i) == 'A'
|| s1.charAt(i) == 'e' || s1.charAt(i) == 'E'
|| s1.charAt(i) == 'i' || s1.charAt(i) == 'I'
|| s1.charAt(i) == 'o' || s1.charAt(i) == 'O'
|| s1.charAt(i) == 'u' || s1.charAt(i) == 'U') {
sb.append(s1.charAt(i));
}
}
if (sb.toString().equals("aeiou"))
res = 1;
else
res = 0;
return res;
}
}

```

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

Solution:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String s1=sc.next();
        String s2=sc.next();
        int res=User.compareDashes (s1,s2);
        if(res==1)
            System.out.println("Yes");
        else
            System.out.println("No");
    }
}

import java.util.ArrayList;
```

```

public class User {
    public static int compareDashes (String s1, String s2) {

        int res=0;
        ArrayList<Integer> a1=new ArrayList<Integer>();
        ArrayList<Integer> a2=new ArrayList<Integer>();
        for(int i=0;i<s1.length();i++)
        {
            if(s1.charAt(i)=='-')
                a1.add(i);
        }
        for(int i=0;i<s2.length();i++)
        {
            if(s2.charAt(i)=='-')
                a2.add(i);
        }
        if(a1.equals(a2))
            res=1;
        else
            res=2;
        return res;
    }
}

```

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.HashMap;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String s1=sc.next();
        String s2=sc.next();
        System.out.println(User.extractMax(s1,s2));
    }
}

public class UserMain {
    public static String extractMax(String s1, String s2) {
        StringBuffer sb=new StringBuffer();
        for(int i=0;i<s1.length()-1;i++)
        {
            sb.append(s1.charAt(i));
            sb.append(s2);
        }
        sb.append(s1.charAt(s1.length()-1));
        return sb.reverse().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
2
0
0
```

Solution :

```
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();
        }
        User.removeTens(a);

    }
}

import java.util.ArrayList;

public class User {
    public static int[] removeTens(int[] a) {
        int[] out = new int[a.length];
        int k = 0;
        for (int i = 0; i < a.length; i++) {
            if (a[i] != 10) {
                out[k] = a[i];
                k++;
            }
        }
    }
}
```

```

for(int i=0;i<a.length;i++)
    System.out.println(out[i]);
return out;
}
}

```

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.

Smample Input :

This is a cat

Sample Output :

S\$\$A\$T

Solution :

```

import java.util.ArrayList;
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(User.getLastLetter(s));

    }
}

import java.util.StringTokenizer;

public class User {

```

```

public static String getLastLetter(String s){

    StringTokenizer st= new StringTokenizer(s, " ");
    String s2=st.nextToken();
    StringBuffer sb= new StringBuffer();
    String s3=Character.toUpperCase(s2.charAt(s2.length()-1))+"";
    while(st.hasMoreTokens())
    {
        s2=st.nextToken();
        s3=s3+$+Character.toUpperCase(s2.charAt(s2.length()-1));

    }
    returns3;
}

*****
*****



public class UserMain {
    public static String getLastLetter(String s)
    {
        StringTokenizer st= new StringTokenizer(s, " ");
        StringBuffer sb= new StringBuffer();

        String b=st.nextToken();
        sb.append(b.charAt(b.length()-1));

        while(st.hasMoreTokens())
        {
            String a=st.nextToken();
            sb.append("$");
            sb.append(a.charAt(a.length()-1));

        }
        return sb.toString().toUpperCase();
    }
}

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

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

Solutions:

```
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<Integer,String> hm= new HashMap<Integer,String>();
        for(int i=0;i<n;i++)
            hm.put(sc.nextInt(), sc.next());
        System.out.println(User.getMaxKeyValue(hm));
```

```

}

}

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
public static String getMaxKeyValue(HashMap<Integer, String> hm) {
    int max=0;
    String nn=null;
    Iterator<Integer> it = hm.keySet().iterator();
    while(it.hasNext())
    {
        int key=it.next();
        String name=hm.get(key);
        if(key>max)
        {
            key=max;
            nn=name;
        }
    }
    return nn;
}
}

```

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

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();  
  
        String[] s= new String[n];  
  
        for(int i=0;i<n;i++)  
  
            s[i]=sc.next();  
  
        int res=User.validateNumber(s);  
    }  
}
```

```

if(res==1)

    System.out.println("Valid");

else

    System.out.println("invalid");

}

}

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
public static int validateNumber(String s[]) {
    int res=0;
    int count=0,temp=0;
    String s1=null;
    for(int i=0;i<s.length;i++)
    {
        s1=s[i];
        count=0;
        for(int j=0;j<s1.length();j++)
        {

            if(s1.charAt(j)>='0' && s1.charAt(j)<='9' || s1.charAt(j)=='.')
                count++;
        }
        if(count==s1.length())
            temp++;
    }
    if(temp==s.length)
        res=1;
    else
        res=-1;
    return res;
}
}

```

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

Sample Output 1:

Friday

Solutions :

User :

```
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.Calendar;  
import java.util.Date;  
  
public class User  
{  
    public static String calculateBornDay(String d) throws ParseException  
    {  
        SimpleDateFormat sdf= new SimpleDateFormat("MM-dd-yyyy");  
        SimpleDateFormat s= new SimpleDateFormat("EEEE");  
        Date d1= new Date();
```

```
d1= sdf.parse(d);  
  
String day=s.format(d1);  
  
return day;  
}  
}
```

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

```
import java.util.HashMap;  
import java.util.Scanner;  
  
public class Main {  
    public static void main(String[] args) {
```

```

Scanner sc = new Scanner(System.in);
String s1=sc.next();
String s2=sc.next();
System.out.println(User.extractMax(s1,s2));

}

}

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
    public static String extractMax(String s1, String s2) {
        StringTokenizer st= new StringTokenizer(s1,s2);
        int max=0,c=0;
        String str=null;
        while(st.hasMoreTokens())
        {
            String s= st.nextToken();
            c=s.length();
            if(c>max)
            {
                max=c;
                str=s;
            }
        }
        return str;
    }
}

```

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

```
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++) {
            hm.put(sc.next(), sc.next());
        }
        String s = sc.next();
        System.out.println(User.getCapital(hm, s));
    }
}

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;
```

```

public class User {
    public static String getCapital(HashMap<String, String> hm, String s) {
        Iterator<String> it=hm.keySet().iterator();
        StringBuffer sb= new StringBuffer();
        while(it.hasNext())
        {
            String state=it.next();
            String cap=hm.get(state);
            if(state.equalsIgnoreCase(s))
            {
                sb.append(cap).append('$').append(state);
            }
        }
        return sb.toString().toLowerCase();
    }
}

```

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

```
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();
        System.out.println(User.calculateWordSum (s));

    }
}

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
    public static int calculateWordSum (String s){
        int sum=0,i=0;
        StringTokenizer st=new StringTokenizer(s, " ");
        String[] s1= new String[st.countTokens()];
        while(st.hasMoreTokens())
        {
            s1[i]=st.nextToken();
            i++;
        }
        if(s1[0].equals(s1[s1.length-1]))
            sum=s1[0].length();
        else
            sum=s1[0].length()+s1[s1.length-1].length();
        return sum;
    }
}
```

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 lowecase.

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
```

```
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();
String[] s= new String[n];
for(int i=0;i<n;i++)
    s[i]=sc.next();
System.out.println(User.matchCharacter(s));
}
```

```

}

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
    public static ArrayList<String> matchCharacter (String[] s) {
        ArrayList<String> a= new ArrayList<String>();
        for(int i=0;i<s.length;i++)
        {
            System.out.println(s[i].charAt(0));
            System.out.println(s[i].charAt(s[i].length()-1));
            if((s[i].charAt(0)=='a'|| s[i].charAt(0)=='e'|| s[i].charAt(0)=='i'|| s[i].charAt(0)=='o'|| s[i].charAt(0)=='u') && (s[i].charAt(s[i].length()-1)=='a'|| s[i].charAt(s[i].length()-1)=='e'|| s[i].charAt(s[i].length()-1)=='i'|| s[i].charAt(s[i].length()-1)=='o'|| s[i].charAt(s[i].length()-1)=='u'))
            {
                a.add(s[i]);
            }
        }
        return a;
    }
}

```

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

```
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);
```

```

int n=sc.nextInt();

HashMap<Integer,String> hm= new HashMap<Integer,String>();

ArrayList<String> a= new ArrayList<String>();

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

    hm.put(sc.nextInt(), sc.next());

a=User.getName(hm);

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

{

    System.out.println(a.get(i));

}

}

}

}

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
public static ArrayList<String> getName(HashMap<Integer, String> hm) {
    ArrayList<String> a= new ArrayList<String>();
    Iterator<Integer> it=hm.keySet().iterator();
    while(it.hasNext())
    {
        int id=it.next();
        String name=hm.get(id);
        for(int i=0;i<name.length();i++)
        {
            if(name.charAt(0)>=97 && name.charAt(0)<=122 &&
               name.charAt(name.length()-1)>=65 &&
               name.charAt(name.length()-1)<=96)
            {
                if(name.charAt(i)>='0' && name.charAt(i)<='9')
                {
                    a.add(name);
                }
            }
        }
    }
    return a;
}
}

```

```
        }
    }
}
}
return a;
}
}
```

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

```
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);
        int n=sc.nextInt();
        ArrayList<Integer> a= new ArrayList<Integer>();
        for(int i=0;i<n*2;i++)
            a.add(sc.nextInt());
        System.out.println(User.getYear(a));
    }
}

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
    public static int getYear(ArrayList<Integer> a) {
        int year=0;
```

```

int max=0;
for(int i=1;i<a.size();i=i+2)
{
    int x=a.get(i);
    if(x>max)
    {
        max=x;
        year=a.get(i-1);
    }
}
return year;
}

```

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

```
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);
        int n=sc.nextInt();
        System.out.println(User.addNumbers(n));
    }
}

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
    public static int addNumbers(int n) {
        int c=0,sum=0;
        for (int i = 1; i <= n; i++)
        {
            c=0;
            for(int j=1;j<=i;j++)
            {
                if(i%j==0)
                    c++;
                if (c==2)
                    ;
                else
                    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

```
import java.text.ParseException;  
  
import java.util.ArrayList;  
  
import java.util.HashMap;  
  
import java.util.Scanner;  
  
  
public class Main {  
public static void main(String[] args) throws ParseException {  
Scanner sc = new Scanner(System.in);  
String n=sc.next();  
System.out.println(User.convertDateFormat(n));  
}
```

```

}

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
    public static String convertDateFormat(String n) throws ParseException{
        SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
        Date d= sdf.parse(n);
        SimpleDateFormat sdf1=new SimpleDateFormat("dd-MM-yyyy");
        String s=sdf1.format(d);
        return s;
    }
}

```

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

```
public static void main(String[] args) {  
  
    Scanner sc=new Scanner(System.in);  
    String s=sc.nextLine();  
    SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");  
    sdf.setLenient(false);  
    int res=0;  
    if(s.matches("[0-9]{2}(/)[0-9]{2}(/)[0-9]{4}"))  
    {  
  
        try {  
            Date d=sdf.parse(s);  
            res=1;  
        } catch (ParseException e) {  
            res=-1;  
        }  
  
        System.out.println(res);  
    }  
}
```

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

```
import java.text.ParseException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) throws ParseException {
        Scanner sc = new Scanner(System.in);
        String n=sc.next();
        System.out.println(User.convertFormat(n));
    }
}

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
    public static String convertFormat(String s) throws ParseException{
        StringTokenizer st=new StringTokenizer(s,"-");
        int i=0;
        String[] s1=new String[st.countTokens()];
        while(st.hasMoreTokens())
        {
            s1[i]=st.nextToken();
            i++;
        }
        StringBuffer sb=new StringBuffer();
        sb.append(s1[0].substring(0,2));
        sb.append("-");
        sb.append(s1[0].substring(2)).append(s1[1].substring(0,1));
        sb.append("-");
        sb.append(s1[1].substring(1)).append(s1[2].substring(0,1));
        sb.append("-");
        sb.append(s1[2].substring(1));
    }
}
```

```

        return sb.toString();
    }

}

import java.util.StringTokenizer;
public class UserMainCode {
    public static String convertFormat(String s)
    {
        StringBuffer sb=new StringBuffer();
        StringTokenizer st=new StringTokenizer(s,"-");
        String s1=st.nextToken();
        String s2=st.nextToken();
        String s3=st.nextToken();
        sb.append(s1.substring(0,2));
        sb.append("-");
        sb.append(s1.substring(s1.length()-1));
        sb.append(s2.substring(0,1));
        sb.append("-");
        sb.append(s2.substring(1));
        sb.append(s3.substring(0,1));
        sb.append("-");
        sb.append(s3.substring(1));
        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

```

import java.text.ParseException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) throws ParseException {
        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 x=sc.nextInt();
        System.out.println(User.addAndReverse(a,x));
    }
}

```

```

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
    public static int addAndReverse(int[] a, int x) {
        int sum=0;
        int rev=0, r=0;
        for(int i=0;i<a.length;i++) {
            if(x<a[i])
                sum=sum+a[i];
        }

        while(sum!=0)
        {
            r=sum%10;
            rev=rev*10+r;
            sum=sum/10;
        }

        return rev;
    }
}

```

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

```
import java.text.ParseException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Scanner;

public class Main {
    public static void main(String[] args) throws ParseException {
        Scanner sc = new Scanner(System.in);
        String n=sc.next();

        System.out.println(User.nextYearDay(n));

    }
}

import java.text.ParseException;
```

```

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
    public static String nextYearDay(String s) throws ParseException {
        SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
        SimpleDateFormat sdf1=new SimpleDateFormat("EEEE");
        Date d= sdf.parse(s);
        Calendar c=Calendar.getInstance();
        c.setTime(d);
        c.add(Calendar.YEAR,1);
        Date year=c.getTime();
        String day=sdf1.format(year);

        return day;
    }
}

```

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

```
import java.text.ParseException;
```

```

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

public class Main {
    public static void main(String[] args) throws ParseException {
        Scanner sc = new Scanner(System.in);
        int n=sc.nextInt();

        System.out.println(User.getSumOfSquaresOfDigits(n));

    }
}

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
    public static int getSumOfSquaresOfDigits(int n) {
        int sum=0,r=0;
        while(n!=0)
        {
            r=n%10;
            sum=sum+(r*r);
            n=n/10;
        }
        return 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

```
import java.text.ParseException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) throws ParseException {
        Scanner sc = new Scanner(System.in);
        int n=sc.nextInt();

        int res=User.sumOfOddEvenPositioned(n);
        if(res==1)
            System.out.println("yes");
        else
            System.out.println("no");

    }
}
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
```

```

import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
    public static int sumOfOddEvenPositioned(int n) {
        int even=0,odd=0;
        int res=0,r=0,m=0;
        int j=0,count=0;
        int n1=n;
        while(n>0)
        {
            n=n/10;
            count++;
        }
        System.out.println(count);
        int[] a=new int[count];
        while(n1!=0)
        {
            r=n1%10;
            a[j]=r;
            j++;
            n1=n1/10;
        }
        int[] b=new int[j];
        for(int k=j-1;k>=0;k--)
        {
            b[m]=a[k];
            m++;
        }

        for(int i=0;i<m;i++)
        {
            System.out.println("a:"+b[i]);
            if(i%2==0)
                even=even+b[i];
            else
                odd=odd+b[i];
        }
        System.out.println(even);
        System.out.println(odd);
        if(even==odd)
            res=1;
        else
            res=-1;
        return res;
    }
}

```

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
```

```
publicclass Main {
publicstaticvoid main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);
int n=sc.nextInt();
ArrayList<Integer> a=new ArrayList<Integer>();
ArrayList<Integer> res=new ArrayList<Integer>();
for(int i=0;i<n;i++)
```

```

        a.add(sc.nextInt());
res=User.removeMultiplesOfThree(a);
for(int i=0;i<res.size();i++)
    System.out.println(res.get(i));
}
}

public class User {
public static ArrayList<Integer> removeMultiplesOfThree(ArrayList<Integer> a)
{
    ArrayList<Integer> b=new ArrayList<Integer>();
    for(int i=0;i<a.size();i++)
    {
        int d=a.get(i);
        if(d%3!=0)
        {
            b.add(a.get(i));
        }
    }
    return b;
}
}

```

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  
import java.text.ParseException;  
  
import java.util.ArrayList;  
  
import java.util.HashMap;  
  
import java.util.Iterator;  
  
import java.util.Scanner;
```

```
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(User.getSubstring(s1, s2));  
  
    }  
  
}
```

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
    public static int getSubstring (String s1,String s2) {
        int count=0;
        int n=s1.length()-(s2.length()-1);
        for(int i=0;i<n;i++)
        {
            String s3=s1.substring(i,i+(s2.length()));
            if(s2.equals(s3))
                count++;
        }
        return count;
    }
}
```

```

public class User {
    public static int getSubstring (String s1, String s2) {
        int count=0;
        int n=s1.length()-(s2.length()-1);
        int s2l=s2.length();
        System.out.println(n);
        for(int i=0;i<n;i++)
        {
            String s3=s1.substring(i,i+s2l);
            if(s2.equals(s3))
                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

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;
```

```
public class User {
    public static int getLuckySum (int a,int b, int c) {
```

```
int res=0;
if(a==13)
    res=c;
else if(b==13)
    res=a;
else if(c==13)
    res=a+b;
else
    res=a+b+c;

return res;

}
```

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

```
import java.text.ParseException;  
  
import java.util.ArrayList;  
  
import java.util.HashMap;  
  
import java.util.Iterator;  
  
import java.util.Scanner;  
  
  
public class Main {
```

```
public static void main(String[] args) throws ParseException {
    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();
    boolean b=User.checkTriplets(a);
    System.out.println(b);
}
}

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;
```

```
public class User {
    public static boolean checkTriplets (int a[]) {
        boolean b=false;
        int c=0;
```

```

for(int i=0;i<a.length-2;i++)
{
    if(a[i]==a[i+1]&&a[i+1]==a[i+2])
        b=true;
    else
        b=false;
}
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

```
import java.text.ParseException;
```

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

```
import java.util.HashMap;
```

```
import java.util.Iterator;
```

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

```
public class Main {
```

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

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

```
        String s=sc.next();
```

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

```
        String res=User.repeatFirstThreeCharacters(s,n);
```

```
for(int i=0;i<n;i++)  
    System.out.print(res);  
  
}  
}
```

```
import java.text.ParseException;  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.Calendar;  
import java.util.Date;  
import java.util.HashMap;  
import java.util.Iterator;  
import java.util.StringTokenizer;
```

```
public class User {  
    public static String repeatFirstThreeCharacters(String s, int n) {  
  
        String front=null;  
        if(s.length()>=3)  
        {  
            front=s.substring(0,3);
```

```
    }  
  
    else  
  
        front=s;  
  
    return front;  
  
}  
  
}
```

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

```
import java.text.ParseException;
```

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) throws ParseException {
        Scanner sc = new Scanner(System.in);
        int n=sc.nextInt();
        String[] a= new String[n];
        for(int i=0;i<n;i++)
            a[i]=sc.next();

        String res[]={};
        res=User.orderElements(a);
        for(int i=0;i<res.length;i++)
            System.out.println(res[i]);
    }
}

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collections;
```

```
import java.util.Date;  
import java.util.HashMap;  
import java.util.Iterator;  
import java.util.LinkedHashSet;  
import java.util.StringTokenizer;  
  
public class User {  
  
    public static String[] orderElements(String[] s) {  
  
        LinkedHashSet<String> lhs=new LinkedHashSet<String>();  
  
        for(int i=0;i<s.length;i++)  
        {  
            lhs.add(s[i]);  
        }  
  
        String[] a= new String[lhs.size()];  
  
        for(int i=0;i<s.length;i++)  
        {  
            lhs.toArray(a);  
        }  
  
        Arrays.sort(a);  
  
        return a;  
    }  
}
```

}

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

```
import java.text.ParseException;  
  
import java.util.ArrayList;  
  
import java.util.HashMap;
```

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

public class Main {
    public static void main(String[] args) throws ParseException {
        Scanner sc = new Scanner(System.in);
        String a= sc.next();

        boolean b=User.CheckID(a);

        System.out.println(b);
    }
}

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collections;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedHashSet;
import java.util.StringTokenizer;
```

```
public class User {  
  
    public static boolean CheckID (String s) {  
  
        boolean b=false;  
  
        if(s.matches("(CPT-)[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$
```

```
import java.text.ParseException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) throws ParseException {
        Scanner sc = new Scanner(System.in);
        int n=sc.nextInt();
        String[] a=new String[n];
        for(int i=0;i<n;i++)
            a[i]=sc.next();
        int s=sc.nextInt();
    }
}
```

```
System.out.println(User.formString(a,s));

}

}

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collections;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedHashSet;
import java.util.StringTokenizer;

public class User {
    public static String formString(String s[],int n) {
        StringBuffer sb=new StringBuffer();
        for(int i=0;i<s.length;i++)
        {
            String st=s[i];
            if(st.length()>=n)
```

```

    {
        sb.append(st.charAt(n-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

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

```
public class Main {
```

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

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

```
        String s=sc.next();
```

```
        System.out.println(User.validate (s));
```

```

}

}

public class User {

    public static boolean validate (String s) {

        boolean b= false;

        if(s.length()==4)

        {

            if(s.matches("[a-zA-Z0-9]{1}(R)[0-9]{1}[A-Za-z0-9]{1}"))

                b=true;

            else

                b=false;

        }

        return b;

    }

}

```

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

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

```
public class Main {
```

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

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

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

System.out.println(User.validate (s));

}

}
```

```
public class User {

public static boolean validate (int s) {

boolean b= false;

if(s>0)

{

if(s>=21&&s<=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

```
import java.text.ParseException;  
  
import java.util.ArrayList;  
  
import java.util.HashMap;  
  
import java.util.Iterator;  
  
import java.util.Scanner;
```

```
public class Main {  
  
    public static void main(String[] args) throws ParseException {  
  
        Scanner sc = new Scanner(System.in);  
  
  
        String s=sc.next();  
  
        System.out.println(User.validatePhone(s));  
  
  
    }  
  
}
```

```
public class User {  
  
    public static boolean validatePhone(String s) {  
  
        boolean b= false;  
  
        if(s.length()==10)  
  
        {  
  
            if(s.matches("(0){2}[0-9]{8}"))  
  
                b=false;  
  
            else if(s.matches("[0-9]{10}"))  
  
                b=true;  
  
            else  
  
                ;  
  
        }  
  
        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

```
publicclass Main {  
publicstaticvoid main(String[] args) throws ParseException {  
Scanner sc = new Scanner(System.in);  
  
String s1=sc.next();  
char s2=sc.next().charAt(0);
```

```

String res[] = User.manipulateLiteral (s1,s2);
for(int i=0;i<res.length;i++)
    System.out.println(res[i]);

}
}

public class User {
public static String[] manipulateLiteral(String s1,char s2) {
    String ss=Character.toString(s2);
    StringTokenizer st=new StringTokenizer(s1,ss);
    ArrayList<String> a=new ArrayList<String>();
    while(st.hasMoreTokens())
    {
        StringBuffer sb=new StringBuffer();
        sb.append(st.nextToken().toLowerCase());
        a.add(sb.reverse().toString());
    }
    String[] s=new String[a.size()];
    for(int i=0;i<a.size();i++)
        s[i]=(String)a.get(i);

    return s;
}
}

```

```

import java.util.ArrayList;
import java.util.StringTokenizer;

public class User {
public static String[] manipulateLiteral(String s1,char s2) {
    String ss=String.valueOf(s2);
    StringTokenizer st=new StringTokenizer(s1,ss);
    ArrayList<String> a=new ArrayList<String>();

    while(st.hasMoreTokens())
    {
        StringBuffer sb=new StringBuffer();
        sb.append(st.nextToken());
        a.add(sb.reverse().toString().toLowerCase());
    }
    int d=a.size();
    System.out.println(d);
    String[] s=new String[d];
    for(int i=0;i<a.size();i++)
    {
        s[i]=a.get(i);
    }
}

```

```
    return s;
}
}
```

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.Scanner;

public class Main {
    public static void main(String[] args) throws ParseException {
        Scanner sc = new Scanner(System.in);

        String s1=sc.next();

        System.out.println(User.tellVowelCount(s1));
    }
}

public class User {
    public static int tellVowelCount(String s1) {
        int count=0;
        String s="aeiou";
        String ss="AEIOU";
        for(int i=0;i<s1.length();i++)
        {
            for(int j=0;j<s.length();j++)
            {
                if(s1.charAt(i)==s.charAt(j) || s1.charAt(i)==ss.charAt(j))
)
                    count++;
            }
        }
        return count;
    }
}
```

```
public static int tellVowelCount(String s1) {
    int count=0;
    for(int i=0;i<s1.length();i++)
    {
        if(s1.charAt(i)=='a' || s1.charAt(i)=='e' ||
s1.charAt(i)=='i'
            || s1.charAt(i)=='o' || s1.charAt(i)=='u' ||
s1.charAt(i)=='A' || s1.charAt(i)=='E' ||
s1.charAt(i)=='I' ||
s1.charAt(i)=='O' || s1.charAt(i)=='U' )

        {
            count++;
        }
    }
    return count;
}
}
```

```
*****
```

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
```

```
publicclass Main {
publicstaticvoid main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);
int n=sc.nextInt();
String[] a= new String[n];
for(int i=0;i<n;i++)
    a[i]=sc.next();
String[] res=User.sortArray(a);
for(int i=0;i<res.length;i++)
    System.out.println(res[i]);
```

```

}
}

public class User {
    public static String[] sortArray (String s[]) {
        String[] a=new String[s.length];
        for(int i=0;i<s.length;i++)
        {
            a[i]=s[i].toLowerCase();
        }
        Arrays.sort(a);
        return a;
    }
}

```

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
```

```
publicclass Main {
publicstaticvoid main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);
int n=sc.nextInt();
int[] a= newint[n];
for(int i=0;i<n;i++)
    a[i]=sc.nextInt();
System.out.println(User.calculateMedian (a));
}
```

```
publicclass User {

publicstaticint calculateMedian(int s[]) {
    double med=0;
    double avg=0;
Arrays.sort(s);
int mid=s.length/2;
if(s.length%2!=0)
    med=s[mid];
else
{
```

```

        avg=(double) (s[mid]+s[mid-1])/2;
        System.out.println(avg);
        med=Math.ceil(avg);
    }
    return (int)med;
}
}

```

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

```
publicclass Main {
publicstaticvoid main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);
int n=sc.nextInt();
int[] a= newint[n];
for(int i=0;i<n;i++)
a[i]=sc.nextInt();
boolean b=User.calculateMedian (a);
System.out.println(b);

}

publicclass User {
publicstaticboolean calculateMedian(int s[]) {
    int[] a={1,2,3};
    int n=s.length-(a.length-1);
    boolean b=false;
    for(int i=0;i<n;i++)
    {
        if(s[i]==a[0] )
        {
            if(s[i+1]==a[1])
            {
                if(s[i+2]==a[2])
                {
                    b=true;
                    break;
                }
                else
                    b=false;
            }
            else
                b=false;
        }
        else
            b=false;
    }
    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

```
publicclass Main {  
publicstaticvoid main(String[] args) throws ParseException {  
Scanner sc = new Scanner(System.in);  
String n=sc.next();  
  
boolean b=User.scanStarNeighbors(n);  
System.out.println(b);  
}  
}
```

```

public class User {
    public static boolean scanStarNeighbors(String s) {
        StringTokenizer st=new StringTokenizer(s, "*");
        boolean b=false;
        while(st.hasMoreTokens())
        {
            String s1=st.nextToken();
            String s2=st.nextToken();
            if(s1.charAt(s1.length()-1)==s2.charAt(0))
            {
                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

```
publicclass Main {
publicstaticvoid main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);
String s1=sc.nextLine();
String s2=sc.next();
int b=User.countWords (s1,s2);
System.out.println(b);

}

publicclass User {
publicstaticint countWords (String s1,String s2) {
    StringTokenizer st=new StringTokenizer(s1, " ");
    int c=0;
    while(st.hasMoreTokens ())
    {
        String s3=st.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:

2

Sample Input 2:

Catcowcat

catp

Sample Output 2:

0

```
public class User {  
    public static int scanStarNeighbors(String s1, String s2) {  
        int ls1=s1.length();  
        int ls2=s2.length();  
        int n=ls1-(ls2-1);  
        System.out.println(n);  
        int ct=0;  
        for(int i=0;i<n;i++)  
        {  
  
            String ss=s1.substring(i,i+(ls2));  
            if(s2.equals(ss))  
                ct++;  
        }  
  
        return ct;  
    }  
}
```

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 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

```
import java.text.ParseException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
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();
int n=sc.nextInt();
System.out.println(User.findFruitName(s1,n));
}

import java.util.ArrayList;
import java.util.Arrays;
import java.util.LinkedHashSet;
import java.util.StringTokenizer;
public class User {
public static String findFruitName(String s1,int n) {
    StringTokenizer st=new StringTokenizer(s1,",");
    int c=0,i=0;
    String ss=null;

    String[] s=new String[st.countTokens()];

    while(st.hasMoreTokens()) {
        s[i]=st.nextToken();
        i++;
    }
    if(i>n)
    {
        ss=s[n-1];
    }
    else
    {
        ss=s[i-1];//last element display
    }

    return ss;
}
}

```

48. Proper 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

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

```
import java.util.StringTokenizer;
public class UserMainCode {
    public static String changeWord(String s)
    {
        StringTokenizer st=new StringTokenizer(s, " ");
        StringBuffer sb=new StringBuffer();
        while(st.hasMoreTokens())
        {
            String s1=st.nextToken();
            sb.append(s1.substring(0,1).toUpperCase());
            sb.append(s1.substring(1));
            sb.append(" ");
        }
        return sb.toString();
    }
}
```

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 $1+2+5$ is not equal to 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

```
publicclass User {
    publicstaticboolean scanStarNeighbors(intn) {
        boolean b;

        int sum=0;

        for(int i=1;i<n;i++)
        {
            if(n%i==0)
            {
                sum=sum+i;
                System.out.println(sum);
            }
        }
        if(sum==n)
        {
            b=true;
        }
        else
        {
            b=false;
        }

        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

Sample Output 3:

2:2

```
import java.util.*;  
  
public class Main {  
  
    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;  
    }  
}
```

```

}

import java.util.StringTokenizer;
public class User{
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(n2-1)=='0')
n2=s2.length()-1;
//String s3=String.valueOf(n1)+":"+String.valueOf(n2);
StringBuffer sb=new StringBuffer();
sb.append(n1).append(":").append(n2);
return sb.toString();
}
}

```

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 and 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
```

```
class Main
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        sc.nextLine();
        HashMap<String,String> hm=new HashMap<String,String>();
    }
}
```

```
for(int i=0;i<n;i++)  
{  
    hm.put(sc.nextLine(),sc.nextLine());  
}  
  
String b=sc.nextLine();  
  
  
HashMap<String,String> op=new HashMap<String,String>();  
op=MainClass.obtainDesig(hm,b);  
Iterator<String> itr=op.keySet().iterator();  
while(itr.hasNext())  
{  
    String key=itr.next();  
    System.out.println(key);  
    String value=hm.get(key);  
    System.out.println(value);  
}  
}  
}}
```

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedHashMap;

public class MainClass {

    public static LinkedHashMap<String, String> obtainDesig(HashMap<String, String> hm, String s)

    {
        LinkedHashMap<String, String> op=new LinkedHashMap<String, String>();
        Iterator<String> itr=hm.keySet().iterator();

        while(itr.hasNext())
        {
            String key=itr.next();
            String value=hm.get(key);
            if(s.equals(value))
            {
                op.put(key,value);
            }
        }

        return op;
    }
}
```

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.LinkedHashMap;
```

```
import java.util.Map;
```

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

```
public class Main
```

```
{
```

```
    public static void main(String[]arg)
```

```
{
```

```
        LinkedHashMap<String,Double>hm=new LinkedHashMap<String,Double>();
```

```
        LinkedHashMap<String,String>hm1=new LinkedHashMap<String,String>();
```

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

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

```
        for(int i=0;i<n;i++)
```

```
{
```

```
            String s=sc.next();
```

```
            double d=sc.nextDouble();
```

```
hm.put(s,d);

}

LinkedHashMap<String,String>hm2=UserMainCode.dis(hm);

for(Map.Entry<String,String>entry:hm2.entrySet())

{

System.out.println(entry.getKey());

System.out.println(entry.getValue());

}}}
```

```
import java.util.LinkedHashMap;

import java.util.Map;

class UserMainCode

{

public static LinkedHashMap<String,String>dis(LinkedHashMap<String,Double>h1)

{

    LinkedHashMap<String,String>h2=new LinkedHashMap<String,String>();

    for(Map.Entry m:h1.entrySet())

    {

        double d=(Double)m.getValue();

        if(d>60)

    {
```

```

String s=(String)m.getKey();

h2.put(s,"pass");

}

else

{

String s=(String)m.getKey();

h2.put(s,"fail");

}

}

return h2;

}

}

(Or)

```

```

import java.util.*;
public class Main {
public static void main(String[] args) {
    Scanner sc=new Scanner(System.in);

    int n=sc.nextInt();
    LinkedHashMap<String,Float> ip=new LinkedHashMap<String,Float>();
    for(int i=0;i<n;i++)
    {
        ip.put(sc.next(),sc.nextFloat());
    }

    LinkedHashMap<String,String> op=new LinkedHashMap<String,String>();
    op=User.noOfDigits(ip);
    Iterator<String> itr= op.keySet().iterator();
    while(itr.hasNext())
    {
        String key=itr.next();
        System.out.println(key);
        String value=op.get(key);
        System.out.println(value);

    }
}
}

```

```

import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedHashMap;
public class User{
public static LinkedHashMap<String, String> noOfDigits(HashMap<String, Float>
hm) {

    LinkedHashMap<String, String> op=new LinkedHashMap<String, String>();
    Iterator<String> itr=hm.keySet().iterator();
    String res=null;
        for(int i=0;i<hm.size();i++)
        {
    while(itr.hasNext())
    {
        String key=itr.next();

        float value=hm.get(key);
        if(value>=60)
            res="pass";
        else
            res="fail";
        op.put(key,res);

    }
    }

    return op;
}
}

import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedHashMap;
public class User{
public static LinkedHashMap<String, String> noOfDigits(HashMap<String, Float>
hm) {

    LinkedHashMap<String, String> op=new LinkedHashMap<String, String>();
    Iterator<String> itr=hm.keySet().iterator();
    String res=null;
        while(itr.hasNext())
    {
        String key=itr.next();

        float value=hm.get(key);
        if(value>=60)
            res="pass";
        else
            res="fail";
        op.put(key,res);

    }
}


```

```
    return op;
}
}
```

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 UserMainCode {
    public static void main(String[] args)
    {
        String str=new String();
        Scanner sc=new Scanner(System.in);
        str=sc.nextLine();
        SimpleDateFormat sdf=new SimpleDateFormat("MM/dd/yyyy");
        sdf.setLenient(false);
        try
        {
            Date d1=sdf.parse(str);
            System.out.println("TRUE");
        }
        catch(Exception e)
        {
            System.out.println("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 Format:

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.print(UserMainCode.getvalues(s,s1));
    }
}

import java.util.Calendar;

import java.util.Date;

public class UserMainCode {
    public static boolean getvalues(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.ArrayList;  
  
import java.util.Scanner;  
  
  
class Main  
{  
  
    public static void main(String[] arg)  
    {  
  
        Scanner sc=new Scanner(System.in);  
  
        int n=sc.nextInt();  
  
        sc.nextLine();  
  
        ArrayList<String> aa=new ArrayList<String>();
```

```
for(int i=0;i<n;i++)
{
    aa.add(sc.nextLine());
}

String a[]=MainClass.convertToString( aa);

for(int i=0;i<a.length;i++)
{
    System.out.println(a[i]);
}

}

import java.util.ArrayList;
import java.util.Collections;

public class MainClass {

    public static String[] convertToString(ArrayList<String> a1)
    {
        Collections.sort(a1);// uses to sort arraylist string
    }
}
```

```

String a[]=new String[a1.size()];

a1.toArray(a);

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 **getStatId** 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

```
import java.util.*;
public class Main
{
    public static void main(String[] args)
    {

        Scanner sc=new Scanner(System.in);

        int n=sc.nextInt();
        String s1[]={};
        for(int i=0;i<n;i++)
        {
            s1[i]=sc.next();
        }

        LinkedHashMap<String, String> ip=new
        LinkedHashMap<String, String>();
        ip=User.Method(s1);
        Iterator<String> itr=ip.keySet().iterator();

        //while(itr.hasNext())
        for(int i=0;i<ip.size();i++)
        {
            String key=itr.next();
            String value=ip.get(key);
            System.out.println(value+":"+key);
        }

    }
}

import java.util.LinkedHashMap;
public class User
{
    public static LinkedHashMap<String, String> Method(String[] s1)
    {
        LinkedHashMap<String, String> op=new LinkedHashMap<String, String>();

        for(int i=0;i<s1.length;i++)
        {
            StringBuffer sb=new StringBuffer();
            StringBuffer key=sb.append(s1[i].substring(0,3)).append(":");
            op.put(s1[i],key.toString().toUpperCase());
        }
    }
}
```

```
    return op;
}
}
```

(or)

STATE id

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.HashMap;

public class Main {

    public static void main(String[] args) {
        BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
        try
        {
            int n=Integer.parseInt(br.readLine());
            String[] input=new String[n];
            for(int i=0;i<n;i++)
            {
                input[i]=br.readLine();
            }
            HashMap<String, String>hm=UserMainCode.costEst
imator(input);
            for(int i=0;i<n;i++)
            {
                String s=input[i];
                String key=hm.get(s);
                System.out.println(key+":"+s);
            }
        }
        catch(Exception e)
        {
        }
    }
}
```

```

}

import java.util.HashMap;

public class UserMainCode {
    public static HashMap<String, String>
costEstimator(String[] name)
    {
        int n=name.length;
        HashMap<String, String> hm=new HashMap<String,
String>();
        for(int i=0;i<n;i++)
        {
            String sub=name[i].substring(0, 3);
            hm.put(name[i], sub.toUpperCase());
        }
        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
```

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        List<String> l1=new ArrayList<String>();
        l1.add("Apple");
        l1.add("Chery");
        l1.add("Grapes");
        List<String> l2=new ArrayList<String>();
        l2.add("Orange");
        l2.add("Mango");
        l2.add("Melon");
        l2.add("Apple");
        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(s1.length()-1)!= 'a' && s1.charAt(s1.length()-1)!= 'A' &&
s1.charAt(s1.length()-1)!= 'e' && s1.charAt(s1.length()-1)!= 'E')
            l3.add(s1); }

        for(int i=0;i<l2.size();i++){
            String s1=l2.get(i);
            if(s1.charAt(0)!= 'm' && s1.charAt(0)!= 'M' && s1.charAt(0)!= 'a' &&
s1.charAt(0)!= 'A')
                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;
    }
}

```

```

import java.util.*;
public class Main {
public static void main(String[] args) {
    List<String> l1=new ArrayList<String>();
    Scanner sc=new Scanner(System.in);
    int n=sc.nextInt();
    for(int i=0;i<n;i++)
    {
        l1.add(sc.next());
    }
    List<String> l2=new ArrayList<String>();

    int n1=sc.nextInt();
    for(int i=0;i<n1;i++)
    {
        l2.add(sc.next());
    }
}

```

```

        String[] s2=User.fruitsList(l1,l2);
        // for(int i=0;i<s2.length;i++)
        //   System.out.println(s2[i].toString());
    for(String s3:s2)
        System.out.println(s3);

    }
}

publicclass User
{
publicstatic String[] fruitsList(List<String> l1, List<String> l2) {
    ArrayList<String> l3=new ArrayList<String>();
    for(int i=0;i<l1.size();i++)
    {
        String s1=l1.get(i);
        int len=s1.length();
        if(s1.charAt(len-1]!='a'&& s1.charAt(len-1)!='A'
           && s1.charAt(len-1)!='e'&& s1.charAt(len-1)!='E')

            l3.add(s1);
    }

    for(int i=0;i<l2.size();i++)
    {
        String s1=l2.get(i);
        if(s1.charAt(0)!='m'&& s1.charAt(0)!='M'&& s1.charAt(0)!='a'
           && s1.charAt(0)!='A')
            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;
}

}

]

/*

```

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
```

```
import java.util.*;
public class Main
{
    public static void main(String[] args)
    {
        int n,m;
        Scanner sc = new Scanner(System.in);
        n = sc.nextInt();
        ArrayList<Integer> a1 = new ArrayList<Integer>();
        for(int i=0;i<n;i++)
        {

            a1.add(sc.nextInt());
        }
        m = sc.nextInt();
        ArrayList<Integer> a2 = new ArrayList<Integer>();
        for(int i=0;i<m;i++)
        {

            a2.add(sc.nextInt());
        }
        int[] result = User.arrayListSubtractor(a1, a2);
        Arrays.sort(result);
        for(int i=0;i<result.length;i++)
            System.out.println(result[i]);
    }
}
```

```
import java.util.*;

public class User
{

    public static int[] arrayListSubtractor(ArrayList<Integer>
arrlist1,ArrayList<Integer> arrlist2)
    {
        TreeSet<Integer> ts1=new TreeSet<Integer>();
        TreeSet<Integer> ts2=new TreeSet<Integer>();
        TreeSet<Integer> ts3=new TreeSet<Integer>();
        ArrayList<Integer> aa=new ArrayList<Integer>();
        for(int i=0;i<arrlist1.size();i++)
            ts1.add(arrlist1.get(i));

        for(int i=0;i<arrlist2.size();i++)
            ts2.add(arrlist2.get(i));

        ts1.addAll(ts2);

        for(int i=0;i<arrlist1.size();i++)
        {
            for(int j=0;j<arrlist2.size();j++)
            {
                if(arrlist1.get(i)==arrlist2.get(j))
                    ts3.add(arrlist1.get(i));
            }
        }
        ts1.removeAll(ts3);
        aa.addAll(ts1);
        int res[]={new int[aa.size()]};
        for(int i=0;i<res.length;i++)
            res[i]=aa.get(i);
        return res;
    }

}
```

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 calculate 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

```
import java.util.*;  
  
public class UserMainCode {  
  
    public static void main(String[] args) {  
  
        HashMap<String, String> m1=new HashMap<String, String>();  
  
        m1.put("monitor", "1200.36");  
  
        m1.put("mouse","100.42");  
  
        m1.put("speaker", "500.25");  
  
        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;

}}
```

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

```
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 User {

    public static void main(String[] args) throws IOException, ParseException {
        // TODO Auto-generated method stub

        String s1="10-02-2012";
        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

```

import java.text.ParseException;

import java.util.*;
public class Main
{
    public static void main(String[] args) throws ParseException {
        Scanner sc=new Scanner(System.in);
        String s=sc.nextLine();
        System.out.println(User.leapYear(s));
    }
}

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

public class User
{
    public static boolean leapYear(String s) throws ParseException {
        SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
        GregorianCalendar g=new GregorianCalendar();

        Calendar cal=Calendar.getInstance();
        Date d1=sdf.parse(s);
        cal.setTime(d1);
        int n=cal.get(Calendar.YEAR);

        boolean b=g.isLeapYear(n);
        return 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

```
import java.util.*;  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        String s1="You are toooo 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;  
    }  
}
```

```
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
```

```
public class Main {  
    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);  
    }  
}
```

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

```
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.display(a));
    }
}
```

```
import java.util.Iterator;
import java.util.LinkedHashSet;

public class UserMainCode {
    public static int display(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

```

public class UserMainCode

{
    public static int display(String s)

    {
        if(s.matches(".*[0-9]{1,}.*") && s.matches(".*[@#$]{1,}.*") && s.length()>=8 &&
s.matches(".*[A-Z]{1,}.*") && s.matches(".*[a-z]{1,}.*"))

            return 1;

        else

            return -1;
    }

}

import java.util.*;
public class Main
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        String s=sc.nextLine();
        System.out.println(User.leapYear(s));
    }
}

public class User{
    public static int leapYear(String s)
    {

if(s.matches(".*[0-9]{1,}.*")
    &&s.matches(".*[@#$%^&]{1,}.*") &&s.length()>=8 &&
s.matches(".*[A-Z]{1,}.*") &&s.matches(".*[a-z]{1,}.*"))

return 1;
else
return -1;
    }
}

```

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
```

```
import java.util.HashMap;  
  
import java.util.Iterator;  
  
import java.util.LinkedHashMap;  
  
  
import java.util.Scanner;
```

```
public class kapes3 {  
  
    public static void main(String []args){  
  
        Scanner sc=new Scanner(System.in);  
  
        int s=Integer.parseInt(sc.nextLine());  
  
        int []a=new int[s];  
  
        for(int i=0;i<s;i++)  
  
        {  
  
            a[i]=sc.nextInt();  
  
        }  
  
        LinkedHashMap<Integer,Integer>hm2=new LinkedHashMap<Integer,Integer>();  
  
        hm2=kapes4.display(a);  
  
        Iterator<Integer> it=hm2.keySet().iterator();  
  
        for(int i=0;i<s;i++)  
  
        {  
  
            int n=it.next();  
  
            int fac=hm2.get(n);  
  
            System.out.println(n+":"+fac);  
  
        }  
  
    }  
  
}  
  
  
import java.text.DecimalFormat;  
  
import java.util.HashMap;
```

```

import java.util.Iterator;

import java.util.LinkedHashMap;

public class kapes4

{public static LinkedHashMap<Integer,Integer> display(int[] a)

{

LinkedHashMap<Integer,Integer>hm=new LinkedHashMap<Integer,Integer>();

for(int i=0;i<a.length;i++)

{

int u=1;

for(int j=1;j<=a[i];j++)

{

u=u*j;

}

hm.put(a[i],u);

}

return hm;

}}

```

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

```
import java.util.StringTokenizer;
```

```
public class User {  
    public static void main(String[] args){  
        String s1="Hi";  
        String s2="Hello";  
        System.out.println(capsStart(s1,s2));  
    }  
    public static String capsStart(String s1,String s2){  
        StringBuffer s5=new StringBuffer();  
        int q=s1.length();  
        int w=s2.length();  
        if(q>w)  
        {  
            s5.append(s1).append(s2).append(s1);  
        }  
    }  
}
```

```
    }

    else

    {

        s5.append(s2).append(s1).append(s2);

    }

    return s5.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

```
import java.util.*;
public class Main
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        String s =sc.nextLine();
        System.out.println(User.display(s));
    }
}

import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
public class User{

    public static String display(String n)
    {

        int year=0;
        String now="01/01/2015";
        SimpleDateFormat sdf1=new SimpleDateFormat("dd/MM/yyyy");
        try
        {

            sdf1.setLenient(false);
            Calendar c1=Calendar.getInstance();
            Date d=sdf1.parse(n);
            c1.setTime(d);
            int y=c1.get(Calendar.YEAR);
            int m=c1.get(Calendar.MONTH);
            int day=c1.get(Calendar.DAY_OF_MONTH);

            Calendar c2=Calendar.getInstance();
            Date d1=sdf1.parse(now);
            c1.setTime(d1);
            int y1=c2.get(Calendar.YEAR);
            int m1=c2.get(Calendar.MONTH);
            int day1=c2.get(Calendar.DAY_OF_MONTH);

            year=y1-y;
            //System.out.println(year);
            if(m>m1)
                year--;
            elseif(m==m1)
```

```
{if(day<day1)
year--;
}
}
catch(Exception e)
{
e.printStackTrace();
}
if(year>18)
return"eligible";
else
return"not-eligible";
} }
```

```
public class UserMainCode{
public static int getMaxSpan(int a[]) {
    int i,j,k,count,max=0,p=0;
    int n=a.length;
    for(i=0;i<n;i++)
    {
        count=0;
        for(j=i+1;j<n;j++)
        {
            if(a[i]==a[j])
            {
                p=j;
            }
        }
        for(k=i;k<=p;k++)
        {
            count++;
        }
        if(count>max)
        {
            max=count;
        }
    }
}
```

```
    }  
    return max;  
}
```

```
}
```

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

Solutions:

```
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[20];
        for (int i = 0; i < n; i++)
            a[i] = sc.nextInt();
        int res = User.addUniqueEven(a);
        if (res == -1)
            System.out.println("no even numbers");
        else
            System.out.println(res);

    }
}
```

```
public class User {
    public static int addUniqueEven(int a[]) {
        int i = 0, j = 0, count = 0, sum = 0;
        int n = a.length;
        for (i = 0; i < n; i++) {
            count = 0;
            for (j = i + 1; j < n; j++) {
                if (a[i] == a[j])
                    count++;
            }
            if (count == 0) {
                if (a[i] % 2 == 0)
                    sum = sum + a[i];
            }
        }
        if (sum == 0)
```

```
return -1;  
else  
return sum;  
}  
}
```

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

Solution :

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String s = sc.next();
        int res = User.checkPalindrome(s);
        if (res == 1)
            System.out.println("valid");
        else
            System.out.println("invalid");

    }
}

public class User {
    public static int checkPalindrome(String s) {
        int res = 0, i = 0, j = 0, count = 0, k = 0;
        StringBuffer sb = new StringBuffer(s);
        sb.reverse();
        if (sb.toString().equals(s)) {
            for (i = 0; i < s.length(); i++) {
                count = 0;
                for (j = i + 1; j < s.length(); j++) {
                    if (s.charAt(i) == s.charAt(j))
                        count++;
                }
                if (count == 0)
                    if (s.charAt(i) == 'a' || s.charAt(i) == 'e'
                        || s.charAt(i) == 'i' || s.charAt(i) == 'o'
                        || s.charAt(i) == 'u' || s.charAt(i) == 'A'
                        || s.charAt(i) == 'E' || s.charAt(i) == 'T'
                        || s.charAt(i) == 'O' || s.charAt(i) == 'U')
                        k++;
                }
            }
            if (k >= 2)
                res = 1;
            else
                res = 0;
        }
        return res;
    }
}
```

```
}
```

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

Solution:

```
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();
        System.out.println(User.replacePlus(s1, s2));
    }
}

public class User {
    public static String replacePlus(String s1, String s2) {
        String ss1 = s1.toLowerCase();
        String ss2 = s2.toLowerCase();
        StringBuffer sb = new StringBuffer();
        for (int i = 0; i < s1.length(); i++) {
            char c = ss1.charAt(i);
            if (ss2.indexOf(c) == -1)
                sb.append('+');
            else
                sb.append(s1.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

Solution:

```
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(User.getLongestWord(s1));
    }
}

public class User {
    public static String getLongestWord(String s) {
```

```

int len, i, p = 0, max = 0, count = 0;
char b;
s = s.concat(" ");
len = s.length();
for (i = 0; i < len; i++) {
    b = s.charAt(i);
    if (b != ' ') {
        count++;
    } else {
        if (count > max) {
            max = count;
            p = i;
        }
        count = 0;
    }
}
return (s.substring(p - max, p));
}

import java.util.Scanner;
public class PalindromeMain {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        String s1 = sc.nextLine();

        System.out.println(Palindrome.checkPalindrome(s1));
    }
}

import java.util.StringTokenizer;

public class Palindrome {

    public static String checkPalindrome(String s1)
    {
        int res,max=0;
        String s2=null;
        StringTokenizer st=new StringTokenizer(s1, " ");
        while(st.hasMoreTokens())
        {
            String s=st.nextToken();
            res=s.length();
            if(res>max)
            {

```

```
    max=res;  
  
    s2=s;  
}  
}  
return s2;  
}  
}
```

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

Solution:

```
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();
        System.out.println(User.countNoOfWords(s1, s2));
    }
}

import java.util.StringTokenizer;
public class User {
    public static int countNoOfWords(String s1, String s2) {
        String[] a = new String[s1.length()];
        String[] b = new String[s2.length()];
        int i = 0, j = 0, count = 0;
        StringTokenizer st1 = new StringTokenizer(s1, " ");
        StringTokenizer st2 = new StringTokenizer(s2, " ");

        while (st1.hasMoreTokens()) {
            a[i] = st1.nextToken();
            i++;
        }
        while (st2.hasMoreTokens()) {
            b[j] = st2.nextToken();
            j++;
        }
        for (int k = 0; k < i; k++) {
            if (b[1].equals(a[k])) {
                count++;
            }
        }
        return count;
    }
}
```

```

import java.util.Scanner;
public class PalindromeMain {
    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);
        String s1 = sc.nextLine();
        String s2 = sc.nextLine();
        System.out.println(Palindrome.checkPalindrome(s1,s2));
    }

}

```

```

import java.util.StringTokenizer;

public class Palindrome {

    public static int checkPalindrome(String s1, String s2)
    {

        int count=0;
        StringTokenizer st=new StringTokenizer(s1, " ");
        StringTokenizer st1=new StringTokenizer(s2, " ");
        String a2=st1.nextToken();
        String b2=st1.nextToken();

        while(st.hasMoreTokens())
        {
            String s=st.nextToken();

            if(s.equalsIgnoreCase(b2))
            {
                count++;
            }
        }
        return count;
    }
}

```

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

Solution :

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

public class Main {

    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        int n = s.nextInt();
        ArrayList<Integer> al1 = new ArrayList<Integer>();
        ArrayList<Integer> al2 = new ArrayList<Integer>();
        ArrayList<Integer> a = new ArrayList<Integer>();
        for (int i = 0; i < n; i++)
            al1.add(s.nextInt());
        for (int i = 0; i < n; i++)
            al2.add(s.nextInt());
        a = User.generateOddEvenList(al1, al2);
        for (int i = 0; i < a.size(); i++)
            System.out.println(a.get(i));
    }
}

import java.util.ArrayList;

public class User {
    public static ArrayList<Integer> generateOddEvenList(ArrayList<Integer> a1,
    ArrayList<Integer> a2)
```

```

{
ArrayList<Integer> a = new ArrayList<Integer>();
int i = 0;
for (i = 0; i < a1.size(); i++) {
if (i % 2 == 0)
a.add(a2.get(i));
else
a.add(a1.get(i));
}
return a;
}
}

```

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

Solution:

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        String ss = s.nextLine();
        System.out.println(User.removeDuplicates(ss));
    }
}

import java.util.Iterator;
import java.util.LinkedHashSet;

public class User {
    public static String removeDuplicates(String s) {
        char a[] = s.toCharArray();
        StringBuffer sb = new StringBuffer();
        LinkedHashSet<Character> lh = new LinkedHashSet<Character>();
        for (int i = 0; i < a.length; i++)
            lh.add(a[i]);
        Iterator<Character> itr = lh.iterator();
        while (itr.hasNext()) {
            char c = itr.next();
            if (c != ' ')
                ;
            sb.append(c);
        }
        return sb.toString();
    }
}
```

```
import java.util.Scanner;  
  
class Main  
{    public static void main(String[] arg)  
{  
    Scanner sc=new Scanner(System.in);  
    String s=sc.nextLine();  
    System.out.println(MainClass.removeDuplicate(s));  
}}
```

```
import java.util.HashSet;  
  
import java.util.LinkedHashSet;  
  
public class MainClass {  
  
    public static String removeDuplicate(String s)  
    {  
        LinkedHashSet<Character> has=new LinkedHashSet<Character>();  
        for(int i=0;i<s.length();i++)  
        {  
            has.add(s.charAt(i));  
        }  
        StringBuffer sb=new StringBuffer();  
        for(Character c:has)  
        {  
            sb.append(c);  
        }  
        return sb.toString();  
    }  
}
```

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

Sample Output 1:

8

Solution:

```

import java.util.HashMap;
import java.util.Scanner;

publicclass Main {

publicstaticvoid main(String[] args) {
Scanner s = new Scanner(System.in);
int n = s.nextInt();
HashMap<Integer, Integer> hm1 = new HashMap<Integer, Integer>();
for (int i = 0; i < n; i++)
hm1.put(s.nextInt(), s.nextInt());
System.out.println(User.getAverageOfOdd(hm1));
}
}

import java.util.HashMap;
import java.util.Iterator;

publicclass User {
publicstaticint getAverageOfOdd(HashMap<Integer, Integer> hm1) {
int sum = 0, count = 0;
Iterator<Integer> itr = hm1.keySet().iterator();
while (itr.hasNext()) {
int key = itr.next();
if (key % 2 != 0) {
count++;
int val = hm1.get(key);
sum = sum + val;
}
}
int avg = sum / count;
return avg;
}
}

```

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.

Sample Input 1:

```
2
2
programmer
3000
8
manager
50000
```

Sample Output 1:

8

55000

Solution :

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.Scanner;

class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        HashMap<Integer, String> h1 = new HashMap<Integer, String>();
        HashMap<Integer, Integer> h2 = new HashMap<Integer, Integer>();
        HashMap<Integer, Integer> hm = new HashMap<Integer, Integer>();
        for (int i = 0; i < n; i++) {
            int id = sc.nextInt();
            h1.put(id, sc.next());
            h2.put(id, sc.nextInt());
        }
        hm = User.dis(n, h1, h2);
        Iterator<Integer> itr = hm.keySet().iterator();
        while (itr.hasNext()) {
            int id = itr.next();
            int sal = hm.get(id);
            System.out.println(id);
            System.out.println(sal);
        }
    }
}

import java.util.HashMap;
import java.util.Iterator;

public class User {
    public static HashMap<Integer, Integer> dis(int n,
        HashMap<Integer, String> h1, HashMap<Integer, Integer> h2) {
        HashMap<Integer, Integer> hm = new HashMap<Integer, Integer>();

        Iterator<Integer> itr = h1.keySet().iterator();
        while (itr.hasNext()) {
            int id = itr.next();
            String deg = h1.get(id);
```

```
    if (deg.equalsIgnoreCase("manager")) {
        hm.put(id, h2.get(id) + 5000);
    }
}
return hm;
}
```

```
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);

    HashMap<Integer, String> ip1=new HashMap<Integer, String>();
    HashMap<Integer, Integer> ip2=new HashMap<Integer, Integer>();
    int n=Integer.parseInt(sc.nextLine());

    for(int i=0;i<n;i++)
    {
        int id=Integer.parseInt(sc.nextLine());

        ip1.put(id,sc.nextLine());
        ip2.put(id,Integer.parseInt(sc.nextLine()));
    }

    HashMap<Integer, Integer> op=new HashMap<Integer, Integer>();
    op=MainClass.addsal(ip1,ip2);

    Iterator<Integer> itr=op.keySet().iterator();
    while(itr.hasNext())
    {
        int key=itr.next();
```

```

        int value=op.get(key);
        System.out.println(key);
        System.out.println(value);

    }

}

/*
int n=sc.nextInt();
for(int i=0;i<n;i++)
{
    int id=sc.nextInt();
    ip1.put(id,sc.nextInt());
    ip2.put(id,sc.nextInt());
}

*/
import java.util.HashMap;
import java.util.Iterator;

public class MainClass {
    public static HashMap<Integer, Integer>
addsal(HashMap<Integer, String> hm1,
       HashMap<Integer, Integer> hm2)
    {
        HashMap<Integer, Integer>op=new
HashMap<Integer, Integer>();

        Iterator<Integer> itr=hm1.keySet().iterator();

        while(itr.hasNext())
        {
            int id=itr.next();
            String s=hm1.get(id);
            if(s.equals("manager"))
            {
                int newsal=hm2.get(id)+5000;
                op.put(id,newsal);
            }
        }
        return op;
    }
}

```

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

Solution:

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

```
publicclass Main {
```

```

public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    String ss = s.nextLine();
    System.out.println(User.check(ss));
}
}

import java.util.StringTokenizer;

public class User {
    public static int check(String s) {
        StringTokenizer st = new StringTokenizer(s, " ");
        int n = st.countTokens();
        String[] s1 = new String[n];
        int i = 0, value = 0;
        while (st.hasMoreTokens()) {
            s1[i] = st.nextToken();
            i++;
        }
        if (s1[0].equals(s1[i - 1]))
            value = s1[0].length();
        else
            value = s1[0].length() + s1[i - 1].length();
        return value;
    }
}

```

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
```

Solution:

```
import java.util.Scanner;  
  
public class Main {  
    public static void main(String[] args) {  
        Scanner s = new Scanner(System.in);  
        int n = s.nextInt();  
        String[] str = new String[n];  
        for (int i = 0; i < n; i++)  
            str[i] = s.next();  
        System.out.println(User.concatCharacter(str));  
    }  
}  
  
public class User {  
    public static String concatCharacter(String[] s) {  
  
        StringBuffer sb = new StringBuffer();  
        for (int i = 0; i < s.length; i++) {  
            sb.append(s[i].charAt(s[i].length() - 1));  
        }  
        return sb.toString();  
    }  
}
```

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

Solutions:

```
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 result = User.getAnagrams(s1, s2);
```

```

if (result == 1)
System.out.println("Anagrams");
else
System.out.println("Not Anagrams");
}

}

import java.util.ArrayList;
import java.util.Collections;

publicclass User {
publicstatic int getAnagrams(String s1, String s2) {

String str1 = s1.toLowerCase();
String str2 = s2.toLowerCase();
ArrayList<Character> al1 = new ArrayList<Character>();
ArrayList<Character> al2 = new ArrayList<Character>();
ArrayList<Character> al3 = new ArrayList<Character>();
int res = 0;
for (int i = 0; i < s1.length(); i++)
al1.add(str1.charAt(i));
for (int i = 0; i < s2.length(); i++)
al2.add(str2.charAt(i));
al3.add(' ');
al1.removeAll(al3);
al2.removeAll(al3);
Collections.sort(al1);
Collections.sort(al2);
if (al1.equals(al2))
res = 1;
else
res = -1;
return res;
}
}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);
String s1 = sc.nextLine();
String s2 = sc.nextLine();
boolean b = Anagrams.check(s1, s2);
if (b == true)
System.out.println("TRUE");
else
System.out.println("FALSE");
}

```

```

    }

}

public class Anagrams
{
    public static boolean check(String s1, String s2)
    {
        boolean res=false;
        ArrayList<Character> a1=new ArrayList<Character>();
        ArrayList<Character> a2=new ArrayList<Character>();
        for(int i=0;i<s1.length();i++)
        {
            a1.add(s1.charAt(i));
        }

        for(int i=0;i<s2.length();i++)
        {
            a2.add(s2.charAt(i));
        }
        Collections.sort(a1);
        Collections.sort(a2);

        if((a1.containsAll(a2)) || (a2.containsAll(a1)))
        {
            res=true;
        }
        return res;
    }
}

```

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-XXXX)*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 1st argument corresponds to the previous meter reading and the 2nd 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

Solution:

```
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();
        System.out.println(User.calculateMeterReading(s1, s2));
    }
}
```

```
public class User {
    public static int calculateMeterReading(String s1, String s2) {
        String str1 = s1.substring(s1.length() / 2);
        String str2 = s2.substring(s2.length() / 2);
        int a = Integer.parseInt(str1);
        int b = Integer.parseInt(str2);
```

```
int res = (b - a) * 4;  
return res;  
}  
}
```

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]

Solution:

```
import java.text.ParseException;
import java.util.LinkedHashMap;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) throws ParseException {
        Scanner s = new Scanner(System.in);
        int n = s.nextInt();
        LinkedHashMap<String, String> hm = new LinkedHashMap<String, String>();
        for (int i = 0; i < n; i++)
            hm.put(s.next(), s.next());
        System.out.println(User.retirementEmployeeList(hm));
    }
}

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.Iterator;
import java.util.LinkedHashMap;

public class User {
    public static ArrayList<String> retirementEmployeeList(
        LinkedHashMap<String, String> hm) throws ParseException {
        ArrayList<String> al = new ArrayList<String>();
        SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");
        String s = "01/01/2014";
        Date d2 = sdf.parse(s);
        Date d1 = new Date();
        Iterator<String> itr = hm.keySet().iterator();
        while (itr.hasNext()) {
            String key = itr.next();
            String val = hm.get(key);
            d1 = sdf.parse(val);
            Calendar c = Calendar.getInstance();
            c.setTime(d1);
            int y1 = c.get(Calendar.YEAR);
            int m1 = c.get(Calendar.MONTH);
            int day1 = c.get(Calendar.DAY_OF_MONTH);
            c.setTime(d2);
```

```

int y2 = c.get(Calendar.YEAR);
int m2 = c.get(Calendar.MONTH);
int day2 = c.get(Calendar.DAY_OF_MONTH);
int y = Math.abs(y1 - y2);
if (m1 == m2) {
if (day1 > day2)
y--;
} elseif (m1 > m2)
y--;
if (y >= 60)
al.add(key);
}
return al;
}
}

```

```

import java.text.ParseException;
import java.util.HashMap;
import java.util.Scanner;

public class NewClassMain {

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

        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        int i=0;
        sc.nextLine();
        HashMap<String, String> hm=new HashMap<String, String>();
        for(i=0;i<n;i++)
        {
            hm.put(sc.nextLine(),sc.nextLine());
        }

        System.out.println(NewClass.retirement(hm));
    }
}

public class NewClass {

```

```
public static ArrayList<String> retirement(HashMap<String, String> hm)
throws ParseException
{
    ArrayList<String> a1=new ArrayList<String>();
    String s="01/01/2014";
    Iterator<String> itr=hm.keySet().iterator();
    while(itr.hasNext())
    {
        String k=itr.next();
        String dob=hm.get(k);

        SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
        Date d1=sdf.parse(dob);
        Date d2=sdf.parse(s);

        Calendar cal=Calendar.getInstance();
        cal.setTime(d1);

        int y1= cal.get(Calendar.YEAR);
        cal.setTime(d2);

        int y2= cal.get(Calendar.YEAR);

        int res=y2-y1;
        if(res>=60)

        {
            a1.add(k);
        }

    }

    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

Solution:

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

```
public class Main {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        int n = sc.nextInt();  
        int i = User.getKaprekarNumber(n);  
        if (i == 1)  
            System.out.println("Kaprekar Number");  
        else  
            System.out.println("Not Kaprekar Number");  
    }  
}
```

```
public class User {  
    public static int getKaprekarNumber(int temp) {  
        int n = temp;  
        int sq = n * n;  
        int sqr = sq;  
        int res = 0;  
        int count = 0;  
        while (sq != 0) {  
            count++;  
            sq = sq / 10;  
        }  
        //String a = Integer.toString(sqr);  
  
        String a = String.valueOf(sqr);
```

```
        String n1 = a.substring(count / 2);  
        String n2 = a.substring(0, count / 2);  
        int i = Integer.parseInt(n1);  
        int j = Integer.parseInt(n2);  
        if ((i + j) == temp)  
            res = 1;
```

```

else
res = -1;
return res;
}
}

public class Palindrome {
    public static int removeDuplicate(int n)
    {
        int temp = n;
        int sq = n * n;
        int sqr=sq;
        int res = 0;
        String sqs=String.valueOf(sq);
        int len=sqs.length();
        String a = String.valueOf(sqr);
        String n1 = a.substring(len/2);
        String n2 = a.substring(0,len/2);
        int i = Integer.parseInt(n1);
        int j = Integer.parseInt(n2);
        if ((i + j) == temp)
        res = 1;
        else
        res = -1;
    return res;
    }
}

```

16. Vowels

Given a String input, write a program to find the word which has 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

Solution:

```
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 s = sc.nextLine();
        System.out.println(User.storeMaxVowelWord(s));
    }
}
```

```
import java.util.StringTokenizer;
```

```
public class User {
    public static String storeMaxVowelWord(String s) {
        StringTokenizer st = new StringTokenizer(s, " ");
        int count = 0, max = 0;
        String s2 = null;
        while (st.hasMoreTokens()) {
            String s1 = st.nextToken();
            count = 0;
            for (int i = 0; i < s1.length(); i++) {
                if (s1.charAt(i) == 'a' || s1.charAt(i) == 'e'
                    || s1.charAt(i) == 'i' || s1.charAt(i) == 'o'
                    || s1.charAt(i) == 'u' || s1.charAt(i) == 'A'
                    || s1.charAt(i) == 'E' || s1.charAt(i) == 'T'
                    || s1.charAt(i) == 'O' || s1.charAt(i) == 'U')
                    count++;
            }
            if (count > max) {
                max = count;
                s2 = s1;
            }
        }
        return s2;
    }
}
```

17. Unique Characters

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:

7

(Hint :Unique characters are : H,W,A,R,E,Y,U and other characters are repeating)

Sample Input 2:

MAMA

Sample Output2:

-1

Solution:

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String s = sc.next();
        System.out.println(User.checkUnique(s));
    }
}
```

```

publicclass User {
publicstaticint checkUnique(String s) {
StringBuffer sb = new StringBuffer(s);
int len = s.length();
int i = 0, j = 0, count;
for (i = 0; i < len; i++) {
count = 0;
for (j = i + 1; j < len; j++) {
if (sb.charAt(i) == sb.charAt(j)) {
sb.deleteCharAt(j);
count++;
j--;
len--;
}
}
if (count > 0) {
sb.deleteCharAt(i);
i--;
len--;
}
}
if(sb.length()==0)
    return -1;
else
return 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

Solutions:

```
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);
        int n = sc.nextInt();
        int[] a = new int[20];
        for (int i = 0; i < n; i++) {
            a[i] = sc.nextInt();
        }
        System.out.println(User.addPrimeIndex(a));
    }
}

public class User {
    public static int addPrimeIndex(int a[], int n) {
        int count=0, sum=0, temp=0;
        int avg=0;
        for(int i=2;i<=n;i++)
    }
```

```

{
count=0;
for(int j=1;j<i;j++)
{
if(i%j==0)
count++;

}
if(count==1)
{
temp++;
sum=sum+a[i];
}
}
avg=sum/temp;
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

Solution:

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

```

ArrayList<Integer> al1 = new ArrayList<Integer>();
ArrayList<Integer> al2 = new ArrayList<Integer>();
ArrayList<Integer> res = new ArrayList<Integer>();
for (int i = 0; i < n; i++)
al1.add(sc.nextInt());
for (int i = 0; i < n; i++)
al2.add(sc.nextInt());
char c = sc.next().charAt(0);
res = User.performSetOperations(al1, al2, c);
for (int i = 0; i < res.size(); i++)
System.out.println(res.get(i));
}
}

```

```

import java.util.ArrayList;
import java.util.LinkedHashSet;

public class User {
public static ArrayList<Integer> performSetOperations(
ArrayList<Integer> al1, ArrayList<Integer> al2, char c) {

LinkedHashSet<Integer> h = new LinkedHashSet<Integer>();
ArrayList<Integer> al3 = new ArrayList<Integer>();
switch (c) {
case '+':
al1.addAll(al2);
h.addAll(al1);
al3.addAll(h);
break;
case '*':
for (int i = 0; i < al1.size(); i++) {
for (int j = 0; j < al2.size(); j++) {
if (al1.get(i) == al2.get(j)) {
al3.add(al1.get(i));
}
}
}
break;
case '-':
for (int i = 0; i < al1.size(); i++) {
for (int j = 0; j < al2.size(); j++) {
if (al1.get(i) == al2.get(j)) {

```

```

al1.remove(i);
}
}
}
al3.addAll(al1);
break;

}

return al3;
}
}
}

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

class Main
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        ArrayList<Integer> aa=new ArrayList<Integer>();
        for(int i=0;i<n;i++)
        {
            aa.add(sc.nextInt());
        }
        ArrayList<Integer> aa2=new ArrayList<Integer>();
        for(int i=0;i<n;i++)
        {
            aa2.add(sc.nextInt());
        }
        char c=sc.next().charAt(0);

        ArrayList<Integer> op=new ArrayList<Integer>();

        op=MainClass.setOPeration(n, aa, aa2, c);
    }
}

```

```
Iterator<Integer> itr=op.iterator();
while(itr.hasNext())
{
    int a=(Integer)itr.next();
    System.out.println(a);
}

import java.util.ArrayList;

public class MainClass {

    public static ArrayList<Integer> setOperation
    (int n,ArrayList<Integer>aa,ArrayList<Integer>aa2,char c)
    {

        ArrayList<Integer> aa3= new ArrayList<Integer>();

        if(c=='+')
        {
            aa.removeAll(aa2);
            aa.addAll(aa2);
            aa3=aa;
        }

        if(c=='*')
        {
            aa.retainAll(aa2);
            aa3=aa;
        }

        if(c=='-')
        {
            aa.removeAll(aa2);
            aa3=aa;
        }
    }
}
```

```
    }  
    return aa3;  
}  
}
```

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

Solution:

```
import java.util.Scanner;  
  
public class Main {  
  
    public static void main(String[] args) {
```

```

Scanner sc = newScanner(System.in);
int n = sc.nextInt();
int[] a = new int[n];
for (int i = 0; i < n; i++)
a[i] = sc.nextInt();
System.out.println(User.getLargestSpan(a));

}
}

```

```

public class User {
public static int getLargestSpan(int[] a) {
int len = a.length;
int i = 0, j = 0, e = 0, count = 0;
for (i = 0; i < len; i++) {
for (j = i + 1; j < len; j++) {
if (a[i] == a[j]) {
e = j;
}
}
if (e - i > count)
count = e - i;
}
return count + 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 the 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
```

Solution:

```
import java.text.ParseException;
import java.util.ArrayList;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) throws ParseException {
        Scanner sc = new Scanner(System.in);
        int n=sc.nextInt();
        ArrayList<String> a=new ArrayList<String>();
        for(int i=0;i<n;i++)
            a.add(sc.next());
        System.out.println(User.highestScorer(a));
        sc.close();
    }
}
```

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

public class User {
```

```

public static String highestScorer(ArrayList<String> a) {
    String ss=null,name=null,Name=null;
    int m1=0,m2=0,m3=0,sum=0,max=0;
    for(int i=0;i<a.size();i++)
    {
        ss=a.get(i);
        StringTokenizer st=new StringTokenizer(ss,"-");
        while(st.hasMoreTokens())
        {
            name=st.nextToken();
            m1=Integer.parseInt(st.nextToken());
            m2=Integer.parseInt(st.nextToken());
            m3=Integer.parseInt(st.nextToken());
            sum=m1+m2+m3;
            if(max<sum)
            {
                max=sum;
                Name=name;
            }
        }
    }
    return Name;
}
}

```

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

Solution:

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

```
publicclass Main {  
publicstaticvoid main(String[] args) {  
Scanner sc = newScanner(System.in);  
String s = sc.nextLine();  
System.out.println(User.getWordWithMaximumVowels(s));  
}  
}
```

```
import java.util.StringTokenizer;
```

```
publicclass User {  
publicstatic String getWordWithMaximumVowels(String s) {  
StringTokenizer st = new StringTokenizer(s, " ");  
int count = 0, max = 0;  
String res = null;  
String f = null;  
while (st.hasMoreTokens()) {  
res = st.nextToken();  
count = 0;  
for (int k = 0; k < res.length(); k++) {  
if (res.charAt(k) == 'a' || res.charAt(k) == 'e'  
|| res.charAt(k) == 'i' || res.charAt(k) == 'o'  
|| res.charAt(k) == 'u' || res.charAt(k) == 'A'  
|| res.charAt(k) == 'E' || res.charAt(k) == 'T'  
|| res.charAt(k) == 'O' || res.charAt(k) == 'U')  
count++;  
if (count > max) {  
max = count;  
f = res;  
}  
}  
}
```

```
return f;  
}  
}
```

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

Solution:

```
import java.util.Scanner;  
  
public class Main {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        String s = sc.nextLine();  
        int res = User.testOrderVowels(s);  
        if (res == 1)  
            System.out.println("valid");  
        else  
            System.out.println("invalid");  
    }  
}
```

```

System.out.println("valid");
else
System.out.println("invalid");
}
}

public class User {
public static int testOrderVowels(String s1) {

StringBuffer sb = new StringBuffer();
int res = 0;
for (int i = 0; i < s1.length(); i++) {
if (s1.charAt(i) == 'a' || s1.charAt(i) == 'A'
|| s1.charAt(i) == 'e' || s1.charAt(i) == 'E'
|| s1.charAt(i) == 'i' || s1.charAt(i) == 'I'
|| s1.charAt(i) == 'o' || s1.charAt(i) == 'O'
|| s1.charAt(i) == 'u' || s1.charAt(i) == 'U') {
sb.append(s1.charAt(i));
}
}
if (sb.toString().equals("aeiou"))
res = 1;
else
res = 0;
return res;
}
}

```

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

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String string=sc.nextLine();
        System.out.println(User.swapPairs(string));
        sc.close();
    }
}
```

```
public class User {
    public static String swapPairs(String s) {
        StringBuffer sb=new StringBuffer();
        if(s.length()%2==0)
        {
            for(int i=0;i<s.length()-1;i+=2)
            {
                sb.append(s.charAt(i+1)).append(s.charAt(i));
            }
        }
        else
        {
            for(int i=0;i<s.length()-1;i+=2)
```

```

{
sb.append(s.charAt(i+1)).append(s.charAt(i));
}
sb.append(s.charAt(s.length()-1));
}
return sb.toString();
}
}

```

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

Solution:

```

import java.util.Scanner;

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = new Scanner(System.in);
String s = sc.next();
System.out.println(User.getdigits(s));
}
}

```

```

public class User {
    public static int getDigits(String s) {
        int sum = 0, n = 0;
        for (int i = 0; i < s.length(); i++) {
            if (s.charAt(i) >= 65 && s.charAt(i) <= 90 || s.charAt(i) >= 97
                && s.charAt(i) <= 122)
            ;
            else {
                n = Character.getNumericValue(s.charAt(i));
                sum = sum + n;
            }
        }
        return sum;
    }
}

public static String removeDuplicate(String s) {

    int sum = 0, n=0;
    for (int i = 0; i < s.length(); i++) {

        if( Character.isDigit(s.charAt(i)) )
        {
            int c=Character.getNumericValue(s.charAt(i));
            sum = sum +c;

        }
    }
    String sum1=String.valueOf(sum);
    return sum1;
}

```

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

Solution:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String s = sc.next();
        boolean flag = User.validatePassword(s);
        if (flag == true)
            System.out.println("valid");
        else
            System.out.println("invalid");
    }
}
```

```
public class User {
    public static boolean validatePassword(String s) {
        int number = 0, c = 0, sp = 0;
        boolean flag = false;
        for (int i = 0; i < s.length(); i++) {
```

```

if (s.length() >= 8) {
    if (Character.isDigit(s.charAt(i))) {
        number++;
    }
    if (Character.isLetter(s.charAt(i))) {
        c++;
    } else {
        if (s.charAt(i) != ' ' && !Character.isDigit(s.charAt(i))
            && !Character.isLetter(s.charAt(i)))
            sp++;
    }
}
}
}

if (number >= 1 && c >= 1 && sp >= 1)
flag = true;
return flag;
}
}
}

```

```

public static boolean removeDuplicate(String s) {

    int number = 0, c = 0, sp = 0, len=0;
    boolean flag = false;
    for (int i = 0; i < s.length(); i++)
    {
        if (s.length() >= 8)
        {
            len++;
        }
        if (Character.isDigit(s.charAt(i)))
        {
            number++;
        }
        if (Character.isLetter(s.charAt(i)))
        {
            c++;
        }

        if (s.charAt(i) != ' ' && !Character.isDigit(s.charAt(i))
            && !Character.isLetter(s.charAt(i)))
        {
            sp++;
        }
    }

    if (number >= 1 && c >= 1 && sp >= 1 && len>1)
    flag = true;
}
}
}

```

```
    return flag;  
}
```

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

Solution:

```
import java.text.ParseException;
import java.util.*;

public class Main {
    public static void main(String[] args) throws ParseException {
        Scanner sc = new Scanner(System.in);
        int n=sc.nextInt();
        TreeMap<Integer, Integer> t=new TreeMap<Integer, Integer>();
        HashMap<Integer, String> h1=new HashMap<Integer, String>();
        HashMap<Integer, Integer> h2=new HashMap<Integer, Integer>();
        for(int i=0;i<n;i++)
        {
            int id=sc.nextInt();
            h1.put(id, sc.next());
            h2.put(id, sc.nextInt());
        }
        t=User.calSalary(h1,h2);
        Iterator<Integer> it1=t.keySet().iterator();
```

```

while(it1.hasNext())
{
    int id=it1.next();
    int val=t.get(id);
    System.out.println(id);
    System.out.println(val);
}
sc.close();
}

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

public class User {
public static TreeMap<Integer, Integer> calSalary(HashMap<Integer, String> h1,
HashMap<Integer, Integer> h2) throws ParseException {
    TreeMap<Integer, Integer> t=new TreeMap<Integer, Integer>();
    Iterator<Integer> it1=h1.keySet().iterator();
    SimpleDateFormat sd=new SimpleDateFormat("dd-MM-yyyy");
    String ss="01-09-2014";
    int new_sal=0;
    while(it1.hasNext())
    {
        int id1=it1.next();
        String dob=h1.get(id1);
        int salary=h2.get(id1);
        Date d1=sd.parse(dob);
        Date d2=sd.parse(ss);
        d1=sd.parse(dob);
        int y1=d1.getYear();
        int y2=d2.getYear();
        int year=Math.abs(y1-y2);
        if(year>=25 && year<=30)
        {
            new_sal=salary+(salary*20/100);
            t.put(id1,new_sal);
        }
        else if(year>=31 && year<=60)
        {
            new_sal=salary+(salary*30/100);
            t.put(id1,new_sal);
        }
        else
        ;
    }
    return t;
}
}

```

28. Grade Calculator

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 to 80 and greater than or equal to 60 store medal as ""SILVER"".
- 3 .If Mark is less than to 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 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 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

Solution:

```
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 n = sc.nextInt();  
  
        int i;  
  
        HashMap<Integer, Integer> hm = new HashMap<Integer, Integer>();  
  
        for (i = 0; i < n; i++) {  
  
            hm.put(sc.nextInt(), sc.nextInt());  
  
        }  
  
        TreeMap<Integer, String> t = new TreeMap<Integer, String>();  
  
        t.putAll(User.display(n, hm));  
  
        Iterator<Integer> it = t.keySet().iterator();  
  
        while (it.hasNext()) {  
  
            int r = it.next();  
  
            if (hm.get(r) == null) {  
                System.out.println("FAIL");  
            } else if (hm.get(r) == 1) {  
                System.out.println("100");  
            } else if (hm.get(r) == 2) {  
                System.out.println("1010");  
            } else {  
                System.out.println("GOLD");  
            }  
        }  
    }  
}
```

```

String g = t.get(r);

System.out.println(r);

System.out.println(g);

}}

import java.util.HashMap;
import java.util.Iterator;
import java.util.TreeMap;

public class User {

public static TreeMap<Integer, String> display(int n,
HashMap<Integer, Integer> h) {
TreeMap<Integer, String> t = new TreeMap<Integer, String>();
Iterator<Integer> i = h.keySet().iterator();
while (i.hasNext()) {
int r = i.next();
int m = h.get(r);
if (m >= 80)
t.put(r, "GOLD");
elseif (m < 80 && m >= 60)
t.put(r, "SILVER");
elseif (m < 60 && m >= 45)
t.put(r, "BRONZE");
else
t.put(r, "FAIL");
}
return t;
}

}
}

```

29. DigitSum

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

Solution:

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

```
publicclass Main {
```

```
publicstaticvoid main(String[] args) {  
Scanner s = newScanner(System.in);  
int n = s.nextInt();  
System.out.println(User.getDigitSum(n));  
}
```

```
}
```

```
publicclass User {
```

```
publicstaticint getDigitSum(int n) {
```

```
int sum = 0;
```

```
while (n > 10) {
```

```
int r = 0;
```

```
sum = 0;
```

```
while (n != 0) {
```

```
r = n % 10;
```

```
sum = sum + r;
```

```
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

Solution:

```
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();
        boolean b = User.checkAnagram(s1, s2);
        if (b == true)
```

```

System.out.println("TRUE");
else
System.out.println("FALSE");
}
}

import java.util.ArrayList;
import java.util.Collections;

public class User {
public static boolean checkAnagram(String s1, String s2) {
boolean b = false;
ArrayList<Character> a1 = new ArrayList<Character>();
ArrayList<Character> a2 = new ArrayList<Character>();
ArrayList<Character> a3 = new ArrayList<Character>();
for (int i = 0; i < s1.length(); i++)
a1.add(s1.toLowerCase().charAt(i));
for (int i = 0; i < s2.length(); i++)
a2.add(s2.toLowerCase().charAt(i));
a3.add(' ');
a1.removeAll(a3);
a2.removeAll(a3);
Collections.sort(a1);
Collections.sort(a2);
if (a1.equals(a2))
b = true;
return b;
}

public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    String s1 = sc.nextLine();
    String s2 = sc.nextLine();
    boolean b = Anagrams.check(s1, s2);
    if (b == true)
        System.out.println("TRUE");
    else
        System.out.println("FALSE");
}
}

```

```

public class Anagrams
{
    public static boolean check(String s1,String s2)
    {
        boolean res=false;
        ArrayList<Character> a1=new ArrayList<Character>();
        ArrayList<Character> a2=new ArrayList<Character>();
        for(int i=0;i<s1.length();i++)
        {
            a1.add(s1.charAt(i));
        }

        for(int i=0;i<s2.length();i++)
        {
            a2.add(s2.charAt(i));
        }
        Collections.sort(a1);
        Collections.sort(a2);

        if((a1.containsAll(a2))|| (a2.containsAll(a1)))
        {
            res=true;
        }
        return res;
    }
}

```

1. Shift Left

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
```

Solution:

```
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 res[]={};  
        for(int i=0;i<res.length;i++)  
            System.out.println(res[i]);  
    }  
}
```

```

public class User {
    public static int[] shiftLeft(int a[],int n)
    {
        int b[]={};
        int k=0;
        for(int i=0;i<n;i++)
        {
            if(a[i]!=5)
            {
                b[k]=a[i];
                k++;
            }
        }
        return b;
    }
}

```

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

Solution:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        String[] s = new String[n];
        for (int i = 0; i < n; i++) {
            s[i] = sc.next();
        }
        System.out.println(User.sumOfDigits(s));
    }
}

public class User {
    public static int sumOfDigits(String[] ss) {
        int sum = 0, n = 0;
        for (int i = 0; i < ss.length; i++) {
            String s = ss[i];
            for (int k = 0; k < s.length(); k++) {
                if (Character.isDigit(s.charAt(k))) {
                    n = Character.getNumericValue(s.charAt(k));
                    sum = sum + n;
                }
            }
        }
        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
```

Solution:

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

public class Piddi {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int size = sc.nextInt();
        String input[] = new String[size];
        for (int i = 0; i < size; i++) {
            input[i] = sc.next();
        }
    }
}
```

```

HashSet<String> hs = new HashSet<String>();
for (int i = 0; i < size; i++) {
hs.add(input[i]);
}

size = hs.size();
int i = 0;

int count = 0;
for (i = 0; i < size; i++) {
for (int j = 0; j < size; j++) {
if (input[i].equals(input[j]) == false) {
if (input[j].startsWith(input[i])) {
count++;
}
}
}
}
System.out.println(count);
}
}

```

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

Solution:

```
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 n2 = sc.nextInt();  
        String[] s2 = new String[n2];  
        for (int i = 0; i < n2; i++) {  
            s2[i] = sc.next();  
        }  
        System.out.println(User.countCommonStrings(s1, s2, n1, n2));  
    }  
}
```

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

```

publicclass User {
publicstatic int countCommonStrings(String[] a, String[] b, int n1, int n2) {
    int count = 0;
    ArrayList<String> al = new ArrayList<String>();
    for (int i = 0; i < n1; i++) {
        for (int j = 0; j < n2; j++) {
            if (a[i].equalsIgnoreCase(b[j]))
                if (!al.contains(b[i])) {
                    count++;
                    al.add(a[i]);
                }
            }
        }
    }
    return count;
}
}

```

```

import java.util.HashSet;
import java.util.Iterator;
public class Palindrome {
    public static int removeDuplicate(String[] words1, String[]
words2)
{
    int count=0;
    HashSet<String> s1=new HashSet<String>();
    HashSet<String> s2=new HashSet<String>();
    for (int i=0;i<words1.length;i++)
    {
        s1.add(words1[i]);
    }
    for (int i=0;i<words2.length;i++)
    {
        s2.add(words2[i]);
    }
    Iterator<String> it1=s1.iterator();

    while (it1.hasNext())
    {
        String its1=it1.next();
        Iterator<String> it2=s2.iterator();
        while (it2.hasNext())
        {
            String its2=it2.next();
            if (its1.equals(its2))
            {
                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 sequnce 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:

Solution:

```

import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        System.out.println(User.getFibonacciSum(n));
    }
}

public class User {
    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;
            d = d + c;
        }
        return d;
    }
}

```

36. Email 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

Solution:

```
import java.util.Scanner;

class Main {
    public static void main(String args[]) {
        Scanner sc = new Scanner(System.in);
        String email = sc.next();
        System.out.println(User.ValidateEmail(email));
    }
}

public class User {
    public static boolean ValidateEmail(String email) {
        boolean b = false;
        if (email.matches("[a-zA-Z0-9]{3,}(@)[a-zA-Z]{5,}(.)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:

```
5
11
5
14
26
3
3
5
3
1
```

Sample Output 1:

```
1
11
14
26
```

Solution:

```
public class Main {
    public static void main(String[] args) throws ParseException {
        Scanner sc=new Scanner(System.in);

        int n1=sc.nextInt();
        int[] a=new int[n1];
        for(int i=0;i<n1;i++)
```

```

a[i]=sc.nextInt();
int n2=sc.nextInt();
int[] b= new int[n2];
for(int i=0;i<n2;i++)
b[i]=sc.nextInt();
int[] res=User.display(a,b,n1,n2);
for(int i=0;i<res.length;i++)
System.out.println(res[i]);

```

```

}
}
```

```

public class User {
public static int[] display(int a[],int b[],int n1,int n2)
{
TreeSet<Integer> ts1=new TreeSet<Integer>();
TreeSet<Integer> ts2=new TreeSet<Integer>();
TreeSet<Integer> ts3=new TreeSet<Integer>();
ArrayList<Integer> aa=new ArrayList<Integer>();
for(int i=0;i<a.length;i++)
ts1.add(a[i]);
for(int i=0;i<b.length;i++)
ts2.add(b[i]);
ts1.addAll(ts2);
for(int i=0;i<n1;i++)
{
for(int j=0;j<n2;j++)
{
if(a[i]==b[j])
ts3.add(a[i]);
}
}
ts1.removeAll(ts3);
aa.addAll(ts1);
int res[]=new int[aa.size()];
for(int i=0;i<res.length;i++)
res[i]=aa.get(i);
return res;
}
}
```

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

Solution:

```
import java.text.ParseException;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) throws ParseException {
        Scanner s=new Scanner(System.in);
        String s1=s.next();
        System.out.println(User.findOldDate(s1));
    }
}

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

public class User {
    public static String findOldDate(String s1) throws ParseException
    {
        SimpleDateFormat sd1=new SimpleDateFormat("dd-MM-yyyy");
        Date d1=sd1.parse(s1);
        SimpleDateFormat sd2=new SimpleDateFormat("EEEEEE");
        String name=sd2.format(d1);
        return name.toLowerCase();
    }
}
```

```
    }  
}
```

39. Addtime

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:

1:23:59:58

Sample Input 2:

23:59:59

23:59:59

Sample Output 2:

1:23:59:58

Solution:

```
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.next();
String s2 = sc.next();
System.out.println(User.addTime(s1, s2));
}
}

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
import java.util.TimeZone;

public class User {
    public static String addTime(String s1, String s2) throws ParseException {
        // adding two times
        SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss");
        sdf.setTimeZone(TimeZone.getTimeZone("UTC"));
        Date d1 = sdf.parse(s1);
        Date d2 = sdf.parse(s2);
        long time = d1.getTime() + d2.getTime();
        String s = sdf.format(new Date(time));
        // to calculate the day
        Calendar c = Calendar.getInstance();
        c.setTime(sdf.parse(s));
        int day = c.get(Calendar.DAY_OF_MONTH);
        if (day > 1)
            day = day - 1;
        String op = day + ":" + s;
        return op;
    }
}

import java.util.StringTokenizer;

public class Palindrome {
    public static String removeDuplicate(String a, String b)
    {

        StringTokenizer st1=new StringTokenizer(a, ":");
        StringTokenizer st2=new StringTokenizer(b, ":");


```

```

int h1=Integer.parseInt(st1.nextToken());
int m1=Integer.parseInt(st1.nextToken());
int s1=Integer.parseInt(st1.nextToken());
int d=0;
int h2=Integer.parseInt(st2.nextToken());
int m2=Integer.parseInt(st2.nextToken());
int s2=Integer.parseInt(st2.nextToken());

int m,h,s;
m=m1+m2;
h=h1+h2;
s=s1+s2;

if(s>=60)
{
    m=m+1;
    s=s-60;
    if(m1>=60)
    {
        h=h+1;
        m=m-60;
        if(h>=24)
        {
            d=d+1;
            h=h-24;
        }
    }
}

if(m1>=60)
{
    h=h+1;
    m=m-60;
    if(h>=24)
    {
        d=d+1;
        h=h-24;
    }
}

if(h>=24)
{
    d=d+1;
    h=h-24;
}

StringBuffer sb=new StringBuffer();

sb.append(d).append(":").append(h).append(":").append(m).append(":").append(s);

return sb.toString();
}
}

```

0:00:01
0:00:02
0:0:0:3

12:45:30
13:50:45
1:2:36:15

12:20:20

22:20:10

12:20:20
22:20:10
1:10:40:30

1:20:20

2:20:10

1:20:20
2:20:10
0:3:40:30

```
import java.util.StringTokenizer;

public class Palindrome {
    public static String removeDuplicate(String a, String b)
    {
        StringTokenizer st1=new StringTokenizer(a,":");
        StringTokenizer st2=new StringTokenizer(b,":");

        int h1=Integer.parseInt(st1.nextToken());
        int m1=Integer.parseInt(st1.nextToken());
        int s1=Integer.parseInt(st1.nextToken());
        int d=0;
        int h2=Integer.parseInt(st2.nextToken());
        int m2=Integer.parseInt(st2.nextToken());
        int s2=Integer.parseInt(st2.nextToken());

        int m,h,s;
        m=m1+m2;
        h=h1+h2;
        s=s1+s2;

        while(s>=60)
```

```

    {
        m=m+1;
        s=s-60;
    }
while (m>=60)
{
    h=h+1;
    m=m-60;
}

while (h>=24)
{
    d=d+1;
    h=h-24;
}

StringBuffer sb=new StringBuffer();

sb.append(d).append(":").append(h).append(":").append(m).append(":").ap
pend(s);

return sb.toString();
}

}

```

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

Solution:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String isbn=s.next();
        boolean b=User.validateISBN(isbn);
        System.out.println(b);
    }
}

import java.util.*;

public class User {
    public static boolean validateISBN(String isbn)
    {
        int sum=0,k=10;
        for(int i=0;i<isbn.length();i++)
        {
            int a=Character.getNumericValue(isbn.charAt(i));
            sum=sum+(a*k);
            k--;
        }
        if(sum%11==0)
            return true;
        else
            return false;
    }
}
```

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

Solution:

```
import java.text.ParseException;
import java.util.Scanner;

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(User.findOldDate(s1,s2));
    }
}

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

public class User {
    public static String findOldDate(String s1,String s2) throws ParseException
    {
        SimpleDateFormat sd1=new SimpleDateFormat("dd-MM-yyyy");
        SimpleDateFormat sd2=new SimpleDateFormat("dd-MM-yyyy");
        Date d1=sd1.parse(s1);
        Date d2=sd2.parse(s2);
        if(d1.compareTo(d2)<0)
            return s1;
        else
            return s2;
    }
}
```

```

        Date d1=sd1.parse(s1);
        Date d2=sd1.parse(s2);
        Calendar c=Calendar.getInstance();
        c.setTime(d1);
        int day1=c.get(Calendar.DAY_OF_MONTH);
        int m1=c.get(Calendar.MONTH);
        int y1=c.get(Calendar.YEAR);
        c.setTime(d2);
        int day2=c.get(Calendar.DAY_OF_MONTH);
        int m2=c.get(Calendar.MONTH);
        int y2=c.get(Calendar.YEAR);
        SimpleDateFormat sd2=new SimpleDateFormat("MM/dd/yyyy");
        String res=null;
        if(y1==y2)
        {
            if(m1==m2)
            {
                if(day1==day2)
                {
                    res=sd2.format(d1);
                }
            }
            else
            {
                if(m1>m2)
                    res=sd2.format(d2);
                else
                    res=sd2.format(d1);
            }
        }
        else
        {
            if(y1>y2)
                res=sd2.format(d2);
            else
                res=sd2.format(d1);
        }
        return res;
    }
}

import java.text.ParseException;
import java.text.SimpleDateFormat;

import java.util.Date;
public class Palindrome {
    public static String removeDuplicate(String s1,String s2) throws
ParseException

```

```

{
    SimpleDateFormat sdf1=new SimpleDateFormat("dd-MM-yyyy");
    Date d1=sdf1.parse(s1);
    Date d2=sdf1.parse(s2);
    String res=null;

    SimpleDateFormat sdf2=new SimpleDateFormat("MM/dd/yyyy");

    if(d1.compareTo(d2)<0)
    {
        res=sdf2.format(d1);
    }
    else
    {
        res=sdf2.format(d2);
    }

    return res;
}

```

42. Interest calculation

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 to 60 and greater than or equal to 30 then interest rate is 7% of Amount.
 - c. If the age less than to 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
```

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
```

Solution:

```
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.findDiscountRate(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);
}
}

```

```

public class UserMaincode
{
public static TreeMap<String, Integer> findDiscountRate
(HashMap<String, String> hm,HashMap<String, Integer> hm1) throws ParseException
{
TreeMap<String, Integer> tm=new TreeMap<String, Integer>();
SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");
Iterator<String> itr1=hm.keySet().iterator();
while(itr1.hasNext())
{
try
{
String id=itr1.next();
String dor=hm.get(id);
int am=hm1.get(id);
Date d1=sdf.parse(dor);
String s1="01-01-2015";
Date d2=sdf.parse(s1);
int y1=d1.getYear();
int m1=d1.getMonth();
int day1=d1.getDay();
int y2=d2.getYear();
int m2=d2.getMonth();
int day2=d2.getDay();
int exp=Math.abs(y1-y2);
if(m1==m2)
{
if(day2>day1)

```

```
        exp--;
    }
    if(m2>m1)
        exp--;
    if(am>=20000 && exp>=5)
    {
        int dis=(int) (0.20*am);
        tm.put(id,dis);
    }
    else if(am>=20000 && exp<5)
    {
        int dis=(int) (0.1*am);
        tm.put(id,dis);
    }
    else if(am<20000 && exp>=5)
    {
        int dis=(int) (0.15*am);
        tm.put(id,dis);
    }
    else if(am<20000 && exp<5)
    {
        int dis=(int) (0.05*am);
        tm.put(id,dis);
    }
}
catch(Exception e){
    System.out.println(e);
}
}
return tm;
}
```

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:

Campus curiosity

Sample Output 1:

Sample dusinsitz

Sample Input 2-

3333

Sample Output 2:

303

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{  
                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 ≥ 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;  
        }  
    }  
}
```

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

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();  
    }  
}
```

UserMainCode:

```
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 1st argument corresponds to the previous meter reading and the 2nd 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 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 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 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;
}
```


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!

Solution :

```
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(User.printCapitalized(s));
    }
}

import java.util.StringTokenizer;

public class User {
    public static String printCapitalized(String s) {
        StringTokenizer st= new StringTokenizer(s, " ");
        StringBuffer sb= new StringBuffer();
        while(st.hasMoreTokens())
        {
            String s1=st.nextToken();
            String s2=s1.substring(0, 1);
            String s3=s1.substring(1);
            sb.append(s2.toUpperCase());
            sb.append(s3);
            sb.append(" ");
        }
        return sb.toString();
    }
}
```

```

    }
    return sb.toString();
}
}

import java.util.StringTokenizer;
public class UserMainCode {
    public static String changeWord(String s)
    {
        StringTokenizer st=new StringTokenizer(s," ");
        StringBuffer sb=new StringBuffer();
        while(st.hasMoreTokens())
        {
            String s1=st.nextToken();
            sb.append(s1.substring(0,1).toUpperCase());
            sb.append(s1.substring(1));
            sb.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.]

Solution :

```
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.println(User.findMaxDistance(a));  
    }  
}  
  
  
public class User {  
    public static int findMaxDistance(int[] a) {  
  
        int dif,max=0;  
        int n=a.length;  
        for (int i=0;i<n-1;i++)  
        {  
            dif=Math.abs(a[i]-a[i+1]);  
  
            // if(max<dif)  
  
            if(dif>max)  
            {  
                if(a[i+1]>a[i])  
                    max=i+1;  
                else  
                    max=i;  
            }  
        }  
        return max;  
    }  
}
```

```
}
```

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)

Solution:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int a = sc.nextInt();
        int b = sc.nextInt();
```

```

System.out.println(User.addPalindromes(a,b));
}

}

public class User {
public static int addPalindromes(int a, int b) {

    int temp=0,sum=0,r,sum1=0;

    for(int i=a;i<=b;i++)
    {
        temp=i;
        sum=0;
    while(temp>0){
            r=temp%10;
            sum=sum*10+r;
            temp=temp/10;
        }
    if(i==sum)
        sum1=sum1+i;
    }
    return sum1;
}
}

```

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

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String s = sc.next();
        int res = User.validatePAN(s);
        if (res == 1)
            System.out.println("Valid");
        else
            System.out.println("Invalid");
    }
}

public class User {
    public static int validatePAN(String s) {
        int res = 0;
        if (s.length() == 8) {
            if (s.matches("[A-Z]{3}[0-9]{4}[A-Z]{1}"))
                res = 1;
            else
                res = 2;
        }
        return res;
    }
}
```

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**]

Solutions:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int s = sc.nextInt();

        System.out.println(User.getSumOfNfibos(s));
    }
}

public class User {
    public static int getSumOfNfibos(int s) {

        int a = 0, b = 1, c = 0, d = 1;
        for (int i = 3; i <= s; i++) {
            c = a + b;
            a = b;
            b = c;
            d = d + 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

Solutions:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String s=sc.next();
        int res=User.testVowels (s);
        if(res==1)
            System.out.println("Yes");
        else
            System.out.println("No");
    }
}

public class User {
    public static int testVowels (String s){

        int res,count=0;
        String s1="aeiou";
    }
}
```

```

String s2=s.toLowerCase();
for(int i=0;i<s2.length();i++)
{
    for(int j=0;j<s1.length();j++)
    {
        if(s2.charAt(i)==s1.charAt(j))
        {
            count++;
        }
    }
}

if(count==s1.length())
    res=1;
else
    res=2;
return res;
}
}

```

```

public class User {
public static int testOrderVowels(String s1) {

StringBuffer sb = new StringBuffer();
int res = 0;
for (int i = 0; i < s1.length(); i++) {
if (s1.charAt(i) == 'a' || s1.charAt(i) == 'A'
|| s1.charAt(i) == 'e' || s1.charAt(i) == 'E'
|| s1.charAt(i) == 'i' || s1.charAt(i) == 'I'
|| s1.charAt(i) == 'o' || s1.charAt(i) == 'O'
|| s1.charAt(i) == 'u' || s1.charAt(i) == 'U') {
sb.append(s1.charAt(i));
}
}
if (sb.toString().equals("aeiou"))
res = 1;
else
res = 0;
return res;
}
}

```

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

Solution:

```
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String s1=sc.next();
        String s2=sc.next();
        int res=User.compareDashes (s1,s2);
        if(res==1)
            System.out.println("Yes");
        else
            System.out.println("No");
    }
}

import java.util.ArrayList;
```

```

publicclass User {
public static int compareDashes (String s1,String s2) {

    int res=0;
    ArrayList<Integer> a1=new ArrayList<Integer>();
    ArrayList<Integer> a2=new ArrayList<Integer>();
    for(int i=0;i<s1.length();i++)
    {
        if(s1.charAt(i)=='-')
            a1.add(i);
    }
    for(int i=0;i<s2.length();i++)
    {
        if(s2.charAt(i)=='-')
            a2.add(i);
    }
    if(a1.equals(a2))
        res=1;
    else
        res=2;
    return res;
}

}

```

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.HashMap;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String s1=sc.next();
        String s2=sc.next();
        System.out.println(User.extractMax(s1,s2));
    }
}

public class UserMain {
    public static String extractMax(String s1, String s2) {
        StringBuffer sb=new StringBuffer();
        for(int i=0;i<s1.length()-1;i++)
        {
            sb.append(s1.charAt(i));
            sb.append(s2);
        }
        sb.append(s1.charAt(s1.length()-1));
        return sb.reverse().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
2
0
0
```

Solution :

```
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();
        }
        User.removeTens(a);
    }
}

import java.util.ArrayList;

public class User {
    public static int[] removeTens(int[] a) {
        int[] out = new int[a.length];
        int k = 0;
        for (int i = 0; i < a.length; i++) {
            if (a[i] != 10) {
                out[k] = a[i];
                k++;
            }
        }
    }
}
```

```

for(int i=0;i<a.length;i++)
    System.out.println(out[i]);
return out;
}
}

```

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.

Smample Input :

This is a cat

Sample Output :

S\$\$A\$T

Solution :

```

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

publicclass Main {
publicstaticvoid main(String[] args) {
Scanner sc = new Scanner(System.in);
String s= sc.nextLine();
System.out.println(User.getLastLetter(s));

}

import java.util.StringTokenizer;

publicclass User {

```

```

public static String getLastLetter(String s){

    StringTokenizer st= new StringTokenizer(s, " ");
    String s2=st.nextToken();
    StringBuffer sb= new StringBuffer();
    String s3=Character.toUpperCase(s2.charAt(s2.length()-1))+"";
    while(st.hasMoreTokens())
    {
        s2=st.nextToken();
        s3=s3+"$"+Character.toUpperCase(s2.charAt(s2.length()-1));

    }
    returns3;
}

*****
*****



public class UserMain {
    public static String getLastLetter(String s)
    {
        StringTokenizer st= new StringTokenizer(s, " ");
        StringBuffer sb= new StringBuffer();

        String b=st.nextToken();
        sb.append(b.charAt(b.length()-1));

        while(st.hasMoreTokens())
        {
            String a=st.nextToken();
            sb.append("$");
            sb.append(a.charAt(a.length()-1));

        }
        return sb.toString().toUpperCase();
    }
}

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

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

Solutions:

```
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<Integer,String> hm= new HashMap<Integer,String>();
        for(int i=0;i<n;i++)
            hm.put(sc.nextInt(), sc.next());
        System.out.println(User.getMaxKeyValue(hm));
```

```

}

}

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
public static String getMaxKeyValue(HashMap<Integer, String> hm) {
    int max=0;
    String nn=null;
    Iterator<Integer> it = hm.keySet().iterator();
    while(it.hasNext())
    {
        int key=it.next();
        String name=hm.get(key);
        if(key>max)
        {
            key=max;
            nn=name;
        }
    }
    return nn;
}
}

```

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

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();  
  
        String[] s= new String[n];  
  
        for(int i=0;i<n;i++)  
  
            s[i]=sc.next();  
  
        int res=User.validateNumber(s);  
    }  
}
```

```

if(res==1)

    System.out.println("Valid");

else

    System.out.println("invalid");

}

}

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
    public static int validateNumber(String s[]) {
        int res=0;
        int count=0,temp=0;
        String s1=null;
        for(int i=0;i<s.length;i++)
        {
            s1=s[i];
            count=0;
            for(int j=0;j<s1.length();j++)
            {

                if(s1.charAt(j)>='0'&& s1.charAt(j)<='9' || s1.charAt(j)=='.')
                    count++;
            }
            if(count==s1.length())
                temp++;
        }
        if(temp==s.length)
            res=1;
        else
            res=-1;
        return res;
    }
}

```

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

Sample Output 1:

Friday

Solutions :

User :

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

public class User
{
    public static String calculateBornDay(String d) throws ParseException
    {
        SimpleDateFormat sdf= new SimpleDateFormat("MM-dd-yyyy");
        SimpleDateFormat s= new SimpleDateFormat("EEEE");
        Date d1= new Date();
    }
}
```

```
d1= sdf.parse(d);  
  
String day=s.format(d1);  
  
return day;  
}  
}
```

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

```
import java.util.HashMap;  
import java.util.Scanner;  
  
public class Main {  
    public static void main(String[] args) {
```

```

Scanner sc = new Scanner(System.in);
String s1=sc.next();
String s2=sc.next();
System.out.println(User.extractMax(s1,s2));

}

}

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
    public static String extractMax(String s1, String s2) {
        StringTokenizer st= new StringTokenizer(s1,s2);
        int max=0,c=0;
        String str=null;
        while(st.hasMoreTokens())
        {
            String s= st.nextToken();
            c=s.length();
            if(c>max)
            {
                max=c;
                str=s;
            }
        }
        return str;
    }
}

```

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

```
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++) {
            hm.put(sc.next(), sc.next());
        }
        String s = sc.next();
        System.out.println(User.getCapital(hm, s));
    }
}

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;
```

```

public class User {
    public static String getCapital(HashMap<String, String> hm, String s) {
        Iterator<String> it=hm.keySet().iterator();
        StringBuffer sb= new StringBuffer();
        while(it.hasNext())
        {
            String state=it.next();
            String cap=hm.get(state);
            if(state.equalsIgnoreCase(s))
            {
                sb.append(cap).append('$').append(state);
            }
        }
        return sb.toString().toLowerCase();
    }
}

```

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

```
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();
        System.out.println(User.calculateWordSum(s));
    }
}
```

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
    public static int calculateWordSum (String s){
        int sum=0,i=0;
        StringTokenizer st=new StringTokenizer(s, " ");
        String[] s1= new String[st.countTokens()];
        while(st.hasMoreTokens())
        {
            s1[i]=st.nextToken();
            i++;
        }
        if(s1[0].equals(s1[s1.length-1]))
            sum=s1[0].length();
        else
            sum=s1[0].length()+s1[s1.length-1].length();
        return sum;
    }
}
```

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 lowecase.

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
```

```
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();
String[] s= new String[n];
for(int i=0;i<n;i++)
    s[i]=sc.next();
System.out.println(User.matchCharacter(s));
}
```

```

}

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
    public static ArrayList<String> matchCharacter (String[] s) {
        ArrayList<String> a= new ArrayList<String>();
        for(int i=0;i<s.length;i++)
        {
            System.out.println(s[i].charAt(0));
            System.out.println(s[i].charAt(s[i].length()-1));
            if((s[i].charAt(0)=='a'|| s[i].charAt(0)=='e'|| s[i].charAt(0)=='i'|| s[i].charAt(0)=='o'|| s[i].charAt(0)=='u') && (s[i].charAt(s[i].length()-1)=='a'|| s[i].charAt(s[i].length()-1)=='e'|| s[i].charAt(s[i].length()-1)=='i'|| s[i].charAt(s[i].length()-1)=='o'|| s[i].charAt(s[i].length()-1)=='u'))
            {
                a.add(s[i]);
            }
        }
        return a;
    }
}

```

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

```
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);
```

```

int n=sc.nextInt();

HashMap<Integer,String> hm= new HashMap<Integer,String>();

ArrayList<String> a= new ArrayList<String>();

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

    hm.put(sc.nextInt(), sc.next());

a=User.getName(hm);

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

{

    System.out.println(a.get(i));

}

}

}

}

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
public static ArrayList<String> getName(HashMap<Integer, String> hm) {
    ArrayList<String> a= new ArrayList<String>();
    Iterator<Integer> it=hm.keySet().iterator();
    while(it.hasNext())
    {
        int id=it.next();
        String name=hm.get(id);
        for(int i=0;i<name.length();i++)
        {
            if(name.charAt(0)>=97 && name.charAt(0)<=122 &&
                name.charAt(name.length()-1)>=65 &&
            name.charAt(name.length()-1)<=96)
            {
                if(name.charAt(i)>='0' && name.charAt(i)<='9')
                {
                    a.add(name);
                }
            }
        }
    }
}

```

```
        }
    }
}
}
return a;
}
}
```

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

```
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);
        int n=sc.nextInt();
        ArrayList<Integer> a= new ArrayList<Integer>();
        for(int i=0;i<n*2;i++)
            a.add(sc.nextInt());
        System.out.println(User.getYear(a));
    }
}

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
    public static int getYear(ArrayList<Integer> a) {
        int year=0;
```

```

int max=0;
for(int i=1;i<a.size();i=i+2)
{
    int x=a.get(i);
    if(x>max)
    {
        max=x;
        year=a.get(i-1);
    }
}
return year;
}

```

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

```
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);
        int n=sc.nextInt();
        System.out.println(User.addNumbers(n));
    }
}

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
    public static int addNumbers(int n) {
        int c=0,sum=0;
        for (int i = 1; i <= n; i++)
        {
            c=0;
            for(int j=1;j<=i;j++)
            {
                if(i%j==0)
                    c++;
                if (c==2)
                    ;
                else
                    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

```
import java.text.ParseException;  
  
import java.util.ArrayList;  
  
import java.util.HashMap;  
  
import java.util.Scanner;  
  
  
public class Main {  
public static void main(String[] args) throws ParseException {  
Scanner sc = new Scanner(System.in);  
String n=sc.next();  
System.out.println(User.convertDateFormat(n));  
}
```

```

}

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
    public static String convertDateFormat(String n) throws ParseException{
        SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
        Date d= sdf.parse(n);
        SimpleDateFormat sdf1=new SimpleDateFormat("dd-MM-yyyy");
        String s=sdf1.format(d);
        return s;
    }
}

```

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

```
public static void main(String[] args) {  
  
    Scanner sc=new Scanner(System.in);  
    String s=sc.nextLine();  
    SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");  
    sdf.setLenient(false);  
    int res=0;  
    if(s.matches("[0-9]{2}(/)[0-9]{2}(/)[0-9]{4}"))  
    {  
  
        try {  
            Date d=sdf.parse(s);  
            res=1;  
        } catch (ParseException e) {  
            res=-1;  
        }  
  
        System.out.println(res);  
    }  
}
```

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

```
import java.text.ParseException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) throws ParseException {
        Scanner sc = new Scanner(System.in);
        String n=sc.next();
        System.out.println(User.convertFormat(n));
    }
}

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
    public static String convertFormat(String s) throws ParseException{
        StringTokenizer st=new StringTokenizer(s,"-");
        int i=0;
        String[] s1=new String[st.countTokens()];
        while(st.hasMoreTokens())
        {
            s1[i]=st.nextToken();
            i++;
        }
        StringBuffer sb=new StringBuffer();
        sb.append(s1[0].substring(0,2));
        sb.append("-");
        sb.append(s1[0].substring(2)).append(s1[1].substring(0,1));
        sb.append("-");
        sb.append(s1[1].substring(1)).append(s1[2].substring(0,1));
        sb.append("-");
        sb.append(s1[2].substring(1));
    }
}
```

```

        return sb.toString();
    }

}

import java.util.StringTokenizer;
public class UserMainCode {
    public static String convertFormat(String s)
    {
        StringBuffer sb=new StringBuffer();
        StringTokenizer st=new StringTokenizer(s,"_");
        String s1=st.nextToken();
        String s2=st.nextToken();
        String s3=st.nextToken();
        sb.append(s1.substring(0,2));
        sb.append("_");
        sb.append(s1.substring(s1.length()-1));
        sb.append(s2.substring(0,1));
        sb.append("_");
        sb.append(s2.substring(1));
        sb.append(s3.substring(0,1));
        sb.append("_");
        sb.append(s3.substring(1));
        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

```

import java.text.ParseException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) throws ParseException {
        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 x=sc.nextInt();
        System.out.println(User.addAndReverse(a,x));
    }
}

```

```

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
    public static int addAndReverse(int[] a, int x) {
        int sum=0;
        int rev=0, r=0;
        for(int i=0;i<a.length;i++) {
            if(x<a[i])
                sum=sum+a[i];
        }

        while(sum!=0)
        {
            r=sum%10;
            rev=rev*10+r;
            sum=sum/10;
        }

        return rev;
    }
}

```

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

```
import java.text.ParseException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Scanner;

public class Main {
    public static void main(String[] args) throws ParseException {
        Scanner sc = new Scanner(System.in);
        String n=sc.next();

        System.out.println(User.nextYearDay(n));

    }
}

import java.text.ParseException;
```

```

import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
    public static String nextYearDay(String s) throws ParseException {
        SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
        SimpleDateFormat sdf1=new SimpleDateFormat("EEEE");
        Date d= sdf.parse(s);
        Calendar c=Calendar.getInstance();
        c.setTime(d);
        c.add(Calendar.YEAR,1);
        Date year=c.getTime();
        String day=sdf1.format(year);

        return day;
    }
}

```

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

```
import java.text.ParseException;
```

```

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

public class Main {
    public static void main(String[] args) throws ParseException {
        Scanner sc = new Scanner(System.in);
        int n=sc.nextInt();

        System.out.println(User.getSumOfSquaresOfDigits(n));

    }
}

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
    public static int getSumOfSquaresOfDigits(int n) {
        int sum=0,r=0;
        while(n!=0)
        {
            r=n%10;
            sum=sum+(r*r);
            n=n/10;
        }
        return 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

```
import java.text.ParseException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) throws ParseException {
        Scanner sc = new Scanner(System.in);
        int n=sc.nextInt();

        int res=User.sumOfOddEvenPositioned(n);
        if(res==1)
            System.out.println("yes");
        else
            System.out.println("no");

    }
}
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
```

```
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
    public static int sumOfOddEvenPositioned(int n) {
        int even=0,odd=0;
        int res=0,r=0,m=0;
        int j=0,count=0;
        int n1=n;
        while(n>0)
        {
            n=n/10;
            count++;
        }
        System.out.println(count);
        int[] a=new int[count];
        while(n1!=0)
        {
            r=n1%10;
            a[j]=r;
            j++;
            n1=n1/10;
        }
        int[] b=new int[j];
        for(int k=j-1;k>=0;k--)
        {
            b[m]=a[k];
            m++;
        }

        for(int i=0;i<m;i++)
        {
            System.out.println("a:"+b[i]);
            if(i%2==0)
                even=even+b[i];
            else
                odd=odd+b[i];
        }
        System.out.println(even);
        System.out.println(odd);
        if(even==odd)
            res=1;
        else
            res=-1;
        return res;
    }
}
```

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
```

```
publicclass Main {
publicstaticvoid main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);
int n=sc.nextInt();
ArrayList<Integer> a=new ArrayList<Integer>();
ArrayList<Integer> res=new ArrayList<Integer>();
for(int i=0;i<n;i++)
```

```

        a.add(sc.nextInt());
res=User.removeMultiplesOfThree(a);
for(int i=0;i<res.size();i++)
    System.out.println(res.get(i));
}
}

public class User {
public static ArrayList<Integer> removeMultiplesOfThree(ArrayList<Integer> a)
{
    ArrayList<Integer> b=new ArrayList<Integer>();
    for(int i=0;i<a.size();i++)
    {
        int d=a.get(i);
        if(d%3!=0)
        {
            b.add(a.get(i));
        }
    }
    return b;
}
}

```

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
```

```
import java.text.ParseException;
```

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

```
import java.util.HashMap;
```

```
import java.util.Iterator;
```

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

```
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(User.getSubstring(s1, s2));
```

```
}
```

```
}
```

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
    public static int getSubstring (String s1,String s2) {
        int count=0;
        int n=s1.length()-(s2.length()-1);
        for(int i=0;i<n;i++)
        {
            String s3=s1.substring(i,i+(s2.length()));
            if(s2.equals(s3))
                count++;
        }
        return count;
    }
}
```

```

public class User {
    public static int getSubstring (String s1, String s2) {
        int count=0;
        int n=s1.length()-(s2.length()-1);
        int s2l=s2.length();
        System.out.println(n);
        for(int i=0;i<n;i++)
        {
            String s3=s1.substring(i,i+s2l);
            if(s2.equals(s3))
                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

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;

public class User {
    public static int getLuckySum (int a,int b, int c) {
```

```
int res=0;
if(a==13)
    res=c;
else if(b==13)
    res=a;
else if(c==13)
    res=a+b;
else
    res=a+b+c;

return res;

}
```

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

```
import java.text.ParseException;  
  
import java.util.ArrayList;  
  
import java.util.HashMap;  
  
import java.util.Iterator;  
  
import java.util.Scanner;  
  
  
public class Main {
```

```
public static void main(String[] args) throws ParseException {
    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();
    boolean b=User.checkTriplets(a);
    System.out.println(b);
}

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.StringTokenizer;
```

```
public class User {
    public static boolean checkTriplets (int a[]) {
        boolean b=false;
        int c=0;
```

```

for(int i=0;i<a.length-2;i++)
{
    if(a[i]==a[i+1]&&a[i+1]==a[i+2])
        b=true;
    else
        b=false;
}
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

```
import java.text.ParseException;
```

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

```
import java.util.HashMap;
```

```
import java.util.Iterator;
```

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

```
public class Main {
```

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

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

```
        String s=sc.next();
```

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

```
        String res=User.repeatFirstThreeCharacters(s,n);
```

```
for(int i=0;i<n;i++)  
    System.out.print(res);  
  
}  
  
}
```

```
import java.text.ParseException;  
  
import java.text.SimpleDateFormat;  
  
import java.util.ArrayList;  
  
import java.util.Calendar;  
  
import java.util.Date;  
  
import java.util.HashMap;  
  
import java.util.Iterator;  
  
import java.util.StringTokenizer;
```

```
public class User {  
  
    public static String repeatFirstThreeCharacters(String s, int n) {  
  
        String front=null;  
  
        if(s.length()>=3)  
        {  
            front=s.substring(0,3);  
        }  
  
        String result="";  
  
        for(int i=0;i<n;i++)  
        {  
            result+=front;  
        }  
  
        return result;  
    }  
}
```

```
    }  
  
    else  
  
        front=s;  
  
    return front;  
  
}  
  
}
```

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

```
import java.text.ParseException;
```

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) throws ParseException {
        Scanner sc = new Scanner(System.in);
        int n=sc.nextInt();
        String[] a= new String[n];
        for(int i=0;i<n;i++)
            a[i]=sc.next();

        String res[]={};
        res=User.orderElements(a);
        for(int i=0;i<res.length;i++)
            System.out.println(res[i]);
    }
}

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collections;
```

```
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedHashSet;
import java.util.StringTokenizer;

public class User {
    public static String[] orderElements(String[] s) {
        LinkedHashSet<String> lhs=new LinkedHashSet<String>();
        for(int i=0;i<s.length;i++)
        {
            lhs.add(s[i]);
        }
        String[] a= new String[lhs.size()];
        for(int i=0;i<s.length;i++)
        {
            lhs.toArray(a);
        }
        Arrays.sort(a);
        return a;
    }
}
```

}

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

```
import java.text.ParseException;  
  
import java.util.ArrayList;  
  
import java.util.HashMap;
```

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

public class Main {
    public static void main(String[] args) throws ParseException {
        Scanner sc = new Scanner(System.in);
        String a= sc.next();

        boolean b=User.CheckID(a);

        System.out.println(b);
    }
}

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collections;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedHashSet;
import java.util.StringTokenizer;
```

```

public class User {

    public static boolean CheckID (String s) {

        boolean b=false;

        if(s.matches("(CPT-)[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$
```

```
import java.text.ParseException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) throws ParseException {
        Scanner sc = new Scanner(System.in);
        int n=sc.nextInt();
        String[] a=new String[n];
        for(int i=0;i<n;i++)
            a[i]=sc.next();
        int s=sc.nextInt();
    }
}
```

```
System.out.println(User.formString(a,s));

}

}

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Calendar;
import java.util.Collections;
import java.util.Date;
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedHashSet;
import java.util.StringTokenizer;

public class User {
    public static String formString(String s[],int n) {
        StringBuffer sb=new StringBuffer();
        for(int i=0;i<s.length;i++)
        {
            String st=s[i];
            if(st.length()>=n)
```

```

    {
        sb.append(st.charAt(n-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

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

```
public class Main {
```

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

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

```
        String s=sc.next();
```

```
        System.out.println(User.validate (s));
```

```

}

}

public class User {

    public static boolean validate (String s) {

        boolean b= false;

        if(s.length()==4)

        {

            if(s.matches("[a-zA-Z0-9]{1}(R)[0-9]{1}[A-Za-z0-9]{1}"))

                b=true;

            else

                b=false;

        }

        return b;

    }

}

```

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

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

```
public class Main {
```

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

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

```

int s=sc.nextInt();

System.out.println(User.validate (s));

}

}

public class User {

public static boolean validate (int s) {

boolean b= false;

if(s>0)

{

if(s>=21&&s<=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

```
import java.text.ParseException;  
  
import java.util.ArrayList;  
  
import java.util.HashMap;  
  
import java.util.Iterator;  
  
import java.util.Scanner;
```

```
public class Main {  
  
    public static void main(String[] args) throws ParseException {  
  
        Scanner sc = new Scanner(System.in);  
  
  
        String s=sc.next();  
  
        System.out.println(User.validatePhone(s));  
  
  
    }  
  
}
```

```
public class User {  
  
    public static boolean validatePhone(String s) {  
  
        boolean b= false;  
  
        if(s.length()==10)  
  
        {  
  
            if(s.matches("(0){2}[0-9]{8}"))  
  
                b=false;  
  
            else if(s.matches("[0-9]{10}"))  
  
                b=true;  
  
            else  
  
                ;  
  
        }  
  
        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

```
publicclass Main {  
publicstaticvoid main(String[] args) throws ParseException {  
Scanner sc = new Scanner(System.in);  
  
String s1=sc.next();  
char s2=sc.next().charAt(0);
```

```

String res[] = User.manipulateLiteral (s1,s2);
for(int i=0;i<res.length;i++)
    System.out.println(res[i]);

}

}

public class User {
public static String[] manipulateLiteral(String s1,char s2) {
    String ss=Character.toString(s2);
    StringTokenizer st=new StringTokenizer(s1,ss);
    ArrayList<String> a=new ArrayList<String>();
    while(st.hasMoreTokens())
    {
        StringBuffer sb=new StringBuffer();
        sb.append(st.nextToken().toLowerCase());
        a.add(sb.reverse().toString());
    }
    String[] s=new String[a.size()];
    for(int i=0;i<a.size();i++)
        s[i]=(String)a.get(i);

    return s;
}
}

```

```

import java.util.ArrayList;
import java.util.StringTokenizer;

public class User {
public static String[] manipulateLiteral(String s1,char s2) {
    String ss=String.valueOf(s2);
    StringTokenizer st=new StringTokenizer(s1,ss);
    ArrayList<String> a=new ArrayList<String>();

    while(st.hasMoreTokens())
    {
        StringBuffer sb=new StringBuffer();
        sb.append(st.nextToken());
        a.add(sb.reverse().toString().toLowerCase());
    }
    int d=a.size();
    System.out.println(d);
    String[] s=new String[d];
    for(int i=0;i<a.size();i++)
    {
        s[i]=a.get(i);
    }
}

```

```
    return s;
}
}
```

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.Scanner;

public class Main {
    public static void main(String[] args) throws ParseException {
        Scanner sc = new Scanner(System.in);

        String s1=sc.next();

        System.out.println(User.tellVowelCount(s1));
    }
}

public class User {
    public static int tellVowelCount(String s1) {
        int count=0;
        String s="aeiou";
        String ss="AEIOU";
        for(int i=0;i<s1.length();i++)
        {
            for(int j=0;j<s.length();j++)
            {
                if(s1.charAt(i)==s.charAt(j) || s1.charAt(i)==ss.charAt(j))
)
                    count++;
            }
        }
        return count;
    }
}
```

```
public static int tellVowelCount(String s1) {
    int count=0;
    for(int i=0;i<s1.length();i++)
    {
        if(s1.charAt(i)=='a' || s1.charAt(i)=='e' ||
s1.charAt(i)=='i'
            || s1.charAt(i)=='o' || s1.charAt(i)=='u' ||
s1.charAt(i)=='A' || s1.charAt(i)=='E' ||
s1.charAt(i)=='I' ||
s1.charAt(i)=='O' || s1.charAt(i)=='U' )

        {
            count++;
        }
    }
    return count;
}
```

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
```

```
publicclass Main {
publicstaticvoid main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);
int n=sc.nextInt();
String[] a= new String[n];
for(int i=0;i<n;i++)
    a[i]=sc.next();
String[] res=User.sortArray(a);
for(int i=0;i<res.length;i++)
    System.out.println(res[i]);
```

```

}
}

public class User {
    public static String[] sortArray (String s[]) {
        String[] a=new String[s.length];
        for(int i=0;i<s.length;i++)
        {
            a[i]=s[i].toLowerCase();
        }
        Arrays.sort(a);
        return a;
    }
}

```

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

```
publicclass Main {  
publicstaticvoid main(String[] args) throws ParseException {  
Scanner sc = new Scanner(System.in);  
int n=sc.nextInt();  
int[] a= newint[n];  
for(int i=0;i<n;i++)  
    a[i]=sc.nextInt();  
System.out.println(User.calculateMedian (a));  
}  
}
```

```
publicclass User {  
  
publicstaticint calculateMedian(int s[]) {  
    double med=0;  
    double avg=0;  
    Arrays.sort(s);  
    int mid=s.length/2;  
    if(s.length%2!=0)  
        med=s[mid];  
    else  
    {
```

```

        avg=(double) (s[mid]+s[mid-1])/2;
        System.out.println(avg);
        med=Math.ceil(avg);
    }
    return (int)med;
}
}

```

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

```
publicclass Main {
publicstaticvoid main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);
int n=sc.nextInt();
int[] a= newint[n];
for(int i=0;i<n;i++)
    a[i]=sc.nextInt();
boolean b=User.calculateMedian (a);
System.out.println(b);

}

publicclass User {
publicstaticboolean calculateMedian(int s[]) {
    int[] a={1,2,3};
    int n=s.length-(a.length-1);
    boolean b=false;
    for(int i=0;i<n;i++)
    {
        if(s[i]==a[0] )
        {
            if(s[i+1]==a[1])
            {
                if(s[i+2]==a[2])
                {
                    b=true;
                    break;
                }
                else
                    b=false;
            }
            else
                b=false;
        }
        else
            b=false;
    }
    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

```
publicclass Main {  
publicstaticvoid main(String[] args) throws ParseException {  
Scanner sc = new Scanner(System.in);  
String n=sc.next();  
  
boolean b=User.scanStarNeighbors(n);  
System.out.println(b);  
}  
}
```

```

public class User {
    public static boolean scanStarNeighbors(String s) {
        StringTokenizer st=new StringTokenizer(s, "*");
        boolean b=false;
        while(st.hasMoreTokens())
        {
            String s1=st.nextToken();
            String s2=st.nextToken();
            if(s1.charAt(s1.length()-1)==s2.charAt(0))
            {
                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

```
publicclass Main {
publicstaticvoid main(String[] args) throws ParseException {
Scanner sc = new Scanner(System.in);
String s1=sc.nextLine();
String s2=sc.next();
int b=User.countWords (s1,s2);
System.out.println(b);

}

}

publicclass User {
publicstaticint countWords (String s1,String s2) {
 StringTokenizer st=new StringTokenizer(s1, " ");
 int c=0;
 while(st.hasMoreTokens ())
 {
     String s3=st.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:

2

Sample Input 2:

Catcowcat

catp

Sample Output 2:

0

```
public class User {
    public static int scanStarNeighbors(String s1, String s2) {
        int ls1 = s1.length();
        int ls2 = s2.length();
        int n = ls1 - (ls2 - 1);
        System.out.println(n);
        int ct = 0;
        for (int i = 0; i < n; i++) {
            String ss = s1.substring(i, i + (ls2));
            if (s2.equals(ss))
                ct++;
        }
        return ct;
    }
}
```

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 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

```
import java.text.ParseException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
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();
int n=sc.nextInt();
System.out.println(User.findFruitName(s1,n));
}

import java.util.ArrayList;
import java.util.Arrays;
import java.util.LinkedHashSet;
import java.util.StringTokenizer;
public class User {
public static String findFruitName(String s1,int n) {
    StringTokenizer st=new StringTokenizer(s1,",");
    int c=0,i=0;
    String ss=null;

    String[] s=new String[st.countTokens()];

    while(st.hasMoreTokens())
    {
        s[i]=st.nextToken();
        i++;
    }
    if(i>n)
    {
        ss=s[n-1];
    }
    else
    {
        ss=s[i-1];//last element display
    }

    return ss;
}
}

```

48. Proper 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

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

```
import java.util.StringTokenizer;
public class UserMainCode {
    public static String changeWord(String s)
    {
        StringTokenizer st=new StringTokenizer(s, " ");
        StringBuffer sb=new StringBuffer();
        while(st.hasMoreTokens())
        {
            String s1=st.nextToken();
            sb.append(s1.substring(0,1).toUpperCase());
            sb.append(s1.substring(1));
            sb.append(" ");
        }
        return sb.toString();
    }
}
```

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 $1+2+5$ is not equal to 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

```
publicclass User {
    publicstaticboolean scanStarNeighbors(intn) {
        boolean b;

        int sum=0;

        for(int i=1;i<n;i++)
        {
            if(n%i==0)
            {
                sum=sum+i;
                System.out.println(sum);
            }
        }
        if(sum==n)
        {
            b=true;
        }
        else
        {
            b=false;
        }

        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

Sample Output 3:

2:2

```
import java.util.*;  
  
public class Main {  
  
    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;  
    }  
}
```

```

}

import java.util.StringTokenizer;
public class User{
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(n2-1)=='0')
n2=s2.length()-1;
//String s3=String.valueOf(n1)+":"+String.valueOf(n2);
StringBuffer sb=new StringBuffer();
sb.append(n1).append(":").append(n2);
return sb.toString();
}
}

```

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 and 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
```

```
class Main
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        sc.nextLine();
        HashMap<String, String> hm=new HashMap<String, String>();
    }
}
```

```
for(int i=0;i<n;i++)  
{  
    hm.put(sc.nextLine(),sc.nextLine());  
}  
  
String b=sc.nextLine();  
  
  
HashMap<String,String> op=new HashMap<String,String>();  
op=MainClass.obtainDesig(hm,b);  
Iterator<String> itr=op.keySet().iterator();  
while(itr.hasNext())  
{  
    String key=itr.next();  
    System.out.println(key);  
    String value=hm.get(key);  
    System.out.println(value);  
}  
}  
}}
```

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedHashMap;

public class MainClass {

    public static LinkedHashMap<String, String> obtainDesig(HashMap<String, String> hm, String s)

    {
        LinkedHashMap<String, String> op=new LinkedHashMap<String, String>();
        Iterator<String> itr=hm.keySet().iterator();

        while(itr.hasNext())
        {
            String key=itr.next();
            String value=hm.get(key);
            if(s.equals(value))
            {
                op.put(key,value);
            }
        }
        return op;
    }
}
```

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.LinkedHashMap;
```

```
import java.util.Map;
```

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

```
public class Main
```

```
{
```

```
    public static void main(String[]arg)
```

```
{
```

```
        LinkedHashMap<String,Double>hm=new LinkedHashMap<String,Double>();
```

```
        LinkedHashMap<String,String>hm1=new LinkedHashMap<String,String>();
```

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

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

```
        for(int i=0;i<n;i++)
```

```
{
```

```
            String s=sc.next();
```

```
            double d=sc.nextDouble();
```

```
hm.put(s,d);

}

LinkedHashMap<String,String>hm2=UserMainCode.dis(hm);

for(Map.Entry<String,String>entry:hm2.entrySet())

{

System.out.println(entry.getKey());

System.out.println(entry.getValue());

}}}
```

```
import java.util.LinkedHashMap;

import java.util.Map;

class UserMainCode

{

public static LinkedHashMap<String,String>dis(LinkedHashMap<String,Double>h1)

{

    LinkedHashMap<String,String>h2=new LinkedHashMap<String,String>();

    for(Map.Entry m:h1.entrySet())

    {

        double d=(Double)m.getValue();

        if(d>60)

    {
```

```
String s=(String)m.getKey();
```

```
h2.put(s,"pass");
```

```
}
```

```
else
```

```
{
```

```
String s=(String)m.getKey();
```

```
h2.put(s,"fail");
```

```
}
```

```
}
```

```
return h2;
```

```
}
```

```
}
```

```
(Or)
```

```
import java.util.*;
public class Main {
public static void main(String[] args) {
    Scanner sc=new Scanner(System.in);

    int n=sc.nextInt();
    LinkedHashMap<String,Float> ip=new LinkedHashMap<String,Float>();
    for(int i=0;i<n;i++)
    {
        ip.put(sc.next(),sc.nextFloat());
    }

    LinkedHashMap<String,String> op=new LinkedHashMap<String,String>();
    op=User.noOfDigits(ip);
    Iterator<String> itr= op.keySet().iterator();
    while(itr.hasNext())
    {
        String key=itr.next();
        System.out.println(key);
        String value=op.get(key);
        System.out.println(value);

    }
}
```

```

import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedHashMap;
public class User{
public static LinkedHashMap<String, String> noOfDigits(HashMap<String, Float>
hm) {

    LinkedHashMap<String, String> op=new LinkedHashMap<String, String>();
    Iterator<String> itr=hm.keySet().iterator();
    String res=null;
        for(int i=0;i<hm.size();i++)
        {
    while(itr.hasNext())
    {
        String key=itr.next();

        float value=hm.get(key);
        if(value>=60)
            res="pass";
        else
            res="fail";
        op.put(key,res);

    }
}

return op;
}
}

import java.util.HashMap;
import java.util.Iterator;
import java.util.LinkedHashMap;
public class User{
public static LinkedHashMap<String, String> noOfDigits(HashMap<String, Float>
hm) {

    LinkedHashMap<String, String> op=new LinkedHashMap<String, String>();
    Iterator<String> itr=hm.keySet().iterator();
    String res=null;
        while(itr.hasNext())
    {
        String key=itr.next();

        float value=hm.get(key);
        if(value>=60)
            res="pass";
        else
            res="fail";
        op.put(key,res);

    }
}
}

```

```
    return op;
}
}
```

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 UserMainCode {
    public static void main(String[] args)
    {
        String str=new String();
        Scanner sc=new Scanner(System.in);
        str=sc.nextLine();
        SimpleDateFormat sdf=new SimpleDateFormat("MM/dd/yyyy");
        sdf.setLenient(false);
        try
        {
            Date d1=sdf.parse(str);
            System.out.println("TRUE");
        }
        catch(Exception e)
        {
            System.out.println("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 Format:

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.print(UserMainCode.getvalues(s,s1));
    }
}

import java.util.Calendar;

import java.util.Date;

public class UserMainCode {
    public static boolean getvalues(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 array as 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.ArrayList;  
  
import java.util.Scanner;  
  
  
class Main  
{  
  
    public static void main(String[] args)  
    {  
  
        Scanner sc=new Scanner(System.in);  
  
        int n=sc.nextInt();  
  
        sc.nextLine();  
  
        ArrayList<String> aa=new ArrayList<String>();  
  
        for (int i=0; i<n; i++)  
        {  
            String s=sc.nextLine();  
            aa.add(s);  
        }  
  
        System.out.println(aa);  
    }  
}
```

```
for(int i=0;i<n;i++)
{
    aa.add(sc.nextLine());
}

String a[]=MainClass.convertToString( aa);

for(int i=0;i<a.length;i++)
{
    System.out.println(a[i]);
}

}

import java.util.ArrayList;
import java.util.Collections;

public class MainClass {

    public static String[] convertToString(ArrayList<String> a1)
    {
        Collections.sort(a1);// uses to sort arraylist string
    }
}
```

```

        String a[]=new String[a1.size()];
        a1.toArray(a);
        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 **getStatId** 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

```
import java.util.*;
public class Main
{
    public static void main(String[] args)
    {

        Scanner sc=new Scanner(System.in);

        int n=sc.nextInt();
        String s1[]={};
        for(int i=0;i<n;i++)
        {
            s1[i]=sc.next();
        }

        LinkedHashMap<String, String> ip=new
        LinkedHashMap<String, String>();
        ip=User.Method(s1);
        Iterator<String> itr=ip.keySet().iterator();

        //while(itr.hasNext())
        for(int i=0;i<ip.size();i++)
        {
            String key=itr.next();
            String value=ip.get(key);
            System.out.println(value+":"+key);
        }

    }
}

import java.util.LinkedHashMap;
public class User
{
    public static LinkedHashMap<String, String> Method(String[] s1)
    {
        LinkedHashMap<String, String> op=new LinkedHashMap<String, String>();

        for(int i=0;i<s1.length;i++)
        {
            StringBuffer sb=new StringBuffer();
            StringBuffer key=sb.append(s1[i].substring(0,3)).append(":");
            op.put(s1[i],key.toString().toUpperCase());
        }
    }
}
```

```
    return op;
}
}
```

(or)

STATE id

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.HashMap;

public class Main {

    public static void main(String[] args) {
        BufferedReader br=new BufferedReader(new
InputStreamReader(System.in));
        try
        {
            int n=Integer.parseInt(br.readLine());
            String[] input=new String[n];
            for(int i=0;i<n;i++)
            {
                input[i]=br.readLine();
            }
            HashMap<String, String>hm=UserMainCode.costEst
imator(input);
            for(int i=0;i<n;i++)
            {
                String s=input[i];
                String key=hm.get(s);
                System.out.println(key+":"+s);
            }
        }

        catch(Exception e)
        {
        }
    }
}
```

```

}

import java.util.HashMap;

public class UserMainCode {
    public static HashMap<String, String>
costEstimator(String[] name)
    {
        int n=name.length;
        HashMap<String, String> hm=new HashMap<String,
String>();
        for(int i=0;i<n;i++)
        {
            String sub=name[i].substring(0, 3);
            hm.put(name[i], sub.toUpperCase());
        }
        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
```

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        List<String> l1=new ArrayList<String>();
        l1.add("Apple");
        l1.add("Chery");
        l1.add("Grapes");
        List<String> l2=new ArrayList<String>();
        l2.add("Orange");
        l2.add("Mango");
        l2.add("Melon");
        l2.add("Apple");
        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(s1.length()-1)!= 'a' && s1.charAt(s1.length()-1)!= 'A' &&
s1.charAt(s1.length()-1)!= 'e' && s1.charAt(s1.length()-1)!= 'E')
            l3.add(s1); }

        for(int i=0;i<l2.size();i++){
            String s1=l2.get(i);
            if(s1.charAt(0)!= 'm' && s1.charAt(0)!= 'M' && s1.charAt(0)!= 'a' &&
s1.charAt(0)!= 'A')
                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;
    }
}

```

```

import java.util.*;
public class Main {
public static void main(String[] args) {
    List<String> l1=new ArrayList<String>();
    Scanner sc=new Scanner(System.in);
    int n=sc.nextInt();
    for(int i=0;i<n;i++)
    {
        l1.add(sc.next());
    }
    List<String> l2=new ArrayList<String>();

    int n1=sc.nextInt();
    for(int i=0;i<n1;i++)
    {
        l2.add(sc.next());
    }
}

```

```

        String[] s2=User.fruitsList(l1,l2);
        // for(int i=0;i<s2.length;i++)
        //     System.out.println(s2[i].toString());
    for(String s3:s2)
        System.out.println(s3);

    }
}

public class User
{
    public static String[] fruitsList(List<String> l1, List<String> l2) {
        ArrayList<String> l3=new ArrayList<String>();
        for(int i=0;i<l1.size();i++)
        {
            String s1=l1.get(i);
            int len=s1.length();
            if(s1.charAt(len-1) !='a'&& s1.charAt(len-1) !='A'
                && s1.charAt(len-1) !='e'&& s1.charAt(len-1) !='E')

                l3.add(s1);
        }

        for(int i=0;i<l2.size();i++)
        {
            String s1=l2.get(i);
            if(s1.charAt(0) !='m'&& s1.charAt(0) !='M'&& s1.charAt(0) !='a'
                && s1.charAt(0) !='A')
                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;
    }
}

]

/*

```

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
```

```
import java.util.*;  
public class Main  
{  
    public static void main(String[] args)  
    {  
        int n,m;  
        Scanner sc = new Scanner(System.in);  
        n = sc.nextInt();  
        ArrayList<Integer> a1 = new ArrayList<Integer>();  
        for(int i=0;i<n;i++)  
        {  
  
            a1.add(sc.nextInt());  
        }  
        m = sc.nextInt();  
        ArrayList<Integer> a2 = new ArrayList<Integer>();  
        for(int i=0;i<m;i++)  
        {  
  
            a2.add(sc.nextInt());  
        }  
        int[] result = User.arrayListSubtractor(a1, a2);  
        Arrays.sort(result);  
        for(int i=0;i<result.length;i++)  
            System.out.println(result[i]);  
    }  
}
```

```
import java.util.*;

public class User
{

    public static int[] arrayListSubtractor(ArrayList<Integer>
arrlist1,ArrayList<Integer> arrlist2)
    {
        TreeSet<Integer> ts1=new TreeSet<Integer>();
        TreeSet<Integer> ts2=new TreeSet<Integer>();
        TreeSet<Integer> ts3=new TreeSet<Integer>();
        ArrayList<Integer> aa=new ArrayList<Integer>();
        for(int i=0;i<arrlist1.size();i++)
            ts1.add(arrlist1.get(i));

        for(int i=0;i<arrlist2.size();i++)
            ts2.add(arrlist2.get(i));

        ts1.addAll(ts2);

        for(int i=0;i<arrlist1.size();i++)
        {
            for(int j=0;j<arrlist2.size();j++)
            {
                if(arrlist1.get(i)==arrlist2.get(j))
                    ts3.add(arrlist1.get(i));
            }
        }

        ts1.removeAll(ts3);
        aa.addAll(ts1);
        int res[]={new int[aa.size()]};
        for(int i=0;i<res.length;i++)
            res[i]=aa.get(i);
        return res;
    }

}
```

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 calculate 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
```

```
import java.util.*;

public class UserMainCode {

    public static void main(String[] args) {

        HashMap<String, String> m1=new HashMap<String, String>();

        m1.put("monitor", "1200.36");

        m1.put("mouse","100.42");

        m1.put("speaker", "500.25");

        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;

}}
```

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

```
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 User {

    public static void main(String[] args) throws IOException, ParseException {
        // TODO Auto-generated method stub

        String s1="10-02-2012";
        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

```

import java.text.ParseException;

import java.util.*;
public class Main
{
    public static void main(String[] args) throws ParseException {
        Scanner sc=new Scanner(System.in);
        String s=sc.nextLine();
        System.out.println(User.leapYear(s));
    }
}

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

public class User
{
    public static boolean leapYear(String s) throws ParseException {
        SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
        GregorianCalendar g=new GregorianCalendar();

        Calendar cal=Calendar.getInstance();
        Date d1=sdf.parse(s);
        cal.setTime(d1);
        int n=cal.get(Calendar.YEAR);

        boolean b=g.isLeapYear(n);
        return 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

```
import java.util.*;  
  
public class Main {  
  
    public static void main(String[] args) {  
  
        String s1="You are toooo 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;  
    }  
}
```

```
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
```

```
public class Main {  
    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);  
    }  
}
```

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
```

```
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.display(a));
    }
}
```

```
import java.util.Iterator;
import java.util.LinkedHashSet;

public class UserMainCode {
    public static int display(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

```

public class UserMainCode

{
    public static int display(String s)

    {
        if(s.matches(".*[0-9]{1,}.*") && s.matches(".*[@#$]{1,}.*") && s.length()>=8 &&
s.matches(".*[A-Z]{1,}.*") && s.matches(".*[a-z]{1,}.*"))

            return 1;

        else

            return -1;
    }
}

import java.util.*;
public class Main
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        String s=sc.nextLine();
        System.out.println(User.leapYear(s));
    }
}

public class User{
    public static int leapYear(String s)
    {

if(s.matches(".*[0-9]{1,}.*")
    &&s.matches(".*[@#$%&]{1,}.*") &&s.length()>=8 &&
s.matches(".*[A-Z]{1,}.*") &&s.matches(".*[a-z]{1,}.*"))
return 1;
else
return -1;
    }
}

```

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

```
import java.util.HashMap;  
  
import java.util.Iterator;  
  
import java.util.LinkedHashMap;  
  
  
import java.util.Scanner;
```

```
public class kapes3 {  
  
    public static void main(String []args){  
  
        Scanner sc=new Scanner(System.in);  
  
        int s=Integer.parseInt(sc.nextLine());  
  
        int []a=new int[s];  
  
        for(int i=0;i<s;i++)  
  
        {  
  
            a[i]=sc.nextInt();  
  
        }  
  
        LinkedHashMap<Integer,Integer>hm2=new LinkedHashMap<Integer,Integer>();  
  
        hm2=kapes4.display(a);  
  
        Iterator<Integer> it=hm2.keySet().iterator();  
  
        for(int i=0;i<s;i++)  
  
        {  
  
            int n=it.next();  
  
            int fac=hm2.get(n);  
  
            System.out.println(n+":"+fac);  
  
        }  
  
    }  
  
}  
  
  
import java.text.DecimalFormat;  
  
import java.util.HashMap;
```

```

import java.util.Iterator;

import java.util.LinkedHashMap;

public class kapes4

{public static LinkedHashMap<Integer,Integer> display(int[] a)

{

LinkedHashMap<Integer,Integer>hm=new LinkedHashMap<Integer,Integer>();

for(int i=0;i<a.length;i++)

{

int u=1;

for(int j=1;j<=a[i];j++)

{

u=u*j;

}

hm.put(a[i],u);

}

return hm;

}}

```

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

```
import java.util.StringTokenizer;
```

```
public class User {  
    public static void main(String[] args){  
        String s1="Hi";  
        String s2="Hello";  
        System.out.println(capsStart(s1,s2));  
    }  
    public static String capsStart(String s1,String s2){  
        StringBuffer s5=new StringBuffer();  
        int q=s1.length();  
        int w=s2.length();  
        if(q>w)  
        {  
            s5.append(s1).append(s2).append(s1);  
        }  
    }  
}
```

```
    }

    else

    {

        s5.append(s2).append(s1).append(s2);

    }

    return s5.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

```
import java.util.*;
public class Main
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        String s =sc.nextLine();
        System.out.println(User.display(s));
    }
}

import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;
public class User{

    public static String display(String n)
    {

        int year=0;
        String now="01/01/2015";
        SimpleDateFormat sdf1=new SimpleDateFormat("dd/MM/yyyy");
        try
        {

            sdf1.setLenient(false);
            Calendar c1=Calendar.getInstance();
            Date d=sdf1.parse(n);
            c1.setTime(d);
            int y=c1.get(Calendar.YEAR);
            int m=c1.get(Calendar.MONTH);
            int day=c1.get(Calendar.DAY_OF_MONTH);

            Calendar c2=Calendar.getInstance();
            Date d1=sdf1.parse(now);
            c1.setTime(d1);
            int y1=c2.get(Calendar.YEAR);
            int m1=c2.get(Calendar.MONTH);
            int day1=c2.get(Calendar.DAY_OF_MONTH);

            year=y1-y;
            //System.out.println(year);
            if(m>m1)
                year--;
            elseif(m==m1)
```

```
{if(day<day1)
year--;
}
}
catch(Exception e)
{
e.printStackTrace();
}
if(year>18)
return"eligible";
else
return"not-eligible";
} }
```

```
public class UserMainCode{
public static int getMaxSpan(int a[]) {
int i,j,k,count,max=0,p=0;
int n=a.length;
for(i=0;i<n;i++)
{
    count=0;
    for(j=i+1;j<n;j++)
    {
        if(a[i]==a[j])
        {
            p=j;
        }
    }
    for(k=i;k<=p;k++)
    {
        count++;
    }
    if(count>max)
    {
        max=count;
    }
}
```

```
    }  
    return max;  
}  
  
}
```

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 lineItem
 * @return boolean
 */
public boolean isBulkOrder(OrderLineItem lineItem) {
    boolean result=false;

    if(lineItem.quantity>5){
        result = true;
    }
    else if(lineItem.quantity<=5 && lineItem.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;  
        }  
        balanceAmount=total-(float)withdrawAmt;  
        return true;  
    }  
}
```

```
 }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,mobileNumber,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");
}

}

}
```

1)Hello world

```
public class PrintMessage
{
    public static void main(String [] args)
    {
        System.out.println("Welcome to Java world");
    }
}
```

2) Student Details

```
import java.util.Scanner;

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

        System.out.println("Applicant name");

        String str=sc.nextLine();

        System.out.println("Marks obtained in HSC");

        int num=sc.nextInt();

        System.out.println("Total possible marks in HSC");

        int num1=sc.nextInt();

        System.out.println("Engineering cutoff mark");

        float num2=sc.nextFloat();

        System.out.println("Marks obtained in SSLC");

        int num3=sc.nextInt();
```

```
System.out.println("Total possible marks in SSLC");

int num4=sc.nextInt();

System.out.println("Gender");

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

System.out.println("Your Application has been Submitted Successfully");

System.out.println("The name of the applicant: "+str);

System.out.println("Engineering Cutoff: "+num2);

System.out.println("Applicant gender: "+ch);

System.out.println("All the best for your Career");

//Fill code here

}

}

3)Runs Scored
```

```
import java.util.Scanner;

import java.util.*;

public class Main

{

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

//Fill code here

System.out.println("Runs scored");

double a=sc.nextDouble();

System.out.println("Boundaries scored");

double b=sc.nextDouble();
```

```

System.out.println("Sixers scored");

double c=sc.nextDouble();

int run =(int)(a-((b*4)+(c*6)));

double p= ((a-run)/a)*100;

double h=Math.round(p);

System.out.println("Runs scored running between wickets "+ run);

System.out.printf("Percentage of runs scored running between wickets %.2f",h);

}

}

```

4)Population census

```

import java.util.Scanner;

public class Main

{

public static void main(String[] args)

{

Scanner sc=new Scanner(System.in);

//Fill code here

System.out.println("Enter population before 10 years");

int p = sc.nextInt();

System.out.println("Enter population after 10 years");

int q = sc.nextInt();

System.out.println("Population before 10 years "+p+"\nPopulation after 10 years

"+q+"\nIncrease in population between 10 years "+(q-p)+"\nPercentage of population increase in

10 years "+(((q-p)*100/p))+"\nPercentage of population increase in a

year"+((((q-p)*100/p))/10));

```

```
}
```

```
}
```

5) find the word

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

```
public class Main
```

```
{
```

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

```
{
```

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

```
//Fill code here
```

```
System.out.println("Enter the alphabet");
```

```
String s = sc.nextLine();
```

```
switch(s)
```

```
{
```

```
case "A" :
```

```
System.out.println("A for Apple");
```

```
break;
```

```
case "B" :
```

```
System.out.println("B for Ball");
```

```
break;
```

```
case "C" :
```

```
System.out.println("C for Cat");
```

```
break;
```

```
case "D" :
```

```
System.out.println("D for Dog");
```

```
break;

case "E" :
    System.out.println("E for Elephant");

break;

default:
    System.out.println(s + " is an invalid input");
}

sc.close();
}

}
```

6)Ludo King

```
import java.util.Scanner;

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

        //Fill code here

        System.out.println("Enter Alex points");

        int a = sc.nextInt();

        if(a < 0 || a > 50)

            System.out.println(a + " is an invalid number");

        System.out.println("Enter Nikil points");

        int b = sc.nextInt();

        if(b < 0 || b > 50)
```

```

System.out.println(b + " is an invalid number");

System.out.println("Enter Sam points");

int c = sc.nextInt();

if(c < 0 || c > 50)

System.out.println(c + " is an invalid number");

if(a > b && a > c )

System.out.println("Alex scored "+a+" points and won the game");

else if(b > c )

System.out.println("Nikil scored "+b+" points and won the game");

else

System.out.println("Sam scored "+c+" points and won the game");

sc.close();

}

}

}

```

7)Resort Booking

```

import java.util.Scanner;

public class Main{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

//Fill the code here

String s = sc.nextLine();

int i = 0;

while(s.charAt(i) != ':')

{

i++;

}

```

```

}

i++;

if((s.charAt(i)-'0') < 0)

System.out.println("Invalid input for number of adults");

else if((s.charAt(i+2)-'0') < 0)

System.out.println("Invalid input for number of children");

else if((s.charAt(i+4)-'0') <= 0)

System.out.println("Invalid input for number of days");

i = 0;

while(s.charAt(i) != ':')

{

System.out.print(s.charAt(i));

i++;

}

i++;

System.out.print(" your booking is confirmed and the total cost is Rs

"+(((s.charAt(i)-'0')1000 + (s.charAt(i+2)-'0')*650))(s.charAt(i+4)-'0')));

}
}

```

8) Find the Winner

```

import java.util.Scanner;

import java.util.*;

public class Main{

public static void main(String[] args){

Scanner sc=new Scanner(System.in);

```

```
LinkedHashMap<Float,String> hm=new LinkedHashMap<Float,String>();

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

int t = sc.nextInt();

if(t<=1){

System.out.println("Invalid input");

System.exit(0);

}

float min_sum =Float.MAX_VALUE;

int m= t;

float sum=0;

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

for(int k=0;k<t;k++){

sum=0;

String str=sc.next();

String arr1[]= str.split ":";

int n =arr1.length;

float arr[]=new float[4];

arr[0]=Float.parseFloat(arr1[1]);

arr[1]=Float.parseFloat(arr1[2]);

arr[2]=Float.parseFloat(arr1[3]);

arr[3]=Float.parseFloat(arr1[4]);

if(arr[0]<1.0 || arr[1]<1.0 || arr[2]<1.0 || arr[3]<1.0){

System.out.println("Invalid number");

return;

}
```

```

sum = arr[0]+arr[1]+arr[2]+arr[3];

hm.put(sum,arr1[0]);

for(int i=0;i<t;i++){

if(sum<min_sum)

min_sum=sum;

}

}

String minKey= hm.get(min_sum);

if(minKey.equals("Prince")){

System.out.printf("King"+" team wins the race in %.2f minutes",min_sum);

}else{

System.out.printf(minKey+" team wins the race in %.2f minutes",min_sum);

}

//Fill the code here

}

}

```

9)Alphabetical order

```

import java.util.*;

public class Main{

static String sortString(String str) {

char []arr = str.toCharArray();

Arrays.sort(arr);

return String.valueOf(arr);

}

public static void main(String[] args){

```

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

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

String str= sc.nextLine();

String arr1[]= str.split(" ");

int n =arr1.length;

for(int j=0;j<n;j++){

char[] chars = arr1[j].toCharArray();

for(char c : chars){

if(Character.isDigit(c)){

System.out.print(str+" is an invalid input");

return;

}

}

}

String arr[]=new String[n];

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

arr[i]=sortString(arr1[i]);

System.out.print(arr[i]+" ");

}

}

}

10)Sum of Max

import java.util.*;

public class Main{

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

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

System.out.println("Enter the array size");

int n = sc.nextInt();

if(n<=1 || n%2!=0){

System.out.println(n +" is an invalid number");

return;

}

int arr[] = new int[n];

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

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

arr[i]=sc.nextInt();

}

Arrays.sort(arr);

ArrayList<Integer> all= new ArrayList<Integer>();

for(int d=arr.length-1;d>=0;d--){

all.add(arr[d]);

}

ArrayList<Integer> al= new ArrayList<Integer>();

for(int j=0;j<n;j++){

al.add(arr[j] + all.get(j));

}

System.out.println("The maximum number is "+Collections.max(al));

}

}

11)Incredible toys
```

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class CustomerDetails {

    private String customerId;
    private String customerName;
    private long phoneNumber;
    private String emailId;
    private String toyType;
    private double price;

    public CustomerDetails(String customerId, String customerName, long phoneNumber, String
        emailId, String toyType, double price)
    {
        this.customerId=customerId;
        this.customerName=customerName;
        this.phoneNumber=phoneNumber;
        this.emailId=emailId;
        this.toyType=toyType;
        this.price=price;
    }

    public String getCustomerId()
    {
        return this.customerId;
    }

    public void setCustomerId(String customerId)
    {
```

```
this.customerId=customerId;  
}  
  
public String getCustomerName()  
{  
    return this.customerName;  
}  
  
public void setCustomerName(String customerName)  
{  
    this.customerName=customerName;  
}  
  
public long getPhoneNumber()  
{  
    return this.phoneNumber;  
}  
  
public void setPhoneNumber(long phoneNumber)  
{  
    this.phoneNumber=phoneNumber;  
}  
  
public String getEmailId()  
{  
    return this.emailId;  
}  
  
public void setEmailId(String emailId)  
{  
    this.emailId=emailId;
```

```
}

public String getToyType()

{

return toyType;

}

public void setToyType(String toyType)

{

this.toyType=toyType;

}

public double getPrice()

{

return this.price;

}

public void setPrice(double price)

{

this.price=price;

}

// Fill the code

public boolean validateCustomerId() {

String customerId=getCustomerId();

String pattern = "Incredible/[0-9]{3}/[0-9]{4}";

Pattern r = Pattern.compile(pattern);

Matcher m = r.matcher(customerId);

if(m.find())

{
```

```
return true;  
}  
  
return false;  
}  
  
public double calculateDiscount() {  
    double price = getPrice();  
    int dis=0;  
  
    toyType=toyType.toLowerCase();  
    switch(toyType)  
    {  
        case "softtoys":  
            dis=5;  
            break;  
        case "fidgettoys":  
            dis=10;  
            break;  
        case "sensorytoys":  
            dis=15;  
            break;  
        case "puzzles":  
            dis=20;  
            break;  
    }  
  
    return (price-(price*dis/100));  
}
```

```
}
```

```
import java.util.Scanner;

public class Main {

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

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

        String customerId=sc.next();

        System.out.println("Enter Customer Name");

        String customerName=sc.next();

        System.out.println("Enter Phone Number");

        long phoneNumber=sc.nextLong();

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

        String emailId=sc.next();

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

        String toyType=sc.next();

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

        double price = Double.parseDouble(sc.next());

        CustomerDetails cd=new CustomerDetails(customerId,customerName,phoneNumber,emailId,toyType,price);

        if(cd.validateCustomerId())

        {

            double amount=cd.calculateDiscount();

            System.out.printf("Amount to be paid by the Customer %.2f",amount);
        }
    }
}
```

```

}

else{
    System.out.println("Provide a proper Customer Id");
}

}

}

}

12) car showroom

public class Car {

    // Fill the code

    private String carId;
    private String modelNumber;
    private String colour;
    private String transmissionType;
    private String fuelType;
    private String engineType;
    private String bodyType;
    private int engineDisplacement;
    private int bootSpace;
    private int seatCapacity;

    public Car(String carId, String modelNumber, String colour, String transmissionType, String
fuelType,
        String engineType, String bodyType, int engineDisplacement, int bootSpace, int seatCapacity)
    {
        this.carId=carId;
        this.modelNumber=modelNumber;
    }
}

```

```
    this.colour=colour;  
  
    this.transmissionType=transmissionType;  
  
    this.fuelType=fuelType;  
  
    this.engineType=engineType;  
  
    this.bodyType=bodyType;  
  
    this.engineDisplacement=engineDisplacement;  
  
    this.bootSpace=bootSpace;  
  
    this.seatCapacity=seatCapacity;  
  
}  
  
public String getCarId()  
{  
  
    return this.carId;  
  
}  
  
public void setCarId(String carId)  
{  
  
    this.carId=carId;  
  
}  
  
public String getModelNumber()  
{  
  
    return this.modelNumber;  
  
}  
  
public void setModelNumber(String modelNumber)  
{  
  
    this.modelNumber=modelNumber;  
  
}
```

```
public String getColour()
{
    return this.colour;
}

public void setColour(String colour)
{
    this.colour=colour;
}

public String getTransmissionType()
{
    return this.transmissionType;
}

public void setTransmissionType(String transmissionType)
{
    this.transmissionType=transmissionType;
}

public String getFuelType()
{
    return this.fuelType;
}

public void setFuelType(String fuelType)
{
    this.fuelType=fuelType;
}

public String getEngineType()
```

```
{  
    return this.engineType;  
}  
  
public void setEngineType(String engineType)  
{  
    this.engineType=engineType;  
}  
  
public String getBodyType()  
{  
    return this.bodyType;  
}  
  
public void setBodyType(String bodyType)  
{  
    this.bodyType=bodyType;  
}  
  
public int getEngineDisplacement()  
{  
    return this.engineDisplacement;  
}  
  
public void setEngineDisplacement(int engineDisplacement)  
{  
    this.engineDisplacement=engineDisplacement;  
}  
  
public int getBootSpace()  
{
```

```
        return this.bootSpace;  
    }  
  
    public void setBootSpace(int bootSpace)  
    {  
        this.bootSpace=bootSpace;  
    }  
  
    public int getSeatCapacity()  
    {  
        return this.seatCapacity;  
    }  
  
    public void setSeatCapacity(int seatCapacity)  
    {  
        this.seatCapacity=seatCapacity;  
    }  
  
}
```

Carutility.java

```
import java.util.Scanner;  
  
public class CarUtility {  
  
    public static void main(String[] args) {  
        // Fill the code  
        Car[] carObj=createCarDetails();
```

```

int n=carObj.length;

System.out.println("No of Car Details "+n);

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

{

    System.out.println("Car Details "+(i+1));

    System.out.println("Car Id "+carObj[i].getCarId());

    System.out.println("Model Number "+carObj[i].getModelNumber());

    System.out.println("Colour "+carObj[i].getColour());

    System.out.println("Transmission Type "+carObj[i].getTransmissionType());

    System.out.println("Fuel Type "+carObj[i].getFuelType());

    System.out.println("Engine Type "+carObj[i].getEngineType());

    System.out.println("Body Type "+carObj[i].getBodyType());

    System.out.println("Engine Displacement "+carObj[i].getEngineDisplacement());

    System.out.println("Boot Space "+carObj[i].getBootSpace());

    System.out.println("Seat Capacity "+carObj[i].getSeatCapacity());


}

}

```

```

public static Car[] createCarDetails(){

    Scanner sc = new Scanner(System.in);

    // Fill the code

    System.out.println("Enter the number of car details to created");

    int n=sc.nextInt();

```

```
Car[] carObj=new Car[n];

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

{

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

    String carId=sc.nextLine();

    carId = sc.nextLine();

    System.out.println("Enter Model Number");

    String modelNumber=sc.nextLine();

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

    String colour=sc.nextLine();

    System.out.println("Enter Transmission Type");

    String transmissionType=sc.nextLine();

    System.out.println("Enter Fuel Type");

    String fuelType=sc.nextLine();

    System.out.println("Enter Engine Type");

    String engineType=sc.nextLine();

    System.out.println("Enter Body Type");

    String bodyType=sc.nextLine();

    System.out.println("Enter Engine Displacement");

    int engineDisplacement=sc.nextInt();

    System.out.println("Enter Boot Space");

    int bootSpace=sc.nextInt();

    System.out.println("Enter Seat Capacity");

    int seatCapacity=sc.nextInt();

    carObj[i]=new Car(carId,modelNumber, colour, transmissionType, fuelType , engineType, bodyType, engineDisplacement, bootSpace, seatCapacity);
```

```
        }

        return carObj;
    }

}

13)Electricity board

public class CustomerDetails {

    private String customerId;

    private String customerName;

    private long phoneNumber;

    private String city;

    private double unitsConsumed;

    private double costPerUnit;

    public CustomerDetails(String customerId, String customerName, long
        phoneNumber, String city, double unitsConsumed, double costPerUnit) {
        this.customerId = customerId;
        this.customerName = customerName;
        this.phoneNumber = phoneNumber;
        this.city = city;
        this.unitsConsumed = unitsConsumed;
        this.costPerUnit = costPerUnit;
    }

    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 getPhoneNumber(){  
    return phoneNumber;  
}  
  
public void setPhoneNumber(long phoneNumber){  
    this.phoneNumber=phoneNumber;  
}  
  
public String getCity(){  
    return city;  
}  
  
public void setCity( String city){  
    this.city=city;  
}  
  
public double getUnitsConsumed(){  
    return unitsConsumed;  
}  
  
public void setUnitsConsumed(double unitsConsumed){  
    this.unitsConsumed=unitsConsumed;
```

```
}

public double getCostPerUnit(){

    return costPerUnit;

}

public void setCustomerId(double costPerUnit){

    this.costPerUnit=costPerUnit;

}

public double calculateElectricityBill(){

    double amount = unitsConsumed * costPerUnit;

    return amount;

}
```

```
import java.util.*;

public class Main {

    public static void main(String[] args) {

        Scanner scanner=new Scanner(System.in);

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

        String str= scanner.nextLine();

        System.out.println("Enter Customer Name");

        String s2 = scanner.nextLine();

        System.out.println("Enter Phone Number");

        long phone = scanner.nextLong();

        System.out.println("Enter City");
```

```
String city = scanner.next();

System.out.println("Enter Units Consumed");

double cu=scanner.nextDouble();

System.out.println("Enter Cost per Unit");

double cuu=scanner.nextDouble();

CustomerDetails c1= new CustomerDetails(str,s2,phone,city,cu,cuu);

System.out.println("Amount to be paid is Rs."+

String.format("%.2f",c1.calculateElectricityBill()));

}

}
```

14)Movie Static

```
import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

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

String str= scanner.nextLine();

System.out.println("Enter no of bookings");

int bookings = scanner.nextInt();

System.out.println("Enter the available tickets");

int availableTickets = scanner.nextInt();

Ticket ticket = new Ticket();

ticket.setAvailableTickets(availableTickets);

for (int i = 0; i < bookings; ++i) {

System.out.println("Enter the ticketid");
```

```

int ticketId = scanner.nextInt();

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

int price = scanner.nextInt();

System.out.println("Enter the no of tickets");

int tickets = scanner.nextInt();

ticket.setTicketid(ticketId);

ticket.setPrice(price);

System.out.println("Available tickets: " + ticket.getAvailableTickets());

int cost = ticket.calculateTicketCost(tickets);

if(cost>0){

System.out.println("Total amount:" + cost);}

else{

System.out.println("Tickets are not available");

return;

}if(ticket.getAvailableTickets()==0){

System.out.println("House full");

}else{

System.out.println("Available ticket after booking:" + ticket.getAvailableTickets());

}

}

}

```

```

public class Ticket {

private int ticketid;

```

```
private int price;

private static int availableTickets;

public int getTicketid() {

    return ticketid;

}

public void setTicketid(int ticketid) {

    this.ticketid = ticketid;

}

public int getPrice() {

    return price;

}

public void setPrice(int price) {

    this.price = price;

}

public static int getAvailableTickets() {

    return availableTickets;

}

public static void setAvailableTickets(int availableTickets) {

    if (availableTickets > 0) {

        Ticket.availableTickets = availableTickets;

    }else{

        Ticket.availableTickets=-1;

    }

}

public int calculateTicketCost(int nooftickets) {
```

```
if (nooftickets <= availableTickets) {  
    availableTickets -= nooftickets;  
    return nooftickets * price;  
} else {  
    return -1;  
}  
}  
}  
}
```

15)game card

```
public class CardPoints {  
  
    private int cardId;  
    private String holderName;  
    private int balancePoints;  
  
    public CardPoints(int cardId, String holderName, int balancePoints) {  
        this.cardId = cardId;  
        this.holderName = holderName;  
        this.balancePoints = balancePoints;  
    }  
  
    public boolean withdrawPoints(int points) {  
        if(balancePoints >= points) {
```

```
balancePoints -= points;

System.out.println("Balance points after used:" + balancePoints);

return true;

} else {

    System.out.println("Sorry!!! No enough points");

    return false;

}

}

public int getCardId() {

    return cardId;

}

public void setCardId(int cardId) {

    this.cardId = cardId;

}

public String getHolderName() {

    return holderName;

}

public void setHolderName(String holderName) {

    this.holderName = holderName;

}
```

```
public int getBalancePoints() {  
    return balancePoints;  
}  
  
public void setBalancePoints(int balancePoints) {  
    this.balancePoints = balancePoints;  
}  


---



```
import java.util.Scanner;

public class GameCardDetails {

 public CardPoints getCardDetails()
 {
 Scanner sc = new Scanner(System.in);

 System.out.println("Enter card id");
 int cardId = sc.nextInt();

 System.out.println("Enter card holder name");
 String holderName = sc.next();

 System.out.println("Enter balance points");
 int balancePoints = sc.nextInt();

 do{
 if (balancePoints <= 0){
```


```

```
System.out.println("Balance points should be positive");

System.out.println("Enter balance points");

balancePoints = sc.nextInt();

continue;

}

else

{

    break;

}

}

while(true);

return new CardPoints(cardId, holderName, balancePoints);

}

public int getPointUsage()

{

Scanner sc = new Scanner(System.in);

int pointUsage = sc.nextInt();

if (pointUsage <= 0){

    System.out.println("Points should be positive");

    return getPointUsage();

}

return pointUsage;

}
```

```
public static void main(String[] arg)
{
    GameCardDetails gameCardDetails = new GameCardDetails();
    CardPoints cp = gameCardDetails.getCardDetails();
    System.out.println("Enter points should be used");
    int points = gameCardDetails.getPointUsage();
    cp.withdrawPoints(points);

}
}
```

16) doctor details

```
public class Doctor {
    private String doctorId;
    private String doctorName;
    private String specialization;
    private Hospital hospital;
    public String getDoctorId() {
        return doctorId;
    }
    public void setDoctorId(String doctorId) {
        this.doctorId = doctorId;
    }
    public String getDoctorName() {
```

```
        return doctorName;  
    }  
  
    public void setDoctorName(String doctorName) {  
        this.doctorName = doctorName;  
    }  
  
    public String getSpecialization() {  
        return specialization;  
    }  
  
    public void setSpecialization(String specialization) {  
        this.specialization = specialization;  
    }  
  
    public Hospital getHospital() {  
        return hospital;  
    }  
  
    public void setHospital(Hospital hospital) {  
        this.hospital = hospital;  
    }  
  
    public Doctor(String doctorId, String doctorName, String specialization, Hospital hospital) {  
        this.doctorId = doctorId;  
        this.doctorName = doctorName;  
        this.specialization = specialization;  
        this.hospital = hospital;  
    }  
}  
  
-----  
public class Hospital {
```

```
private String hospitalName;  
private long contactNumber;  
private String city;  
  
public String getHospitalName() {  
    return hospitalName;  
}  
  
public long getContactNumber() {  
    return contactNumber;  
}  
  
public String getCity() {  
    return city;  
}  
  
public void setHospitalName(String hospitalName) {  
    this.hospitalName = hospitalName;  
}  
  
public void setContactNumber(long contactNumber) {  
    this.contactNumber = contactNumber;  
}  
  
public void setCity(String city) {  
    this.city = city;  
}  
  
public Hospital(String hospitalName, long contactNumber, String city) {  
    this.hospitalName = hospitalName;  
    this.contactNumber = contactNumber;  
    this.city = city;
```

```
 }  
 }  
  
-----
```

```
import java.util.Scanner;  
  
public class Main {  
  
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
  
        Doctor doc=createDoctorDetails();  
  
        System.out.println("Doctor id:"+doc.getDoctorId());  
  
        System.out.println("Doctor name:"+doc.getDoctorName());  
  
        System.out.println("Specialization:"+doc.getSpecialization());  
  
        System.out.println("Hospital name:"+doc.getHospital().getHospitalName());  
  
        System.out.println("Contact Number:"+doc.getHospital().getContactNumber());  
  
        System.out.println("City:"+doc.getHospital().getCity());  
    }  
  
    public static Doctor createDoctorDetails() {  
        Scanner sc = new Scanner(System.in);  
  
        System.out.println("Enter Hospital Name");  
  
        String hName=sc.nextLine();  
  
        System.out.println("Enter Contact Number");  
  
        long cNo=sc.nextLong();  
  
        sc.nextLine();  
  
        System.out.println("Enter City");  
    }  
}
```

```
String city=sc.nextLine();

Hospital h=new Hospital(hName,cNo,city);

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

String id=sc.nextLine();

System.out.println("Enter Doctor Name");

String dName=sc.nextLine();

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

String specialization = sc.nextLine();

Doctor doc=new Doctor(id,dName,specialization,h);

return doc;

}

}

17)Disney
```

```
public class BoatHouseBooking extends Booking{
```

```
// Fill the code
```

```
private int noOfDays;
```

```
private String foodType;
```

```
public int getNoOfDays() {
```

```
    return noOfDays;
```

```
}
```

```
public void setNoOfDays(int noOfDays) {
```

```
this.noOfDays = noOfDays;  
}  
  
  
public String getFoodType() {  
    return foodType;  
}  
  
  
public void setFoodType(String foodType) {  
    this.foodType = foodType;  
}  
  
  
public BoatHouseBooking(String customerName, String cityName, String phoneNumber, int noOfPeople,  
int noOfDays,String foodType) {  
    super(customerName, cityName, phoneNumber, noOfPeople);  
    this.noOfDays=noOfDays;  
    this.foodType=foodType;  
}  
  
  
public double calculateTotalAmount() {  
  
  
    // Fill the code  
  
  
    if(this.foodType.equals("NonVeg")){  
        return (double)(this.noOfPeople*800) + (this.noOfDays*3000)+500;  
    }else if(this.foodType.equals("Veg") || this.foodType.equals("VEG")){  
        return (double)((this.noOfPeople*800) + (this.noOfDays*3000)+250);  
    }  
}
```

```
}
```

```
return 0.0;
```

```
}
```

```
}
```

```
public class BoatRideBooking extends Booking{
```

```
// Fill the code
```

```
private float noOfHours;
```

```
private String guide;
```

```
public BoatRideBooking(String customerName, String cityName, String phoneNumber, int noOfPeople,  
float noOfHours, String guide) {
```

```
super(customerName, cityName, phoneNumber, noOfPeople);
```

```
this.noOfHours=noOfHours;
```

```
this.guide=guide;
```

```
}
```

```
public double calculateTotalAmount() {
```

```
// Fill the code

if(this.guide.equals("Yes") || this.guide.equals("yes")){
    return (double)(this.noOfPeople*80) + (this.noOfHours*300) + 150;
} else if(this.guide.equals("No") || this.guide.equals("no")){
    return (double)(this.noOfPeople*80) + (this.noOfHours*300);
}
return 0.0;
}

}
```

```
public abstract class Booking {

// Fill the code

protected String customerName;
protected String cityName;
protected String phoneNumber;
protected int noOfPeople;

public Booking(String customerName, String cityName, String phoneNumber, int noOfPeople) {
    this.customerName = customerName;
    this.cityName = cityName;
    this.phoneNumber = phoneNumber;
}
```

```
this.noOfPeople = noOfPeople;  
}  
  
  
public String getCustomerName() {  
    return customerName;  
}  
  
  
public void setCustomerName(String customerName) {  
    this.customerName = customerName;  
}  
  
  
public String getCityName() {  
    return cityName;  
}  
  
  
public void setCityName(String cityName) {  
    this.cityName = cityName;  
}  
  
  
public String getPhoneNumber() {  
    return phoneNumber;  
}  
  
  
public void setPhoneNumber(String phoneNumber) {  
    this.phoneNumber = phoneNumber;
```

```
}
```

```
public int getNoOfPeople() {
```

```
    return noOfPeople;
```

```
}
```

```
public void setNoOfPeople(int noOfPeople) {
```

```
    this.noOfPeople = noOfPeople;
```

```
}
```

```
public abstract double calculateTotalAmount();
```

```
}
```

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

```
public class UserInterface {
```

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

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

```
        // Fill the code
```

```
        System.out.println("Enter the Customer Name");
```

```
        String customerName = s.next();
```

```
        System.out.println("Enter the City name");
```

```
String cityName = s.next();  
System.out.println("Enter the phone number");  
String phoneNumber = s.next();  
System.out.println("Enter number of people");  
int noOfPeople = s.nextInt();  
System.out.println("Enter the option\n" +  
"1. Boat House Booking\n" +  
"2. Boat Ride Booking"  
);  
int option = s.nextInt();  
  
switch (option){  
case 1:  
System.out.println("Enter number of days");  
int noOfDays = s.nextInt();  
System.out.println("Enter food type (Veg/NonVeg)");  
String foodType = s.next();  
  
BoatHouseBooking BHB = new BoatHouseBooking(  
customerName,  
cityName,  
phoneNumber,  
noOfPeople,  
noOfDays,  
foodType);
```

```
System.out.println("Your booking has been confirmed pay Rs."+BHB.calculateTotalAmount());  
break;
```

case 2:

```
System.out.println("Enter number of hours");  
float noOfHours = s.nextFloat();  
System.out.println("Do you want guide (Yes/No)");  
String guide = s.next();
```

```
BoatRideBooking BRB = new BoatRideBooking(  
customerName,  
cityName,  
phoneNumber,  
noOfPeople,  
noOfHours,  
guide  
);
```

```
System.out.println("Your booking has been confirmed pay Rs."+BRB.calculateTotalAmount());
```

```
break;
```

```
}
```

```
}
```

```
}
```

18)College Fees

```
public class DayScholar extends Student{
```

```
// Fill the code
```

```
private int busNumber;
```

```
private float distance;
```

```
public DayScholar(int studentId, String studentName, String department, String gender, String category,  
double collegeFee,int busNumber,float distance) {
```

```
super(studentId, studentName, department, gender, category, collegeFee);
```

```
this.busNumber=busNumber;
```

```
this.distance=distance;
```

```
}
```

```
public int getBusNumber() {
```

```
return busNumber;
```

```
}
```

```
public void setBusNumber(int busNumber) {
```

```
this.busNumber = busNumber;  
}
```

```
public float getDistance() {  
    return distance;  
}
```

```
public void setDistance(float distance) {  
    this.distance = distance;  
}
```

```
public double calculateTotalFee() {
```

// Fill the code

```
int busFees=0;
```

```
if(this.distance > 30 && this.distance <= 40){
```

busFees=28000;

```
}else if(this.distance > 20 && this.distance <= 30){
```

busFees=20000;

```
}else if(this.distance > 10 && this.distance <= 20){
```

```
busFees=12000;  
}else if(this.distance <= 10){  
busFees=6000;  
}  
if(this.collegeFee>0){  
return (double)(this.collegeFee+busFees);  
}  
  
return 0;  
}  
  
}
```

```
public abstract class Student {
```

```
// Fill the code  
  
protected int studentId;  
protected String studentName;  
protected String department;  
protected String gender;
```



```
public String getStudentName() {  
    return studentName;  
}
```

```
public void setStudentName(String studentName) {  
    this.studentName = studentName;  
}
```

```
public String getDepartment() {  
    return department;  
}
```

```
public void setDepartment(String department) {  
    this.department = department;  
}
```

```
public String getGender() {
```

```
    return gender;
```

```
}
```

```
public void setGender(String gender) {
```

```
    this.gender = gender;
```

```
}
```

```
public String getCategory() {
```

```
    return category;
```

```
}
```

```
public void setCategory(String category) {
```

```
    this.category = category;
```

```
}
```

```
public double getCollegeFee() {  
    return collegeFee;  
}  
  
public void setCollegeFee(double collegeFee) {  
    this.collegeFee = collegeFee;  
}  
  
public abstract double calculateTotalFee();  
}  
  
-----  
import java.util.Scanner;
```

```
public class UserInterface {
```

```
public static void main(String[] args) {  
  
    Scanner s = new Scanner(System.in);  
    // Fill the code  
  
    System.out.println("Enter Student Id");  
    int studentId = s.nextInt();  
  
    System.out.println("Enter Student name");  
    String studentName = s.next();  
  
    System.out.println("Enter Department name");  
    String department = s.next();  
  
    System.out.println("Enter gender");  
    String gender = s.next();  
  
    System.out.println("Enter category");  
    String category = s.next();  
  
    System.out.println("Enter College fee");  
    double collegeFee = s.nextDouble();  
  
    switch(category){  
        case "DayScholar":  
            System.out.println("Enter Bus number");  
            int busNumber = s.nextInt();  
            System.out.println("Enter the distance");  
    }  
}
```

```
float distance = s.nextFloat();

DayScholar dayScholar = new DayScholar(
    studentId,
    studentName,
    department,
    gender,
    category,
    collegeFee,
    busNumber,
    distance
);

System.out.println("Total College fee is "+dayScholar.calculateTotalFee());
break;

case "Hosteller":
    System.out.println("Enter the room number");
    int roomNumber = s.nextInt();
```

```
System.out.println("Enter the Block name");
char blockName = s.next().charAt(0);
System.out.println("Enter the room type");
String roomType = s.next();

Hosteller hosteller = new Hosteller(
studentId,
studentName,
department,
gender,
category,
collegeFee,
roomNumber,
blockName,
roomType
);

System.out.println("Total College fee is "+hosteller.calculateTotalFee());
break;
}
}
```

```
}
```

```
public class Hosteller extends Student{
```

```
// Fill the code
```

```
private int roomNumber;  
private char blockName;  
private String roomType;
```

```
public Hosteller(int studentId, String studentName, String department, String gender, String category,  
double collegeFee, int roomNumber, char blockName, String roomType) {  
super(studentId, studentName, department, gender, category, collegeFee);  
this.roomNumber=roomNumber;  
this.blockName=blockName;  
this.roomType=roomType;  
}
```

```
@Override  
  
public double calculateTotalFee() {  
  
    int hostelFee=0,roomTypeAmount=0;  
  
    switch (this.blockName){  
  
        case 'A':  
  
            hostelFee=60000;  
  
            if(this.roomType.equals("AC")){  
  
                roomTypeAmount=8000;  
  
            }  
  
            break;  
  
        case 'B':  
  
            hostelFee=50000;  
  
            if(this.roomType.equals("AC")){  
  
                roomTypeAmount=5000;  
  
            }  
  
            break;  
  
        case 'C':  
  
            hostelFee=40000;  
  
            if(this.roomType.equals("AC")){  
  
                roomTypeAmount=2500;  
  
            }  
  
            break;  
  
    }  
}
```

```
if(this.roomType.equals("AC")){
    return (double)(this.collegeFee+hostelFee+roomTypeAmount);
} else if(!this.roomType.equals("AC")){
    return (double)(this.collegeFee+hostelFee);
}
return 0.0;
}
```

19)Departmental store

```
public interface BonusPoints {
    // Fill the code
    public abstract double calculateBonusPoints();
}
```

```
public class CustomerDetails implements BonusPoints, DoorDelivery {
```

```
// Fill the code

private String customerName;

private String phoneNumber;

private String streetName;

private double billAmount;

private int distance;

public String getCustomerName() {

    return customerName;

}

public void setCustomerName(String customerName) {

    this.customerName = customerName;

}

public String getPhoneNumber() {

    return phoneNumber;

}

public void setPhoneNumber(String phoneNumber) {

    this.phoneNumber = phoneNumber;

}

public String getStreetName() {

    return streetName;

}

public void setStreetName(String streetName) {

    this.streetName = streetName;

}

public double getBillAmount() {
```

```
        return billAmount;  
    }  
  
    public void setBillAmount(double billAmount) {  
        this.billAmount = billAmount;  
    }  
  
    public int getDistance() {  
        return distance;  
    }  
  
    public void setDistance(int distance) {  
        this.distance = distance;  
    }  
  
    public CustomerDetails(String customerName, String phoneNumber, String streetName, double  
billAmount,  
                           int distance) {  
        this.customerName = customerName;  
        this.phoneNumber = phoneNumber;  
        this.streetName = streetName;  
        this.billAmount = billAmount;  
        this.distance = distance;  
    }  
  
  
    public double calculateBonusPoints() {  
        // Fill the code  
        double billAmount=this.getBillAmount();  
        if(billAmount<250)  
    {
```



```
}
```

```
public interface DoorDelivery {  
  
    // Fill the code  
  
    public abstract double deliveryCharge();  
  
}
```

```
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 the customer name");  
  
        String name=sc.nextLine();  
  
        System.out.println("Enter the phone number");  
  
        String phone=sc.nextLine();  
  
        System.out.println("Enter the street name");  
  
        String street=sc.nextLine();  
  
        System.out.println("Enter the bill amount");  
  
        double amount=sc.nextDouble();  
  
        sc.nextLine();  
  
        System.out.println("Enter the distance");  
  
        int distance=sc.nextInt();  
  
        CustomerDetails obj=new CustomerDetails(name,phone,street,amount,distance);
```

```
        double bonus=obj.calculateBonusPoints();

        double delivery=obj.deliveryCharge();

        System.out.println("Customer name "+obj.getCustomerName());

        System.out.println("Phone number "+obj.getPhoneNumber());

        System.out.println("Street name "+obj.getStreetName());

        System.out.println("Bonus points "+bonus);

        System.out.println("Delivery charge "+delivery);

    }

}

20)Viveks
```

```
public abstract class Bero {

    protected String beroType;

    protected String beroColour;

    protected double price;

    public String getBeroType() {

        return beroType;

    }

    public void setBeroType(String beroType) {

        this.beroType = beroType;

    }

    public String getBeroColour() {

        return beroColour;

    }

    public void setBeroColour(String beroColour) {

        this.beroColour = beroColour;

    }

}
```

```
}

public double getPrice() {

    return price;

}

public void setPrice(double price) {

    this.price = price;

}

public Bero(String beroType, String beroColour) {

    this.beroType = beroType;

    this.beroColour = beroColour;

}

public abstract void calculatePrice();

}
```

```
public class CustomerDetails {

    private String customerName;

    private long phoneNumber;

    private String address;

    public String getCustomerName() {

        return customerName;

    }

    public void setCustomerName(String customerName) {

        this.customerName = customerName;

    }

    public long getPhoneNumber() {
```

```
        return phoneNumber;  
    }  
  
    public void setPhoneNumber(long phoneNumber) {  
        this.phoneNumber = phoneNumber;  
    }  
  
    public String getAddress() {  
        return address;  
    }  
  
    public void setAddress(String address) {  
        this.address = address;  
    }  
  
    public CustomerDetails(String customerName, long phoneNumber, String address) {  
        super();  
        this.customerName = customerName;  
        this.phoneNumber = phoneNumber;  
        this.address = address;  
    }  
}
```

```
public class Discount {  
  
    public double calculateDiscount(Bero bObj)  
    {  
        double price=bObj.getPrice();  
        double discount=0;  
        if(bObj instanceof SteelBero)
```

```
{  
    discount=price*0.1;  
}  
  
else if(bObj instanceof WoodenBero)  
{  
    discount=price*0.15;  
}  
  
return discount;  
}  
}
```

```
public class SteelBero extends Bero {  
  
    private int beroHeight;  
  
    public int getBeroHeight() {  
        return beroHeight;  
    }  
  
    public void setBeroHeight(int beroHeight) {  
        this.beroHeight = beroHeight;  
    }  
  
    public SteelBero(String type, String colour, int height) {  
        // TODO Auto-generated constructor stub  
    }  
}
```

```
super(type, colour);

this.beroHeight = height;

}

@Override

public void calculatePrice() {

    int height=this.getBeroHeight();

    if(height==3)

    {

        this.setPrice(5000);

    }

    else if(height==5)

    {

        this.setPrice(8000);

    }

    else if(height==7)

    {

        this.setPrice(10000);

    }

}

}

-----
```

```
public class WoodenBero extends Bero {

    private String woodType;
```

```
public String getWoodType() {  
    return woodType;  
}  
  
public void setWoodType(String woodType) {  
    this.woodType = woodType;  
}  
  
public WoodenBero(String beroType, String beroColour, String woodType) {  
    super(beroType, beroColour);  
    this.woodType = woodType;  
}  
  
@Override  
public void calculatePrice() {  
    // TODO Auto-generated method stub  
    String type=this.getWoodType();  
    if(type.equals("Ply Wood"))  
    {  
        this.setPrice(15000);  
    }  
    else if(type.equals("Teak Wood"))  
    {  
        this.setPrice(12000);  
    }  
}
```

```
        else if(type.equals("Engineered Wood"))

    {

        this.setPrice(10000);

    }

}

-----
```

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        System.out.println("Enter Customer Name");

        String name=sc.nextLine();

        System.out.println("Enter Phone Number");

        long phone=sc.nextLong();

        sc.nextLine();

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

        String address=sc.nextLine();

        System.out.println("Enter Bero Type");

        String type=sc.nextLine();

        System.out.println("Enter Bero Colour");

        String colour=sc.nextLine();

        double price=0;
```

```
Discount d=new Discount();

Bero obj;

if(type.equals("Steel Bero"))

{

    System.out.println("Enter Bero Height");

    int height=sc.nextInt();

    sc.nextLine();

    obj=new SteelBero(type,colour,height);

    obj.calculatePrice();

    price=obj.getPrice()- d.calculateDiscount(obj);

}

else if(type.equals("Wooden Bero"))

{

    System.out.println("Enter Wood Type");

    String wtype=sc.nextLine();

    obj=new WoodenBero(type,colour,wtype);

    obj.calculatePrice();

    price=obj.getPrice()- d.calculateDiscount(obj);

}

System.out.print("Amount needs to be paid Rs.");

System.out.printf("%.2f",price);

}

}
```

21)Campus radio

```
//make the necessary change to make this class a Exception  
public class StationNotAvailableException extends Exception{  
    public StationNotAvailableException(String message){  
        super(message);  
    }  
}
```

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

```
public class UserInterface {  
  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
        System.out.println("Scan the radio station");  
        float freq = sc.nextFloat();  
  
        try{  
            Validator vd=new Validator();  
            vd.validateStation(freq);  
            System.out.println("Radio Station on!");  
        }catch(StationNotAvailableException e){  
            System.out.println("Radio Station not available");  
        }  
        // fill in the code  
    }  
}
```

```
    }

}

-----
```

```
public class Validator {

    public static boolean validateStation(float freq) throws StationNotAvailableException {
        boolean flag = false;

        if (freq == 91.2f || freq == 93.5f || freq == 98.9f || freq == 109.4f)
            flag = true;
        else{
            throw new StationNotAvailableException("Radio Station not available");
            // fill the logic
        }
        return flag;
    }
}

22)String extract

import java.util.Scanner;
```

```
public class Main {  
  
    public static void main(String args[])  
    {  
        Main stringExtraction = new Main();  
  
        Scanner sc=new Scanner(System.in);  
  
        // Fill the code  
  
        System.out.println("Enter the String");  
  
        String sentence = sc.nextLine();  
  
        System.out.println("Enter First Index");  
  
        int s = sc.nextInt();  
  
        System.out.println("Enter Second Index");  
  
        int e = sc.nextInt();  
  
        System.out.println(stringExtraction.extractString(sentence,s,e));  
  
    }  
  
    public String extractString(String sentence,int number1,int number2)  
    {  
        // Fill the code  
    }  
}
```

```
StringBuilder stringBuilder = new StringBuilder();

try{
    if(number1<0 || number1 > sentence.length()){
        throw new StringIndexOutOfBoundsException();
    }else if(number2<0 || number2 > sentence.length()){
        throw new StringIndexOutOfBoundsException();
    }else if(number1>number2){
        throw new StringIndexOutOfBoundsException();
    }
    else{
        for (int i = number1; i < number2; i++) {
            stringBuilder.append(sentence.charAt(i));
        }
        stringBuilder.append(".");
    }
} catch (StringIndexOutOfBoundsException s){
    stringBuilder.append("Extraction of String using the given index is not possible.Thanks for using the application. ");
}finally {
    stringBuilder.append("Thanks for using the application.");
}

return stringBuilder.toString();
}
```

}

23)Vega Cricket

```
import java.util.*;
import java.util.stream.*;

public class Main {
    public static void main(String args[]){
        Scanner sc=new Scanner(System.in);
        ManagementUtility M=new ManagementUtility();
        Map<String,Integer> playerMap=new HashMap<String,Integer>();
        M.setPlayerMap(playerMap);
        int z=0;
        while(z!=3)
        {
            System.out.println("Select an option:");
            System.out.println("1.Add player score");
            System.out.println("2.Display");
            System.out.println("3.Exit");
            z=sc.nextInt();
            String name="";
            int score=0;
            if(z==1)
            {
```

```
System.out.println("Enter the player name");

name=sc.next();

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

score=sc.nextInt();

M.addPlayerScore(name,score);

}

else if(z==2)

{

Map<String,Integer> playermap=M.getPlayerMap();

Stream<Map.Entry<String,Integer>> playerStream=playermap.entrySet().stream();

int max_score=M.maximumScore(playerStream);

if(max_score==0)

{

System.out.println("No players found");

}

else

{

System.out.println("The maximum score of an individual player for these match is

"+max_score);

}

}

else{

System.out.println("Thank you for using the application.");

break;

}
```

```
    }  
}  
  
-----
```

```
import java.util.Map;  
import java.util.*;  
import java.util.stream.Stream;  
import java.util.stream.Collectors;  
  
  
public class ManagementUtility {  
    private Map<String, Integer> playerMap;  
  
  
    public Map<String, Integer> getPlayerMap() {  
        return playerMap;  
    }  
  
  
    public void setPlayerMap(Map<String, Integer> playerMap) {  
        this.playerMap = playerMap;  
    }  
  
  
    public void addPlayerScore(String playerName, int score) {  
        Map<String, Integer> playerMap = getPlayerMap();  
        playerMap.put(playerName, score);  
  
  
        //Fill the code here  
    }  
}
```

```
}
```

```
public static int maximumScore(Stream<Map.Entry<String,Integer>> playerStream){  
  
    //List list = playerStream.collect(Collectors.toList());  
  
    //Collections.sort(hashList, Collections.reverseOrder());  
  
    List<Integer> l=playerStream.map(Map.Entry::getValue).collect(Collectors.toList());  
  
    if(l.isEmpty())  
  
    {  
  
        return 0;  
  
    }  
  
    else{  
  
        int m=Collections.max(l);  
  
        System.out.println(m);  
  
        // Fill the code here  
  
        return m;  
    }  
}
```

```
}
```

24)Top tier

```
import java.util.*;
```

```
public class Main {
```

```

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

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

    int n = Integer.parseInt(sc.nextLine());

    VehicleUtility vh = new VehicleUtility();

    vh.setVehicleMap(new HashMap<String,Double>());

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

        System.out.println("Enter the vehicle name and price of Vehicle "+(i+1));

        String veh=sc.nextLine();

        double price=Double.parseDouble(sc.nextLine());

        vh.addVehiclePriceDetails(veh,price);

    }

    Map<String,Double> polmap=vh.getVehicleMap();

    char choice;

    do

    {

        System.out.println("Enter the vehicle name to be searched");

        String search = sc.nextLine();

        if(vh.calculateCostAfterDiscount(search)!=0){

            System.out.println("Price after discount for "+ search +" is

"+vh.calculateCostAfterDiscount(search));

        }else{

            System.out.println( "TVS Apache RTR 180 is not available currently");

        }

        System.out.println("Do you want to continue (Y/N)");
    }
}

```

```
choice=sc.nextLine().charAt(0);

if((choice=='n')||(choice=='N')){

    System.out.println("Thank you for using the Application");

    break;

}

}

while((choice=='y')||(choice=='Y'));

}

-----
```

```
import java.util.HashMap;

import java.util.Map;

public class VehicleUtility {

    private Map<String, Double> vehicleMap = new HashMap<String, Double>();

    public Map<String, Double> getVehicleMap() {

        return vehicleMap;

    }

    public void setVehicleMap(Map<String, Double> vehicleMap) {

        this.vehicleMap = vehicleMap;

    }

    // This method should add the vehicleName as key and the price of the
```

```

// vehicle as value into a Map

public void addVehiclePriceDetails(String vehicleName, double price) {

    // fill the code

    vehicleMap.put(vehicleName,price);

}

// This method should calculate the discount and return the selling price

// after the discount for the vehicle name passed as an argument.

public double calculateCostAfterDiscount(String vehicleName) {

    if(vehicleMap.containsKey(vehicleName)){

        double discount=0;

        // fill the code

        double sp=0;

        double h=vehicleMap.get(vehicleName);

        if(vehicleName.contains("TVS")){
            discount=10;
        }else if(vehicleName.contains("Honda")){
            discount=5;
        }else if(vehicleName.contains("Yamaha")){
            discount=7;
        }

        sp= h*(100-discount)/100;

        return sp;
    }else{

```

```
    return 0;
```

```
}
```

```
}
```

```
}
```

25)Global Hospital

```
import java.util.Scanner;  
import java.util.stream.Stream;  
import java.util.List;
```

```
public class Main {
```

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

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

```
        // Fill the code here
```

```
        ManagementUtility mu = new ManagementUtility();
```

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

```
        int n = Integer.parseInt(sc.nextLine());
```

```
        System.out.println("Enter the donor details");
```

```
        String[] Strings = new String[n];
```

```
        for(int i=0;i<n;i++)
```

```
{
```

```
        String donor = sc.nextLine();
```

```
        Strings[i] = donor;
```

```
}

List<Donor> res = mu.getDonorDetails(Strings);

Stream<Donor> result = mu.getStreamOfDonor(res);

System.out.println("Enter the blood group to search");

String bloodGroup = sc.nextLine();

List<String> finalResult = mu.shortlistedDonor(result,bloodGroup);

if(finalResult.size() != 0)

{

    System.out.println("List of shortlisted donors");

    finalResult.forEach(s ->{System.out.println(s);});

}

else

{

    System.out.println("No donor found");

}

}

-----
```

```
import java.util.ArrayList;

import java.util.Collections;

import java.util.stream.Stream;

import java.util.List;

public class ManagementUtility {
```

```
public List<Donor> getDonorDetails(String [] details) {  
  
    // Fill the code here  
  
    List<Donor> temp = new ArrayList<Donor>();  
  
    for(int i=0;i<details.length;i++)  
    {  
        String newTemp = details[i];  
        String[] splitString = newTemp.split(":");  
        long mb = Long.parseLong(splitString[3]);  
        temp.add(new Donor(splitString[0],splitString[1],splitString[2],mb));  
    }  
    return temp;  
}  
  
public Stream<Donor> getStreamOfDonor(List<Donor> donorDetails) {  
  
    // Fill the code here  
  
    return donorDetails.stream();  
}  
  
public List<String> shortlistedDonor(Stream<Donor> donorStream, String bloodGroup) {  
  
    // Fill the code here
```

```
        List<String> result = new ArrayList<String>();

        donorStream.filter(s-> s.getBloodGroup().equals(bloodGroup)).forEach(s ->
{result.add(s.getDonorName());});

        Collections.sort(result);

        return result;

    }

}
```

```
//Don't change the skeleton

public class Donor {

    private String donorId;

    private String donorName;

    private String bloodGroup;

    private long mobileNumber;

    public Donor(String id,String name,String group,long mn)

    {

        this.donorId = id;

        this.donorName = name;

        this.bloodGroup = group;

        this.mobileNumber = mn;

    }

    public String getDonorId() {

        return donorId;

    }
```

```
public void setDonorId(String donorId) {  
    this.donorId = donorId;  
}  
  
public String getDonorName() {  
    return donorName;  
}  
  
public void setDonorName(String donorName) {  
    this.donorName = donorName;  
}  
  
public String getBloodGroup() {  
    return bloodGroup;  
}  
  
public void setBloodGroup(String bloodGroup) {  
    this.bloodGroup = bloodGroup;  
}  
  
public long getMobileNumber() {  
    return mobileNumber;  
}  
  
public void setMobileNumber(long mobileNumber) {  
    this.mobileNumber = mobileNumber;  
}  
}
```

26)Hyper super market

```
import java.util.Scanner;

import java.util.stream.Stream;

import java.util.*;

public class Main {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int num = 0;

        ManagementUtility m=new ManagementUtility();

        while(num != 3){

            System.out.println("Select an option:\n1.Add Vegetable to Basket\n2.Calculate Bill\n3.Exit");

            num = sc.nextInt();

            if(num == 1){

                VegetableBasket vege=new VegetableBasket();

                System.out.println("Enter the vegetable name");

                String name=sc.next();

                System.out.println("Enter weight in Kgs");

                int weight = sc.nextInt();

                System.out.println("Enter price per Kg");

                int price = sc.nextInt();

                vege.setVegetableName(name);
```

```
vege.setWeightInKgs(weight);

vege.setPricePerKg(price);

m.addToBasket(vege);

}

else if(num==2)

{

List<VegetableBasket>vegeBasketList =m.getVegetableBasketList();

if(vegeBasketList.size()==0)

{

System.out.println("Customer basket is empty. Please add vegetables.");

}

else{

    Stream<VegetableBasket>vegetableBasketStream = vegeBasketList.stream();

    int amount = m.calculateBill(vegetableBasketStream);

    System.out.println("The estimated bill amount is Rs "+amount);

}

}

System.out.println("Thank you for using the application.");

}



---



//Don't change the skeleton


```

```
public class VegetableBasket {  
    private String vegetableName;  
    private int weightInKgs;  
    private int pricePerKg;  
  
    public String getVegetableName() {  
        return vegetableName;  
    }  
  
    public void setVegetableName(String vegetableName) {  
        this.vegetableName = vegetableName;  
    }  
  
    public int getWeightInKgs() {  
        return weightInKgs;  
    }  
  
    public void setWeightInKgs(int weightInKgs) {  
        this.weightInKgs = weightInKgs;  
    }  
  
    public int getPricePerKg() {  
        return pricePerKg;  
    }  
  
    public void setPricePerKg(int pricePerKg) {  
        this.pricePerKg = pricePerKg;  
    }  
}
```

```
}
```

```
import java.util.List;  
  
import java.util.stream.Stream;  
  
import java.util.*;  
  
import java.util.stream.Collectors;  
  
import java.util.function.Function;  
  
public class ManagementUtility {  
  
    private List<VegetableBasket> vegetableBasketList = new ArrayList<VegetableBasket>();  
  
  
    public List<VegetableBasket> getVegetableBasketList() {  
  
        return this.vegetableBasketList;  
    }  
  
  
    public void setVegetableBasketList(List<VegetableBasket> vegetableBasketList) {  
  
        this.vegetableBasketList = vegetableBasketList;  
    }  
  
  
    public void addToBasket(VegetableBasket fbObj) {  
  
        List<VegetableBasket> vge= getVegetableBasketList();  
  
        vge.add(fbObj);  
  
        setVegetableBasketList(vge);  
  
        //Fill the code here  
    }  
}
```

```

public static int calculateBill(Stream<VegetableBasket> vegetableBasketStream) {
    Map<Integer, Integer> V = vegetableBasketStream.collect(Collectors.toMap(VegetableBasket->
VegetableBasket.getWeightInKgs(), VegetableBasket -> VegetableBasket.getPricePerKg()));

    int amount=0;

    for(Map.Entry<Integer, Integer>entry : V.entrySet()) {
        amount=amount+(entry.getKey()*entry.getValue());
    }

    return amount;
}

}

```

27)Food cart

```

import java.io.File; // Import the File class

import java.io.FileNotFoundException; // Import this class to handle errors

import java.util.Scanner;

public class Main

{

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        int flag=1;

        try {

            File myObj = new File("customer.txt");

            Scanner myReader = new Scanner(myObj);

```

```
String data = myReader.nextLine();

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

String Value = sc.next();

while (myReader.hasNextLine())

{

    final String lineFromFile = myReader.nextLine();

    if(lineFromFile.contains(Value))

    {

        // System.out.println("customer.txt");

        System.out.println(lineFromFile);

        flag=0;

        break;

    }

}

if (flag==1)

{

    System.out.println("No customer found");

}

myReader.close();

} catch (FileNotFoundException e) {
```

```
        System.out.println("An error occurred.");
        e.printStackTrace();
    }
}
```

Robert:7329102839:Pizza:2:2000

Advaith:8329103940:BBQ:1:350

Thanos:9762453210:Burger:4:570

Lewi:9036524133:Pizza:2:633

Albert:6852416398:BBQ:6:2000

Rahul:6663410275:Burger:4:451

Robert:8763957410:Pizza:2:1026

Jerold:9096741023:BBQ:2:457

Virat:8078451296:Burger:4:600

Dhoni:9363021245:Pizza:2:974

Praga:7685963210:BBQ:4:540

Aravind:6241053689:Burger:4:570

Praveen:8956451203:Pizza:2:3201

Issac:6410258963:BBQ:1:2000

Manish:9512023654:Burger:4:941

Ranjith:7456899874:BBQ:1:720

Akash:6352987451:Burger:4:570

Rakesh:8090461375:Pizza:8:610

Anees:9080708090:BBQ:7:350

Alex:6501200332:Burger:4:340

28)Fuel

```
@FunctionalInterface

public interface FuelCalc {

    public double calculateFuel(String fuelName,double litres);

}

import java.util.*;

public class UserInterface {

    public static FuelCalc generateRate()

    {

        return (String fuelName, double litres)->

        { double price=0;

            fuelName=fuelName.toLowerCase();

            if(fuelName.equals("petrol"))

                price=litres*90.4;

            else if(fuelName.equals("diesel"))

                price=litres*86.8;

            return price;

        };

    }

    public static void main(String [] args)

    {

        Scanner sc=new Scanner(System.in);

        System.out.println("Enter the type of fuel");

        String s=sc.next();

        System.out.println("Enter litres");
```

```

        double l=sc.nextDouble();

        FuelCalc f=generateRate();

        double p=f.calculateFuel(s,l);

System.out.printf("Total Cost of the fuel =Rs. %.2f",p);

    }

}

```

29)College account

```

public interface TuitionFee {

    int calculateTuitionFees(String courseType, int basicFee, int noOfSemesters);

}

```

```

import java.util.Scanner;

public class UserInterface {

    public static TuitionFee generateFeeReceipt() {

        return (String courseType, int basicFee, int noOfSemesters) -> {

            int regularFee = basicFee * noOfSemesters;

            if (courseType.equalsIgnoreCase("regular")) {

                return regularFee;

            } else if (courseType.equalsIgnoreCase("selfFinance")) {

                return regularFee + 50000;

            } else {

                return 0;
            }
        }
    }
}

```

```
        }

    };

}

public static void main(String[] args) {

    Scanner scanner = new Scanner(System.in);

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

    int registrationNumber = scanner.nextInt();

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

    String studentName = scanner.next();

    System.out.println("Enter no of semesters");

    int noOfSemesters = scanner.nextInt();

    System.out.println("Enter basic fee");

    int basicFee = scanner.nextInt();

    System.out.println("Course type");

    String courseType = scanner.next();

    TuitionFee tuitionFee = generateFeeReceipt();

    int tuitionFeeAmount = tuitionFee.calculateTuitionFees(courseType, basicFee, noOfSemesters);

    System.out.println("Fees Receipt");

    System.out.println("Registration number: "+ registrationNumber);

    System.out.println("Student name: " + studentName);

    System.out.println(String.format("Tuition fee for %s student: %d", courseType, tuitionFeeAmount));

}

}
```

29)Water distributor

```
public class Container {  
    public String distributorName;  
    public int volume;  
    public int count;  
  
    public String getDistributorName() {  
        return distributorName;  
    }  
  
    public void setDistributorName(String distributorName) {  
        this.distributorName = distributorName;  
    }  
  
    public int getVolume() {  
        return volume;  
    }  
  
    public void setVolume(int volume) {  
        this.volume = volume;  
    }  
  
    public int getCount() {  
        return count;  
    }  
  
    public void setCount(int count) {  
        this.count = count;  
    }  
  
    public Container(String distributorName, int volume, int count) {
```

```
super();  
  
this.distributorName = distributorName;  
  
this.volume = volume;  
  
this.count = count;  
  
}  
  
}
```

```
public interface DiscountInfo {  
  
    // Fill the code  
  
    public double calculatePayableAmount (Container containerObj);  
  
}
```

```
import java.util.Scanner;  
  
public class UserInterface {  
  
    public static DiscountInfo generateBillAmount() {  
  
        return(Container containerobj)->  
  
        { double dp=0;  
  
            if(containerobj.count>=100){  
  
                if(containerobj.volume==10)  
  
                    dp=(20*containerobj.count)-(1*containerobj.count*20);  
  
                else if(containerobj.volume==25)  
  
                    dp=50*containerobj.count-(.15*containerobj.count*50);  
  
            }  
  
            else{  
  
                if(containerobj.volume==10)
```

```

dp=(containerobj.count*20);

else if(containerobj.volume==25)

dp= (containerobj.count*50);

}

return dp;

};

}

public static void main(String args[]) {

Scanner sc=new Scanner(System.in);

System.out.println("Enter the name of the distributor");

String s=sc.next();

System.out.println("Enter the volume of the container(in litre)");

int d=sc.nextInt();

System.out.println("Enter the no of containers");

int h=sc.nextInt();




if(d==25 || d==10)

{

Container obj=new Container(s,d,h);

DiscountInfo k=generateBillAmount();

double g=k.calculatePayableAmount (obj);

System.out.println("Generated Bill Amount");

System.out.println("Distributor name: "+s);

System.out.printf("Amount to be paid: Rs.%,.2f",g);

}

```

```
        else
        {
            System.out.println("There is no Discount");
        }

    }

}

31)Book sale

// DO NOT EDIT OR ADD ANY CODE

public class Book {

    private String bookName;

    private String authorName;

    private double bookCost;

    //Setters and Getters

    public String getBookName() {
        return bookName;
    }

    public void setBookName(String bookName) {
        this.bookName = bookName;
    }

    public String getAuthorName() {
        return authorName;
    }

    public void setAuthorName(String authorName) {
```

```
        this.authorName = authorName;  
    }  
  
    public double getBookCost() {  
        return bookCost;  
    }  
  
    public void setBookCost(double bookCost) {  
        this.bookCost = bookCost;  
    }  
  
    @Override  
  
    public String toString() {  
        return "Book [bookName=" + bookName + ", authorName=" + authorName + ",  
bookCost=" + bookCost + "]";  
    }  
  
}
```

```
import java.util.List;  
  
import java.util.stream.Stream;  
  
import java.util.*;  
  
import java.util.Map;  
  
import java.util.stream.Collectors;  
  
import java.util.Collections;  
  
public class BookUtility {  
  
    private List<Book> bookList;
```

```

public List<Book> getBookList(){
    return this.bookList;
}

public void setBookList(List<Book> bookList){
    this.bookList = bookList;
}

//FILL THE CODE HERE

public Stream<Book> listToStream() {
    //FILL THE CODE HERE
    return getBookList().stream();
}

public List<Double> calculateBookDiscount(Stream<Book> stream1) {
    List<Double> ans = stream1.map(Book::getBookCost).collect(Collectors.toList());
    Double MUTIPLY_ELEMENT=0.90;
    List<Double> answer = ans.parallelStream().map(in -> in *
        MUTIPLY_ELEMENT).collect(Collectors.toList());
    return answer;
}
}

```

```

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

public class UserInterface {

    public static void main(String [] args)
    {

```

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

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

int N=sc.nextInt();

if(N>0)

{

List<Book> list = new ArrayList<Book>();

System.out.println("Enter book details");

sc.nextLine();

String arr[]=new String[N];

for(int i=0;i<N;i++)

{

arr[i]=sc.nextLine();

}

for(int i=0;i<N;i++)

{

Book b=new Book();

String[] word=arr[i].split ":";

b.setAuthorName(word[1]);

b.setBookName(word[0]);

double v=Double.parseDouble(word[2]);

b.setBookCost(v);

list.add(b);

}

BookUtility B=new BookUtility();

B.setBookList(list);
```

```
Stream<Book> stream=B.listToStream();

List<Double> answer=B.calculateBookDiscount(stream);

System.out.println("Updated Book Cost:");

for(int i=0;i<N;i++)

{

System.out.println(answer.get(i));

}

}

else{

System.out.println("Invalid Entry");

}

}

}

}

32)Association contest
```

//DO NOT EDIT OR ADD ANY CODE

```
public class Contestant {

    private String id;

    private String name;

    private String address;
```

//setters & getters

```
public String getId() {

    return id;
```

```
}

public void setId(String id) {
    this.id = id;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getAddress() {
    return address;
}

public void setAddress(String address) {
    this.address = address;
}

}
```

```
import java.util.List;
import java.util.Map;
import java.util.*;
```

```
import java.util.stream.Stream;
import java.util.stream.Collectors;

public class ContestUtility {

    private List<Contestant> ContestList;

    public List<Contestant> getContestList(){
        return this.ContestList;
    }

    public void setContestList(List<Contestant> ContestList)
    {
        this.ContestList=ContestList;
    }

    public Stream<Contestant> changeToStream() {
        List<Contestant> list = getContestList();

        return list.stream();
    }

    public List searchFromDetails(Stream<Contestant> stream1,char varString) {
        Map<String,String>file = stream1.collect(Collectors.toMap(Contestant->Contestant.getId(),Contestant->Contestant.getName()));

        List<String> l = new ArrayList<String>();
        for (Map.Entry<String,String> entry : file.entrySet())
        {
            String s=Character.toString(varString);
            if(entry.getValue().toLowerCase().startsWith(s.toLowerCase())))

```

```
{  
    l.add(entry.getKey());  
}  
  
}  
  
return l;  
  
}  


---



```
import java.util.*;
import java.util.stream.Stream;
public class UserInterface {
```


```

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

    System.out.println("Enter the number of contestant details to enter:");

    int N=sc.nextInt();

    if(N>0)

    {

        System.out.println("Enter the contestant details:");

        List<Contestant> ContestList = new ArrayList<Contestant>();

        ContestUtility C = new ContestUtility();

        for(int i=0;i<N;i++)

        {

            String arr=sc.next();

            String[] word=arr.split("/");

            String id=word[0];

            String name = word[1];

            String place=word[2];

            Contestant c=new Contestant();

            c.setId(id);

            c.setName(name);

            c.setAddress(place);

            ContestList.add(c);

        }

        C.setContestList(ContestList);
    }
}
```

```
Stream<Contestant>stream = C.changeToStream();

System.out.println("Enter character to search");

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

List list=C.searchFromDetails(stream,alpha);

List<Contestant> Contest = C.getContestList();

if(list.size()!=0)

{

    System.out.println("Filter the names which starts with "+alpha+":");

    for(int i=0;i<N;i++)

    {

        if(list.contains(Contest.get(i).getId()))

        {

            System.out.println("Id: "+Contest.get(i).getId()+" , Name:

"+Contest.get(i).getName());

        }

    }

}

else{

    System.out.println("No record found");

}

}

else{

    System.out.println("Invalid");
}
```

```
    }

}

}

33) vehicle registration

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class UserInterface {

    public static void main(String [] args){

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the number of vehicle details:");

        int a = Integer.parseInt(sc.nextLine());

        if (a<=0){

            System.out.println("Invalid");

        }

    }

    else{

        System.out.println("Enter the vehicle details:");

        String [] x = new String[a];

        List<Vehicle> clm=new ArrayList<Vehicle>();

        for (int j=0;j<a;j++){

            x[j] =sc.nextLine();

            String[] p = x[j].split("/");

            Vehicle cd = new Vehicle();

            cd.setVtype(p[0]);

            cd.setVno(p[1]);

            cd.setCname(p[2]);

            cd.setCnic(p[3]);

            cd.setCadd(p[4]);

            cd.setCtel(p[5]);

            clm.add(cd);

        }

        System.out.println("Total "+clm.size()+" vehicles registered");

        System.out.println("Enter the vehicle number to search");

        String vno = sc.nextLine();

        Vehicle v = null;

        for (int i=0;i<x.length;i++){

            if (x[i].contains(vno)) v = clm.get(i);

        }

        if (v==null) System.out.println("Vehicle not found");

        else System.out.println("Vehicle found");

    }

}
```

```

        cd.setVehicleNo(Integer.parseInt(p[0]));
        cd.setVehicleHolderName(p[1]);
        cd.setVehicleHolderAddress(p[2]);
        clm.add(cd);

    }

    VehicleUtility obj = new VehicleUtility();
    obj.setVehicleList(clm);
    System.out.println("Enter length of the number to search");
    int uh = Integer.parseInt(sc.nextLine());
    List<Vehicle> ans = obj.filterVehicleDetails(obj.convertToStream(),uh);
    if (ans.isEmpty()){
        System.out.println("No vehicle found");

    }

    else{
        System.out.println("Filter the vehicle numbers which has a length of "+uh+":");

        for (int g=0;g<ans.size();g++){
            System.out.println("VehicleNumber: "+ans.get(g).getVehicleNo()+", Name: "+ans.get(g).getVehicleHolderName());
        }
    }
}

```

```
}
```

```
}
```

```
// DO NOT EDIT OR ADD ANY CODE
```

```
public class Vehicle {
```

```
    private int vehicleNo;
```

```
    private String vehicleHolderName;
```

```
    private String vehicleHolderAddress;
```

```
//Setters and Getterss
```

```
    public int getVehicleNo() {
```

```
        return vehicleNo;
```

```
}
```

```
    public void setVehicleNo(int vehicleNo) {
```

```
        this.vehicleNo = vehicleNo;
```

```
}
```

```
    public String getVehicleHolderName() {
```

```
        return vehicleHolderName;
```

```
}
```

```
    public void setVehicleHolderName(String vehicleHolderName) {
```

```
        this.vehicleHolderName = vehicleHolderName;
```

```
}
```

```
    public String getVehicleHolderAddress() {
```

```
        return vehicleHolderAddress;  
    }  
  
    public void setVehicleHolderAddress(String vehicleHolderAddress) {  
        this.vehicleHolderAddress = vehicleHolderAddress;  
    }  
  
    @Override  
    public String toString() {  
        return "Vehicle [vehicleNo=" + vehicleNo + ", vehicleHolderName=" +  
vehicleHolderName  
                + ", vehicleHolderAddress=" + vehicleHolderAddress + "]";  
    }  
  
}  
  
-----  
  
import java.util.Comparator;  
  
import java.util.List;  
  
import java.util.stream.Collectors;  
  
import java.util.stream.Stream;public class VehicleUtility {  
  
    private List<Vehicle> vehicleList;  
  
    //FILL THE CODE HERE  
  
    public List<Vehicle> getVehicleList(){  
  
        return vehicleList;  
    }  
  
    public void setVehicleList(List<Vehicle> vehicleList){
```

```
this.vehicleList=vehicleList;

}

public Stream<Vehicle> convertToStream() {

    Stream<Vehicle> w =vehicleList.stream();

    return w;

}

public List filterVehicleDetails(Stream<Vehicle> stream1,int length){

    List<Vehicle> shot=stream1.filter(x->String.valueOf(x.getVehicleNo()).length()>=length).sorted(Comparator.comparingInt(Vehicle::getVehicleNo)).collect(Collectors.toList());

    return shot;

}

}

}

34) Library management
```

```
//DO NOT EDIT OR ADD ANY CODE HERE

public class Book {

    private String bookId;

    private String bookName;

    private String bookType;

    private String bookCategory;

    private String bookEdition;
```

```
public Book() {
    super();

}

public Book(String bookId, String bookName, String bookType, String bookCategory, String
bookEdition) {
    super();
    this.bookId = bookId;
    this.bookName = bookName;
    this.bookType = bookType;
    this.bookCategory = bookCategory;
    this.bookEdition = bookEdition;
}

public String getBookId() {
    return bookId;
}

public void setBookId(String bookId) {
    this.bookId = bookId;
}

public String getBookName() {
    return bookName;
}
```

```
public void setBookName(String bookName) {  
    this.bookName = bookName;  
}  
  
public String getBookType() {  
    return bookType;  
}  
  
public void setBookType(String bookType) {  
    this.bookType = bookType;  
}  
  
public String getBookEdition() {  
    return bookEdition;  
}  
  
public void setBookEdition(String bookEdition) {  
    this.bookEdition = bookEdition;  
}  
  
public String getBookCategory() {  
    return bookCategory;  
}  
  
public void setBookCategory(String bookCategory) {  
    this.bookCategory = bookCategory;  
}  
  
}
```

```
import java.util.ArrayList;
import java.util.List;
public class BookUtility implements Runnable{
    private List<Book> bookList = new ArrayList<Book>();
    private String searchbookName;
    private int counter;
    //FILL THE CODE HERE
    public List<Book> getBookList(){
        return bookList;
    }
    public String getSearchbookName(){
        return searchbookName;
    }
    public void setBookList(List<Book> a){
        this.bookList=a;
    }
    public void setSearchbookName(String x){
        this.searchbookName=x;
    }
    public int getCounter(){
```

```
    return counter;

}

public void setCounter(int counter){

    this.counter=counter;

}

public void toValidateBookType(Book obj) throws InvalidBookException{

    //FILL THE CODE

    if (obj.getBookType().compareToIgnoreCase("Engineering")==0){

        bookList.add(obj);

    }

    else{

        throw new InvalidBookException("Book type Invalid");

    }

}

public void run(){

    //FILL THE CODE HERE

    int count=0;

    for(int j=0;j<bookList.size();j++){

        if (bookList.get(j).getBookName().compareToIgnoreCase(searchbookName)==0){

            count=count+1;

        }

    }

}
```

```
    }

}

if (count==0) {

    System.out.println("No Books found");

}

else {

    System.out.println("Count of books in the library with the book name
"+searchbookName.toUpperCase()+" is "+count);

}

}

-----  
public class InvalidBookException extends Exception{

//FILL THE CODE

String s;

public InvalidBookException (String s){

    super(s);

}

}
```

```
}
```

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;public class UserInterface {

    public static void main(String [] args){

        Scanner sc=new Scanner(System.in);

        BookUtility ans = new BookUtility();

        // FILL THE CODE HERE

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

        int a = Integer.parseInt(sc.nextLine());

        System.out.println("Enter the bookId/bookName/bookType/bookCategory/bookEdition");

        String [] x = new String[a];

        List<Book> clm=new ArrayList<Book>();

        boolean flag = false;

        for (int j =0;j<a;j++){

            x[j] =sc.nextLine();

            String[] p = x[j].split("/");

            Book cd= new Book();

            cd.setBookId(p[0]);

            cd.setBookName(p[1]);

            cd.setBookType(p[2]);

            cd.setBookCategory(p[3]);

            cd.setBookEdition(p[4]);

            try{
```

```
ans.toValidateBookType(cd);

clm.add(cd);

}

catch(InvalidBookException e){

    flag=true;

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

}

}

if (flag==false){

    ans.setBookList(clm);

    System.out.println("Print the book details");

    for(int y=0;y<clm.size();y++){

        System.out.println("bookId="+clm.get(y).getBookId()+
bookName="+clm.get(y).getBookName()+" , bookType="+clm.get(y).getBookType());

    }

    System.out.println("Enter book name to search");

    String hh = sc.nextLine();

    ans.setSearchbookName(hh);

    Thread myThread = new Thread(ans);

    myThread.run();

}

}
```

}

}

ProductOrderDAO.java

```
package com.cts.zepcpd.dao;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;
import com.cts.zepcpd.exception.ProductOrderException;
import com.cts.zepcpd.util.*;
import com.cts.zepcpd.vo.ProductOrder;

public class ProductOrderDAO {

    public boolean addProductOrderDetails(List<ProductOrder> pdtOrder) throws
ProductOrderException {
        boolean recordsAdded = false;

        //Code here..
        Connection con = DBConnectionManager.getInstance().getConnection();
        PreparedStatement ps = null;
        try {
            // inserting values of list pdtOrders into database
            String query = "insert into ZEPC_MANAGE_ORDER (ORDER_ID,
PRODUCT_CODE, DATE_OF_ORDER, PRODUCT_LEVEL, DATE_OF_DELIVERY, NO_OF_PRODUCTS,
NO_OF_KMS_FOR_DELIVERY,MANAGER_APPROVAL, PRODUCT_COST, GST_TAX, DELIVERY_COST,
TOTAL_ORDER_COST, FINAL_STATUS_OF_ORDER)values(?,?,?,?,?,?,?,?,?,?,
?,?,?,?,?)";
            for (ProductOrder e : pdtOrder) {
                ps = con.prepareStatement(query);
                ps.setString(1, e.getOrderId());
                ps.setString(2, e.getProductCode());
                ps.setDate(3,
ApplicationUtil.convertUtilToSqlDate(e.getDateOfOrder()));
                ps.setString(4, e.getProductLevel());
                ps.setDate(5,
ApplicationUtil.convertUtilToSqlDate(e.getDateOfDelivery()));
                ps.setInt(6, e.getNoOfProducts());
                ps.setDouble(7, e.getNoOfKmsForDelivery());
                ps.setString(8, e.getManagerApproval());
                ps.setDouble(9, e.getProductCost());
                ps.setDouble(10, e.getGstTax());
                ps.setDouble(11, e.getDeliveryCost());
                ps.setDouble(12, e.getTotalOrderCost());
                ps.setString(13, e.getFinalStatusOfOrder());
                int i = ps.executeUpdate();
                if (i > 0) {
                    recordsAdded = true;
                } else {
                    recordsAdded = false;
                }
            }
        } catch (SQLException e) {
            try {
                con.rollback();
            } catch (Exception e1) {
                e.printStackTrace();
            }
        } catch (Exception e) {
```

```

        e.printStackTrace();
        // throw new ProductOrderException("Database Value Insertion
Failed",
        // e.getCause());

    } finally {
        try {
            ps.close();
            con.close();
        } catch (Exception e) {
            e.printStackTrace();
            // throw new ProductOrderException("Database Value
Insertion Failed",
            // e.getCause());
        }
    }

    return recordsAdded;
}

public List<ProductOrder> getAllProductOrderDetails() throws
ProductOrderException {
    List<ProductOrder> pdtOrder = new ArrayList<ProductOrder>();

    //Code here..
    // Retrieval of all records from database
    String query = "select * from ZEPC_MANAGE_ORDER";
    try (Connection con =
DBConnectionManager.getInstance().getConnection()) {
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery(query));
        while (rs.next()) {
            // storing retrieved records in object
            ProductOrder obj = new ProductOrder();
            obj.setOrderId(rs.getString(1));
            obj.setProductCode(rs.getString(2));
            obj.setDateOfOrder(new
java.util.Date(rs.getDate(3).getTime()));
            obj.setProductLevel(rs.getString(4));
            obj.setDateOfDelivery(new
java.util.Date(rs.getDate(5).getTime()));
            obj.setNoOfProducts(rs.getInt(6));
            obj.setNoOfKmsForDelivery(rs.getDouble(7));
            obj.setManagerApproval(rs.getString(8));
            obj.setProductCost(rs.getDouble(9));
            obj.setGstTax(rs.getDouble(10));
            obj.setDeliveryCost(rs.getDouble(11));
            obj.setTotalOrderCost(rs.getDouble(12));
            obj.setFinalStatusOfOrder(rs.getString(13));
            // adding ProductOrders object into arraylist
            pdtOrder.add(obj);
        }

    } catch (Exception e) {
        e.printStackTrace();
        // throw new StudentAdmissionException("Database Value
Retrieval Failed",
    }
}

```

```

        // e.getCause());
    }

    return pdtOrder;
}

}

```

ProductOrderException.java

```

package com.cts.zepcpd.exception;

public class ProductOrderException extends Exception {

    private static final long serialVersionUID = -1105431869622052445L;

    /**
     * @param message
     * @param cause
     */
    public ProductOrderException(String message, Throwable cause) {
        super(message, cause);
    }
}

```

MainApp.java

```

package com.cts.zepcpd.main;

import com.cts.zepcpd.service.*;
import com.cts.zepcpd.skeletonvalidator.SkeletonValidator;
import com.cts.zepcpd.util.*;

public class MainApp {

    private MainApp() {

    }

    public static void main(String[] args) {
        //Don't delete this code
        //Skeletonvalidaton starts
        new SkeletonValidator();
        //Skeletonvalidation ends
        //Write your code here..
        try {
            ProductOrderService service = new ProductOrderService();

            System.out.println(service.addProductOrderDetails("inputFeed.txt"));
            System.out.println(service.searchProductOrder("R005"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```

        // List<StudentAdmission> studentAdmissionList =
        //
service.buildStudentAdmissionsList(ApplicationUtil.readFile("inputFeed.txt"));
        /*
         * for(StudentAdmission e:studentAdmissionList) {
System.out.println(e); }
        */
    }

}

```

ProductOrderService.java

```

package com.cts.zepcpd.service;

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

import com.cts.zepcpd.dao.*;
import com.cts.zepcpd.exception.ProductOrderException;
import com.cts.zepcpd.util.*;
import com.cts.zepcpd.vo.ProductOrder;

public class ProductOrderService {

    /**
     * @param productOrderRecords
     * @return List<ProductOrder>
     */
    public static List<ProductOrder> buildProductOrdersList(List<String>
productOrderRecords) {
        List<ProductOrder> productOrderList = new ArrayList<ProductOrder>();

        //Code here..
        for (String e : productOrderRecords) {
            String res[] = e.split(",");
            String orderId = res[0];
            String productCode = res[1];
            String dateOfOrder = res[2];    //DateType
            String productLevel = res[3];
            String dateOfDelivery = res[4];  //DateType
            String noOfProducts= res[5];
            String noOfKmsForDelivery = res[6];
            String managerApproval = res[7];
            ProductOrder obj = new ProductOrder();
            obj.setOrderId(orderId);
            obj.setProductCode(productCode);

            // converting String to java.util.Date

            obj.setDateOfOrder(ApplicationUtil.convertStringToDate(dateOfOrder));
            obj.setProductLevel(productLevel);

            // converting String to java.util.Date

            obj.setDateOfDelivery(ApplicationUtil.convertStringToDate(dateOfDelivery));
            obj.setNoOfProducts(Integer.parseInt(noOfProducts));
        }
    }
}

```

```

        obj.setNoOfKmsForDelivery(Double.parseDouble(noOfKmsForDelivery));
        obj.setManagerApproval(managerApproval);
        double[] productOrderCosts =
            calculateTotalOrderCost(
                Integer.parseInt(noOfProducts),
                Double.parseDouble(noOfKmsForDelivery), productLevel );

        obj.setProductCost(productOrderCosts[0]);
        obj.setGstTax(productOrderCosts[1]);
        obj.setDeliveryCost(productOrderCosts[2]);
        obj.setTotalOrderCost(productOrderCosts[3]);
        obj.setFinalStatusOfOrder("OrderSuccessfull");
        productOrderList.add(obj);
    }
    return productOrderList ;
}

public boolean addProductOrderDetails(String inputFeed) throws
ProductOrderException {
    List<ProductOrder> productOrderList = ProductOrderService
        .buildProductOrdersList(ApplicationUtil.readFile(inputFeed));
    ProductOrderDAO stdDao = new ProductOrderDAO();
    return stdDao.addProductOrderDetails(productOrderList);
//Code here..

    //TODO change this return value
}

public static double[] calculateTotalOrderCost(int noOfProducts, double
noOfKmsForDelivery, String productLevel) {
    double[] productOrderCosts = new double[4];

    if ("Level01".equals(productLevel)) {
        productOrderCosts[0] = 500;
        productOrderCosts[1] = 2;
        productOrderCosts[2] = 8;
    } else if ("Level02".equals(productLevel)) {
        productOrderCosts[0] = 600;
        productOrderCosts[1] = 3;
        productOrderCosts[2] = 10;
    } else if ("Level03".equals(productLevel)) {
        productOrderCosts[0] = 800;
        productOrderCosts[1] = 5;
        productOrderCosts[2] = 11;
    } else if ("Level04".equals(productLevel)) {
        productOrderCosts[0] = 1200;
        productOrderCosts[1] = 7;
        productOrderCosts[2] = 13;
    } else if ("Level05".equals(productLevel)) {
        productOrderCosts[0] = 1750;
        productOrderCosts[1] = 8;
        productOrderCosts[2] = 14;
    } else if ("Level06".equals(productLevel)) {
        productOrderCosts[0] = 2500;
    }
}

```

```

        productOrderCosts[1] = 9;
        productOrderCosts[2] = 14;
    }
    productOrderCosts[0] =
Math.round(productOrderCosts[0]*noOfProducts*100.0)/100.0;
    productOrderCosts[1] = Math.round((productOrderCosts[1] *
productOrderCosts[0])/100*100.0)/100.0;
    productOrderCosts[2] =
Math.round(productOrderCosts[2]*noOfKmsForDelivery*100.0)/100.0;
    productOrderCosts[3] =
productOrderCosts[0]+productOrderCosts[1]+productOrderCosts[2];

return productOrderCosts;
}

public boolean searchProductOrder(String orderId) throws
ProductOrderException {
    boolean status = false;

//Code here..
ProductOrderDAO ptdDao = new ProductOrderDAO();
List<ProductOrder> ptdOrders = ptdDao.getAllProductOrderDetails();
for (ProductOrder e : ptdOrders ) {
    if (e.getOrderId().equals(orderId)) {
        status = true;
        System.out.println(e);
        break;
    }
    else {
        System.out.println("Order Request not Found");
    }
}
return status;
}
}

```

ApplicationUtil.java

```

package com.cts.zepcpd.util;

import java.io.*;
import java.text.*;
import java.util.*;

import com.cts.zepcpd.exception.ProductOrderException;

public class ApplicationUtil {

```

```

/**
 * @param fileName
 * @return List<String>
 * @throws ProductOrderException
 */
public static List<String> readFile(String fileName) throws
ProductOrderException {
    List<String> productOrderList = new ArrayList<String>();
    //Code Here
    FileReader fr = null;
    BufferedReader br = null;
    try {
        fr = new FileReader(fileName);
        br = new BufferedReader(fr);
        String line = null;
        while ((line = br.readLine()) != null) {
            String[] res = line.split(",");
            String managerApproval = res[7];
            Date dtOfOrder = convertStringToDate(res[2]);
            Date dtOfDelivery = convertStringToDate(res[4]);
            if (checkIfValidOrder(dtOfOrder,dtOfDelivery,
managerApproval)) {
                productOrderList.add(line);
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return productOrderList;
}

```

```

/**
 * @param util
 *          Date
 * @return sql Date
 */
public static java.sql.Date convertUtilToSqlDate(java.util.Date uDate) {
    java.sql.Date sDate = new java.sql.Date(uDate.getTime());

    //Code here..

    return sDate;
}

/**
 * @param inDate
 * @return Date
 */
public static Date convertStringToDate(String inDate) {
    //Code here..
    try {
        SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd",
Locale.ENGLISH);
        return format.parse(inDate);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

```

        return null;
    }

}

public static boolean checkIfValidOrder(Date dtOfOrder, Date dtOfDelivery,
String manager) {
    boolean orderValidity = false;

    //Code here..
    if ("Approved".equals(manager)&& ((dtOfDelivery.getTime() -
dtOfOrder.getTime()) / (1000 * 60 * 60 * 24)) % 365 >=7) {
        orderValidity = true;
    }
    return orderValidity;
}
}

```

DBConnectionManager.java

```

* Don't change this code
*/
package com.cts.zepcpd.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.zepcpd.exception.ProductOrderException;

public class DBConnectionManager {

    private static DBConnectionManager instance;

    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 ProductOrderException
         */
        private DBConnectionManager() throws ProductOrderException {
            loadProperties();
            try {
                Class.forName(props.getProperty(DRIVER));
                this.connection =
                    DriverManager.getConnection(props.getProperty(URL),
                        props.getProperty(USER_NAME),
                        props.getProperty(PASSWORD));
            } catch (ClassNotFoundException ex) {
                throw new ProductOrderException("Could not find Driver
class ", ex.getCause());
            } catch (SQLException e) {
                throw new ProductOrderException("Database Connection
Creation Failed", e.getCause());
            }
        }

        /**
         * @return Connection
         */
        public Connection getConnection() {
            return connection;
        }
    }
}

```

```

    }

    /**
     * @return DBConnectionManager
     * @throws ProductOrderException
     */
    public static DBConnectionManager getInstance() throws
ProductOrderException {

    // Code here

        instance = new DBConnectionManager();

        return instance;

    }

    /**
     * @throws ProductOrderException
     */
    private void loadProperties() throws ProductOrderException {

        FileInputStream inputStream = null;
        try {
            inputStream = new FileInputStream(PROPERTY_FILE);
            props = new Properties();
            props.load(inputStream);
        } catch (FileNotFoundException e) {
            throw new ProductOrderException("Database Property File
Not Found", e.getCause());
        } catch (IOException e) {
            throw new ProductOrderException("Exception during
property file I/O", e.getCause());
        } finally {
            if (inputStream != null) {

```

```
        try {
            inputStream.close();
        } catch (IOException e) {
            throw new
ProductOrderException("Exception during property file I/O", e.getCause());
        }
    }
}
```

ProductOrder .Java

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

```
*/
```

```
package com.cts.zepcpd.vo;
```

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

```
public class ProductOrder {
```

```
    String orderId;
    String productCode;
    Date dateOfOrder;
    String productLevel;
    Date dateOfDelivery;
    int noOfProducts;
    double noOfKmsForDelivery;
    String managerApproval;
    double productCost;
    double gstTax;
    double deliveryCost;
    double totalOrderCost;
```

```
String finalStatusOfOrder;

public ProductOrder() {
    super();
}

public ProductOrder(String orderId, String productCode, Date dateOfOrder,
String productLevel, Date dateOfDelivery,
int noOfProducts, double noOfKmsForDelivery, String
managerApproval, double productCost, double gstTax,
double deliveryCost, double totalOrderCost, String
finalStatusOfOrder) {
    super();
    this.orderId = orderId;
    this.productCode = productCode;
    this.dateOfOrder = dateOfOrder;
    this.productLevel = productLevel;
    this.dateOfDelivery = dateOfDelivery;
    this.noOfProducts = noOfProducts;
    this.noOfKmsForDelivery = noOfKmsForDelivery;
    this.managerApproval = managerApproval;
    this.productCost = productCost;
    this.gstTax = gstTax;
    this.deliveryCost = deliveryCost;
    this.totalOrderCost = totalOrderCost;
    this.finalStatusOfOrder = finalStatusOfOrder;
}

public String getOrderId() {
    return orderId;
}
```

```
public void setOrderId(String orderId) {  
    this.orderId = orderId;  
}  
  
public String getProductCode() {  
    return productCode;  
}  
  
public void setProductCode(String productCode) {  
    this.productCode = productCode;  
}  
  
public Date getDateOfOrder() {  
    return dateOfOrder;  
}  
  
public void setDateOfOrder(Date dateOfOrder) {  
    this.dateOfOrder = dateOfOrder;  
}  
  
public String getProductLevel() {  
    return productLevel;  
}
```

```
public void setProductLevel(String productLevel) {  
    this.productLevel = productLevel;  
}
```

```
public Date getDateOfDelivery() {  
    return dateOfDelivery;  
}
```

```
public void setDateOfDelivery(Date dateOfDelivery) {  
    this.dateOfDelivery = dateOfDelivery;  
}
```

```
public int getNoOfProducts() {  
    return noOfProducts;  
}
```

```
public void setNoOfProducts(int noOfProducts) {  
    this.noOfProducts = noOfProducts;  
}
```

```
public double getNoOfKmsForDelivery() {  
    return noOfKmsForDelivery;  
}
```

```
public void setNoOfKmsForDelivery(double noOfKmsForDelivery) {  
    this.noOfKmsForDelivery = noOfKmsForDelivery;  
}
```

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

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

```
public double getProductCost() {  
    return productCost;  
}
```

```
public void setProductCost(double productCost) {  
    this.productCost = productCost;  
}
```

```
public double getGstTax() {  
    return gstTax;  
}
```

```
public void setGstTax(double gstTax) {
```

```
this.gstTax = gstTax;  
}  
  
public double getDeliveryCost() {  
    return deliveryCost;  
}  
  
public void setDeliveryCost(double deliveryCost) {  
    this.deliveryCost = deliveryCost;  
}  
  
public double getTotalOrderCost() {  
    return totalOrderCost;  
}  
  
public void setTotalOrderCost(double totalOrderCost) {  
    this.totalOrderCost = totalOrderCost;  
}  
  
public String getFinalStatusOfOrder() {  
    return finalStatusOfOrder;  
}  
  
public void setFinalStatusOfOrder(String finalStatusOfOrder) {  
    this.finalStatusOfOrder = finalStatusOfOrder;
```

```

    }

    @Override
    public String toString() {
        return "ZepcManagementOrder Details: [orderId=" + orderId + ",  

productCode=" + productCode + ", dateOfOrder="

                + dateOfOrder + ", productLevel=" + productLevel +  

", dateOfDelivery=" + dateOfDelivery + ", noOfProducts="

                + noOfProducts + ", noOfKmsForDelivery=" +  

noOfKmsForDelivery + ", managerApproval=" + managerApproval

                + ", productCost=" + productCost + ", gstTax=" +  

gstTax + ", deliveryCost=" + deliveryCost + ", totalOrderCost=" + totalOrderCost

                + ", finalStatusOfOrder=" + finalStatusOfOrder + "]";
    }
}

```

SkeletonValidator.java

```

package com.cts.zepcpd.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.zepcpd.util.DBConnectionManager");  
        validateClassName("com.cts.zepcpd.util.ApplicationUtil");  
        validateClassName("com.cts.zepcpd.service.ProductOrderService");  
        validateClassName("com.cts.zepcpd.dao.ProductOrderDAO");  
        validateClassName("com.cts.zepcpd.vo.ProductOrder");  
  
        validateClassName("com.cts.zepcpd.exception.ProductOrderException");  
  
        validateMethodSignature(  
            "addProductOrderDetails:boolean,getAllProductOrderDetails>List",  
            "com.cts.zepcpd.dao.ProductOrderDAO");  
        validateMethodSignature(  
            "buildProductOrdersList>List,addProductOrderDetails:boolean,calculateTotal  
OrderCost:double[],searchProductOrder:boolean",  
            "com.cts.zepcpd.service.ProductOrderService");  
        validateMethodSignature(  
            "readFile>List,convertUtilToSqlDate>Date,convertStringToDate>Date,checkIfV  
alidOrder:boolean",  
            "com.cts.zepcpd.util.ApplicationUtil");  
        validateMethodSignature(  
            "getConnection:Connection,getInstance:DBConnectionManager",  
            "com.cts.zepcpd.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.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");  
  
}  
  
}
```

Project Assignment For Fresher

User Persona

John is a recruiter. He has many companies to contact for hiring and has very busy schedule that leads many things slipping out of his mind.

To help John manage his companies you need to build a web application.

Problem Statement:

Create a web application which will perform following operations to help John to manage his companies. Where he can see all his data & find data.

- Sign Up & Login
- List of companies (Use: [Datatable](#) to show list)
- Add Company (with following fields)
 - Company Name
 - Company Website (Validation Required)
 - Company Phone Number
 - Company Address
 - Company City
 - Company State
 - Company Country
 - Industry List (Dropdown field with values Account, IT, Sales, Health Care)
- Edit Company
- Delete Company

Also he can be able to logout from the system

Tech Requirement:

- Use any CSS framework for design
- You can use any frontend framework React/Vue/Plain HTML
- Use Javascript
- Use Datatable plugin to show table (Use of Ajax to load data is preferable)
- PHP, Python, JAVA
- MySql

Project Submission:

Submit your project on following link.

<https://docs.google.com/forms/d/e/1FAIpQLSe63oHW6mR5oEINbaTF5vAF7H6DYVMYanenRMsCt59WK3dGFg/viewform>

Please upload your project documentation file & source code in .zip format (max 100 Mb).

Also please put filename in the below format before upload

Project Document:

your-full-name - Project Document.docx
eg. girish-chaudhari - Project Document.docx

Source Code:

your-full-name - Project Source code.zip
eg. girish-chaudhari - Source code.zip

Note:

- Prefer Mobile friendly UI, not necessary
- Please do not hesitate to clarify any queries and ideally as soon as possible.
- You can email us on admin.engg@recruitcrm.io, one of our engineer will connect with you to resolve your queries.
- You also need to submit the project document. Please see attached sample project doc.

We're really excited to be blown away by your work on this project & have you join the team! You're just a few steps away, All the best :)

ProductOrderDAO.java

```
package com.cts.zepcpd.dao;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;
import com.cts.zepcpd.exception.ProductOrderException;
import com.cts.zepcpd.util.*;
import com.cts.zepcpd.vo.ProductOrder;

public class ProductOrderDAO {

    public boolean addProductOrderDetails(List<ProductOrder> pdtOrder) throws
ProductOrderException {
        boolean recordsAdded = false;

        //Code here..
        Connection con = DBConnectionManager.getInstance().getConnection();
        PreparedStatement ps = null;
        try {
            // inserting values of list pdtOrders into database
            String query = "insert into ZEPC_MANAGE_ORDER (ORDER_ID,
PRODUCT_CODE, DATE_OF_ORDER, PRODUCT_LEVEL, DATE_OF_DELIVERY, NO_OF_PRODUCTS,
NO_OF_KMS_FOR_DELIVERY,MANAGER_APPROVAL, PRODUCT_COST, GST_TAX, DELIVERY_COST,
TOTAL_ORDER_COST, FINAL_STATUS_OF_ORDER)values(?,?,?,?,?,?,?,?,?,?,
?,?)";
            for (ProductOrder e : pdtOrder) {
                ps = con.prepareStatement(query);
                ps.setString(1, e.getOrderId());
                ps.setString(2, e.getProductCode());
                ps.setDate(3,
ApplicationUtil.convertUtilToSqlDate(e.getDateOfOrder()));
                ps.setString(4, e.getProductLevel());
                ps.setDate(5,
ApplicationUtil.convertUtilToSqlDate(e.getDateOfDelivery()));
                ps.setInt(6, e.getNoOfProducts());
                ps.setDouble(7, e.getNoOfKmsForDelivery());
                ps.setString(8, e.getManagerApproval());
                ps.setDouble(9, e.getProductCost());
                ps.setDouble(10, e.getGstTax());
                ps.setDouble(11, e.getDeliveryCost());
                ps.setDouble(12, e.getTotalOrderCost());
                ps.setString(13, e.getFinalStatusOfOrder());
                int i = ps.executeUpdate();
                if (i > 0) {
                    recordsAdded = true;
                } else {
                    recordsAdded = false;
                }
            }
        } catch (SQLException e) {
            try {
                con.rollback();
            } catch (Exception e1) {
                e.printStackTrace();
            }
        } catch (Exception e) {
```

```

        e.printStackTrace();
        // throw new ProductOrderException("Database Value Insertion
Failed",
        // e.getCause());

    } finally {
        try {
            ps.close();
            con.close();
        } catch (Exception e) {
            e.printStackTrace();
            // throw new ProductOrderException("Database Value
Insertion Failed",
            // e.getCause());
        }
    }

    return recordsAdded;
}

public List<ProductOrder> getAllProductOrderDetails() throws
ProductOrderException {
    List<ProductOrder> pdtOrder = new ArrayList<ProductOrder>();

    //Code here..
    // Retrieval of all records from database
    String query = "select * from ZEPC_MANAGE_ORDER";
    try (Connection con =
DBConnectionManager.getInstance().getConnection()) {
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery(query));
        while (rs.next()) {
            // storing retrieved records in object
            ProductOrder obj = new ProductOrder();
            obj.setOrderId(rs.getString(1));
            obj.setProductCode(rs.getString(2));
            obj.setDateOfOrder(new
java.util.Date(rs.getDate(3).getTime()));
            obj.setProductLevel(rs.getString(4));
            obj.setDateOfDelivery(new
java.util.Date(rs.getDate(5).getTime()));
            obj.setNoOfProducts(rs.getInt(6));
            obj.setNoOfKmsForDelivery(rs.getDouble(7));
            obj.setManagerApproval(rs.getString(8));
            obj.setProductCost(rs.getDouble(9));
            obj.setGstTax(rs.getDouble(10));
            obj.setDeliveryCost(rs.getDouble(11));
            obj.setTotalOrderCost(rs.getDouble(12));
            obj.setFinalStatusOfOrder(rs.getString(13));
            // adding ProductOrders object into arraylist
            pdtOrder.add(obj);
        }

    } catch (Exception e) {
        e.printStackTrace();
        // throw new StudentAdmissionException("Database Value
Retrieval Failed",
    }
}

```

```

        // e.getCause());
    }

    return pdtOrder;
}

}

```

ProductOrderException.java

```

package com.cts.zepcpd.exception;

public class ProductOrderException extends Exception {

    private static final long serialVersionUID = -1105431869622052445L;

    /**
     * @param message
     * @param cause
     */
    public ProductOrderException(String message, Throwable cause) {
        super(message, cause);
    }
}

```

MainApp.java

```

package com.cts.zepcpd.main;

import com.cts.zepcpd.service.*;
import com.cts.zepcpd.skeletonvalidator.SkeletonValidator;
import com.cts.zepcpd.util.*;

public class MainApp {

    private MainApp() {

    }

    public static void main(String[] args) {
        //Don't delete this code
        //Skeletonvalidaton starts
        new SkeletonValidator();
        //Skeletonvalidation ends
        //Write your code here..
        try {
            ProductOrderService service = new ProductOrderService();

            System.out.println(service.addProductOrderDetails("inputFeed.txt"));
            System.out.println(service.searchProductOrder("R005"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```

        // List<StudentAdmission> studentAdmissionList =
        //
service.buildStudentAdmissionsList(ApplicationUtil.readFile("inputFeed.txt"));
        /*
         * for(StudentAdmission e:studentAdmissionList) {
System.out.println(e); }
        */
    }

}

```

ProductOrderService.java

```

package com.cts.zepcpd.service;

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

import com.cts.zepcpd.dao.*;
import com.cts.zepcpd.exception.ProductOrderException;
import com.cts.zepcpd.util.*;
import com.cts.zepcpd.vo.ProductOrder;

public class ProductOrderService {

    /**
     * @param productOrderRecords
     * @return List<ProductOrder>
     */
    public static List<ProductOrder> buildProductOrdersList(List<String>
productOrderRecords) {
        List<ProductOrder> productOrderList = new ArrayList<ProductOrder>();

        //Code here..
        for (String e : productOrderRecords) {
            String res[] = e.split(",");
            String orderId = res[0];
            String productCode = res[1];
            String dateOfOrder = res[2];    //DateType
            String productLevel = res[3];
            String dateOfDelivery = res[4];  //DateType
            String noOfProducts= res[5];
            String noOfKmsForDelivery = res[6];
            String managerApproval = res[7];
            ProductOrder obj = new ProductOrder();
            obj.setOrderId(orderId);
            obj.setProductCode(productCode);

            // converting String to java.util.Date

            obj.setDateOfOrder(ApplicationUtil.convertStringToDate(dateOfOrder));
            obj.setProductLevel(productLevel);

            // converting String to java.util.Date

            obj.setDateOfDelivery(ApplicationUtil.convertStringToDate(dateOfDelivery));
            obj.setNoOfProducts(Integer.parseInt(noOfProducts));
        }
    }
}

```

```

        obj.setNoOfKmsForDelivery(Double.parseDouble(noOfKmsForDelivery));
        obj.setManagerApproval(managerApproval);
        double[] productOrderCosts =
            calculateTotalOrderCost(
                Integer.parseInt(noOfProducts),
                Double.parseDouble(noOfKmsForDelivery), productLevel );

        obj.setProductCost(productOrderCosts[0]);
        obj.setGstTax(productOrderCosts[1]);
        obj.setDeliveryCost(productOrderCosts[2]);
        obj.setTotalOrderCost(productOrderCosts[3]);
        obj.setFinalStatusOfOrder("OrderSuccessfull");
        productOrderList.add(obj);
    }
    return productOrderList ;
}

public boolean addProductOrderDetails(String inputFeed) throws
ProductOrderException {
    List<ProductOrder> productOrderList = ProductOrderService
        .buildProductOrdersList(ApplicationUtil.readFile(inputFeed));
    ProductOrderDAO stdDao = new ProductOrderDAO();
    return stdDao.addProductOrderDetails(productOrderList);
//Code here..

    //TODO change this return value
}

public static double[] calculateTotalOrderCost(int noOfProducts, double
noOfKmsForDelivery, String productLevel) {
    double[] productOrderCosts = new double[4];

    if ("Level01".equals(productLevel)) {
        productOrderCosts[0] = 500;
        productOrderCosts[1] = 2;
        productOrderCosts[2] = 8;
    } else if ("Level02".equals(productLevel)) {
        productOrderCosts[0] = 600;
        productOrderCosts[1] = 3;
        productOrderCosts[2] = 10;
    } else if ("Level03".equals(productLevel)) {
        productOrderCosts[0] = 800;
        productOrderCosts[1] = 5;
        productOrderCosts[2] = 11;
    } else if ("Level04".equals(productLevel)) {
        productOrderCosts[0] = 1200;
        productOrderCosts[1] = 7;
        productOrderCosts[2] = 13;
    } else if ("Level05".equals(productLevel)) {
        productOrderCosts[0] = 1750;
        productOrderCosts[1] = 8;
        productOrderCosts[2] = 14;
    } else if ("Level06".equals(productLevel)) {
        productOrderCosts[0] = 2500;
    }
}

```

```

        productOrderCosts[1] = 9;
        productOrderCosts[2] = 14;
    }
    productOrderCosts[0] =
Math.round(productOrderCosts[0]*noOfProducts*100.0)/100.0;
    productOrderCosts[1] = Math.round((productOrderCosts[1] *
productOrderCosts[0])/100*100.0)/100.0;
    productOrderCosts[2] =
Math.round(productOrderCosts[2]*noOfKmsForDelivery*100.0)/100.0;
    productOrderCosts[3] =
productOrderCosts[0]+productOrderCosts[1]+productOrderCosts[2];

    return productOrderCosts;
}

public boolean searchProductOrder(String orderId) throws
ProductOrderException {
    boolean status = false;

    //Code here..
    ProductOrderDAO ptdDao = new ProductOrderDAO();
    List<ProductOrder> ptdOrders = ptdDao.getAllProductOrderDetails();
    for (ProductOrder e : ptdOrders ) {
        if (e.getOrderId().equals(orderId)) {
            status = true;
            System.out.println(e);
            break;
        }
        else {
            System.out.println("Order Request not Found");
        }
    }

    return status;
}
}

```

ApplicationUtil.java

```

package com.cts.zepcpd.util;

import java.io.*;
import java.text.*;
import java.util.*;

import com.cts.zepcpd.exception.ProductOrderException;

public class ApplicationUtil {

```

```

/**
 * @param fileName
 * @return List<String>
 * @throws ProductOrderException
 */
public static List<String> readFile(String fileName) throws
ProductOrderException {
    List<String> productOrderList = new ArrayList<String>();
    //Code Here
    FileReader fr = null;
    BufferedReader br = null;
    try {
        fr = new FileReader(fileName);
        br = new BufferedReader(fr);
        String line = null;
        while ((line = br.readLine()) != null) {
            String[] res = line.split(",");
            String managerApproval = res[7];
            Date dtOfOrder = convertStringToDate(res[2]);
            Date dtOfDelivery = convertStringToDate(res[4]);
            if (checkIfValidOrder(dtOfOrder,dtOfDelivery,
managerApproval)) {
                productOrderList.add(line);
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return productOrderList;
}

```

```

/**
 * @param util
 *          Date
 * @return sql Date
 */
public static java.sql.Date convertUtilToSqlDate(java.util.Date uDate) {
    java.sql.Date sDate = new java.sql.Date(uDate.getTime());

    //Code here..

    return sDate;
}

/**
 * @param inDate
 * @return Date
 */
public static Date convertStringToDate(String inDate) {
    //Code here..
    try {
        SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd",
Locale.ENGLISH);
        return format.parse(inDate);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

```

        return null;
    }

}

public static boolean checkIfValidOrder(Date dtOfOrder, Date dtOfDelivery,
String manager) {
    boolean orderValidity = false;

    //Code here..
    if ("Approved".equals(manager)&& ((dtOfDelivery.getTime() -
dtOfOrder.getTime()) / (1000 * 60 * 60 * 24)) % 365 >=7) {
        orderValidity = true;
    }
    return orderValidity;
}
}

```

DBConnectionManager.java

```

* Don't change this code
*/
package com.cts.zepcpd.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.zepcpd.exception.ProductOrderException;

public class DBConnectionManager {

    private static DBConnectionManager instance;

    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 ProductOrderException
         */
        private DBConnectionManager() throws ProductOrderException {
            loadProperties();
            try {
                Class.forName(props.getProperty(DRIVER));
                this.connection =
                    DriverManager.getConnection(props.getProperty(URL), props.getProperty(USER_NAME),
                        props.getProperty(PASSWORD));
            } catch (ClassNotFoundException ex) {
                throw new ProductOrderException("Could not find Driver
class ", ex.getCause());
            } catch (SQLException e) {
                throw new ProductOrderException("Database Connection
Creation Failed", e.getCause());
            }
        }

        /**
         * @return Connection
         */
        public Connection getConnection() {
            return connection;
        }
    }
}

```

```

    }

    /**
     * @return DBConnectionManager
     * @throws ProductOrderException
     */
    public static DBConnectionManager getInstance() throws
ProductOrderException {

    // Code here

        instance = new DBConnectionManager();

        return instance;

    }

    /**
     * @throws ProductOrderException
     */
    private void loadProperties() throws ProductOrderException {
        FileInputStream inputStream = null;
        try {
            inputStream = new FileInputStream(PROPERTY_FILE);
            props = new Properties();
            props.load(inputStream);
        } catch (FileNotFoundException e) {
            throw new ProductOrderException("Database Property File
Not Found", e.getCause());
        } catch (IOException e) {
            throw new ProductOrderException("Exception during
property file I/O", e.getCause());
        } finally {
            if (inputStream != null) {

```

```
        try {
            inputStream.close();
        } catch (IOException e) {
            throw new
ProductOrderException("Exception during property file I/O", e.getCause());
        }
    }
}
```

ProductOrder .Java

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

```
*/
```

```
package com.cts.zepcpd.vo;
```

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

```
public class ProductOrder {
```

```
    String orderId;
    String productCode;
    Date dateOfOrder;
    String productLevel;
    Date dateOfDelivery;
    int noOfProducts;
    double noOfKmsForDelivery;
    String managerApproval;
    double productCost;
    double gstTax;
    double deliveryCost;
    double totalOrderCost;
```

```
String finalStatusOfOrder;

public ProductOrder() {
    super();
}

public ProductOrder(String orderId, String productCode, Date dateOfOrder,
String productLevel, Date dateOfDelivery,
int noOfProducts, double noOfKmsForDelivery, String
managerApproval, double productCost, double gstTax,
double deliveryCost, double totalOrderCost, String
finalStatusOfOrder) {
    super();
    this.orderId = orderId;
    this.productCode = productCode;
    this.dateOfOrder = dateOfOrder;
    this.productLevel = productLevel;
    this.dateOfDelivery = dateOfDelivery;
    this.noOfProducts = noOfProducts;
    this.noOfKmsForDelivery = noOfKmsForDelivery;
    this.managerApproval = managerApproval;
    this.productCost = productCost;
    this.gstTax = gstTax;
    this.deliveryCost = deliveryCost;
    this.totalOrderCost = totalOrderCost;
    this.finalStatusOfOrder = finalStatusOfOrder;
}

public String getOrderId() {
    return orderId;
}
```

```
public void setOrderId(String orderId) {  
    this.orderId = orderId;  
}  
  
public String getProductCode() {  
    return productCode;  
}  
  
public void setProductCode(String productCode) {  
    this.productCode = productCode;  
}  
  
public Date getDateOfOrder() {  
    return dateOfOrder;  
}  
  
public void setDateOfOrder(Date dateOfOrder) {  
    this.dateOfOrder = dateOfOrder;  
}  
  
public String getProductLevel() {  
    return productLevel;  
}
```

```
public void setProductLevel(String productLevel) {  
    this.productLevel = productLevel;  
}
```

```
public Date getDateOfDelivery() {  
    return dateOfDelivery;  
}
```

```
public void setDateOfDelivery(Date dateOfDelivery) {  
    this.dateOfDelivery = dateOfDelivery;  
}
```

```
public int getNoOfProducts() {  
    return noOfProducts;  
}
```

```
public void setNoOfProducts(int noOfProducts) {  
    this.noOfProducts = noOfProducts;  
}
```

```
public double getNoOfKmsForDelivery() {  
    return noOfKmsForDelivery;  
}
```

```
public void setNoOfKmsForDelivery(double noOfKmsForDelivery) {  
    this.noOfKmsForDelivery = noOfKmsForDelivery;  
}
```

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

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

```
public double getProductCost() {  
    return productCost;  
}
```

```
public void setProductCost(double productCost) {  
    this.productCost = productCost;  
}
```

```
public double getGstTax() {  
    return gstTax;  
}
```

```
public void setGstTax(double gstTax) {
```

```
        this.gstTax = gstTax;
    }

    public double getDeliveryCost() {
        return deliveryCost;
    }

    public void setDeliveryCost(double deliveryCost) {
        this.deliveryCost = deliveryCost;
    }

    public double getTotalOrderCost() {
        return totalOrderCost;
    }

    public void setTotalOrderCost(double totalOrderCost) {
        this.totalOrderCost = totalOrderCost;
    }

    public String getFinalStatusOfOrder() {
        return finalStatusOfOrder;
    }

    public void setFinalStatusOfOrder(String finalStatusOfOrder) {
        this.finalStatusOfOrder = finalStatusOfOrder;
    }
```

```

    }

    @Override
    public String toString() {
        return "ZepcManagementOrder Details: [orderId=" + orderId + ",  

productCode=" + productCode + ", dateOfOrder="

                + dateOfOrder + ", productLevel=" + productLevel +  

", dateOfDelivery=" + dateOfDelivery + ", noOfProducts="

                + noOfProducts + ", noOfKmsForDelivery=" +  

noOfKmsForDelivery + ", managerApproval=" + managerApproval

                + ", productCost=" + productCost + ", gstTax=" +  

gstTax + ", deliveryCost=" + deliveryCost + ", totalOrderCost=" + totalOrderCost

                + ", finalStatusOfOrder=" + finalStatusOfOrder + "]";
    }
}

```

SkeletonValidator.java

```

package com.cts.zepcpd.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.zepcpd.util.DBConnectionManager");  
        validateClassName("com.cts.zepcpd.util.ApplicationUtil");  
        validateClassName("com.cts.zepcpd.service.ProductOrderService");  
        validateClassName("com.cts.zepcpd.dao.ProductOrderDAO");  
        validateClassName("com.cts.zepcpd.vo.ProductOrder");  
  
        validateClassName("com.cts.zepcpd.exception.ProductOrderException");  
  
        validateMethodSignature(  
            "addProductOrderDetails:boolean,getAllProductOrderDetails>List",  
            "com.cts.zepcpd.dao.ProductOrderDAO");  
        validateMethodSignature(  
            "buildProductOrdersList>List,addProductOrderDetails:boolean,calculateTotal  
OrderCost:double[],searchProductOrder:boolean",  
            "com.cts.zepcpd.service.ProductOrderService");  
        validateMethodSignature(  
            "readFile>List,convertUtilToSqlDate>Date,convertStringToDate>Date,checkIfV  
alidOrder:boolean",  
            "com.cts.zepcpd.util.ApplicationUtil");  
        validateMethodSignature(  
            "getConnection:Connection,getInstance:DBConnectionManager",  
            "com.cts.zepcpd.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.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");  
  
}  
  
}
```

```

using System;
using System.Collections.Generic;

namespace Appointment_Booking_Application
{
    public class Program
    {
        public static void Main()
        {
            bool isValidAppointmentDate;
            AppointmentDateVerification a1 = new AppointmentDateVerification();

            try
            {
                PatientDetail patientDetail = new PatientDetail();

                string doctorname = string.Empty;

                Console.Write("Patient Name: ");
                patientDetail.PatientName = Console.ReadLine();
                Console.Write("Patient Age: ");
                patientDetail.PatientAge = Convert.ToInt32(Console.ReadLine());

                var departments = new PatientDetail().GetDepartments();
                Console.WriteLine("\nDepartments List\n");
                foreach (var item in departments)
                {
                    Console.WriteLine(item);
                }

                Console.Write("\nChoose the department number from the above list (1-5) : ");
                int option = Convert.ToInt32(Console.ReadLine());
                string department = departments[option - 1].Substring(2);
                bool isValidDoctor = true;
                do
                {

                    List<string> doctors = new PatientDetail().GetDoctors(option);

                    if (doctors.Count > 0)
                    {
                        Console.WriteLine("\nDoctors in the {0} department\n", department);
                        foreach (var item in doctors)
                        {
                            Console.WriteLine(item);
                        }

                        Console.Write("\nDoctor Name : ");
                        doctorname = Console.ReadLine();

                        if (!doctors.Contains(doctorname))
                        {
                            isValidDoctor = false;
                            Console.WriteLine("{0} not found in our list", doctorname);
                        }
                    }
                }
            }
        }
    }
}

```

```

        }
        else
        {
            isValidDoctor = true;
        }

    }

} while (!isValidDoctor);

do
{
    Console.Write("\nAppointment Request Date (MM/dd/yyyy) : ");
    DateTime appointmentRequestDate = Convert.ToDateTime(Console.ReadLine());

    //Validate the appointmentRequestDate and print appropriate message
    string s = new
AppointmentDateVerification().CheckAppointmentRequestDate(appointmentRequestDate);
    if (s == "Appointment Confirmed!")
    {
        Console.WriteLine(s);
        Random rd = new Random();
        int p_id = rd.Next(int.MaxValue, int.MinValue);
        Console.WriteLine("Patient Id - " + p_id);
        Console.WriteLine("Please Contact " + doctorname + " on " + appointmentRequestDate);
        isValidAppointmentDate = true;
    }
    else
    {
        Console.WriteLine(s);
        isValidAppointmentDate = false;
    }

} while (!isValidAppointmentDate);

}
catch (Exception e)
{
    Console.WriteLine(e.Message);
}
}

}

public class PatientDetail
{
    public string PatientName;
    public int PatientAge;
    public List<string> GetDepartments()
    {
        List<string> list = new List<string>();

        list.Add("ENT");
        list.Add("Gynecology");
        list.Add("Cardiology");
    }
}

```

```

        list.Add("Neurology");
        list.Add("Nephrology");
        return list;
    }
    public List<string> GetDoctors(int option)
    {
        List<string> l1 = new List<string>();
        switch (option)
        {
            case 1:
                l1.Add("Dr. Murugadoss");
                l1.Add("Dr. Kalaivani");
                break;
            case 2:
                l1.Add("Dr. Abirami");
                l1.Add("Dr. Lakshmi");
                l1.Add("Dr. Revathi");
                break;
            case 3:
                l1.Add("Dr. Amudhan");
                l1.Add("Dr. Gunaseelan");
                l1.Add("Dr. Agarwal");
                break;
            case 4:
                l1.Add("Dr. Natarajan");
                l1.Add("Dr. Nanda");
                l1.Add("Dr. Keerthi");
                break;
            case 5:
                l1.Add("Dr. Ashirvatham");
                l1.Add("Dr. Cherian");
                l1.Add("Dr. Ram");
                break;
        }
        return l1;
    }

    public class AppointmentDateVerification
    {
        public string CheckAppointmentRequestDate(DateTime appointmentRequestDate)
        {
            DateTime dt = DateTime.Today;
            DayOfWeek th = appointmentRequestDate.DayOfWeek;
            if (appointmentRequestDate > dt)
            {
                if (appointmentRequestDate.Year == dt.Year)
                {
                    if (th != DayOfWeek.Monday)
                    {
                        return "Appointment Confirmed!";
                    }
                    else
                    {
                        return "Sorry!!! Appointment cannot be given on Monday!";
                    }
                }
            }
        }
    }
}

```

```

        {
            return "Appointment Rejected, You can book appointment only for the current year!";
        }
    }
else
{
    return "Appointment Rejected, Date must be a future date!";
}
}

```

Appointment booking:

program.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AirlineTicketing // Do not change the namespace name
{
    public class Program // Do not change the class name
    {
        static void Main(string[] args) // Do not change the method name
        {
            //Implement the code here
            int nt;
            string td;
            int ns;
            string stbs;
            Console.WriteLine("Enter the number of tickets");
            nt = int.Parse(Console.ReadLine());
            List<Ticket>ticketList = new List<Ticket>();
            for(int i=1; i<=nt;i++)
            {
                td = Console.ReadLine();
                string[] details = td.Split(':');
                string name = details[0];
                string seatType = details[1];
                string seatId = details[2];

                if(TicketingDepartment.ValidateSeatId(seatId))
                {

```

```

        Ticket ticket = new Ticket(name,seatType,seatId);
        ticketList.Add(ticket);
    }
}
Console.WriteLine("Enter the number of seats to be searched");
ns = int.Parse(Console.ReadLine());
TicketingDepartment tdp;
for( int i=1;i<= ns; i++)
{
    Console.WriteLine("Enter seat type");
    stbs = Console.ReadLine();
    tdp = new TicketingDepartment(stbs,ticketList);
    Console.WriteLine(stbs+":"+tdp.SearchBySeatType(stbs));
}

}
}
}

```

ticket.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AirlineTicketing // Do not change the namespace name
{
    public class Ticket // Do not change the class name
    {
        //Implement the code here
        private string passangerName;
        private string seatType;
        private string seatId;

        public string PassangerName
        {
            get{ return passangerName; }
            set{ passangerName = value; }
        }

        public string SeatType
        {
            get{ return seatType; }
            set { seatType = value; }
        }
    }
}

```

```

    }

    public string SeatId
    {
        get{ return seatId;}
        set{ seatId = value;}
    }

    public Ticket(){}
    public Ticket( string passangerName,string seatType, string seatId)
    {
        this.passangerName = passangerName;
        this.seatType = seatType;
        this.seatId = seatId;
    }
    public string GetSeatType()
    {
        return seatType;
    }
}
}
}
}

```

TicketingDepartment.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AirlineTicketing // Do not change the namespace name
{
    public class TicketingDepartment // Do not change the class name
    {
        //Implement the code here
        private string seatType;
        private List<Ticket>ticketList;

        public string SeatType
        {
            get{ return seatType;}
            set { seatType =value;}
        }

        public List<Ticket>TicketList
        {

```

```

        get{ return ticketList;}
        set{ ticketList = value;}
    }

    public TicketingDepartment(){}
}

public TicketingDepartment(string seatType, List<Ticket>ticketList)
{
    this.seatType= seatType;
    this.ticketList = ticketList;
}

public static bool ValidateSeatId(string seatId)
{
    try
    {
        if(seatId.Length !=3)
        {
            throw new SeatIdInvalidException("Invalid Seat Id");
        }
        char f_letter = seatId[0];
        if( f_letter !='A' && f_letter != 'B' && f_letter !='C' && f_letter !='D')
            throw new SeatIdInvalidException("Invalid Seat Id");
        char sL = seatId[1];
        char tL = seatId[2];
        if(!char.IsDigit(sL) || !char.IsDigit(tL))
            throw new SeatIdInvalidException("Invalid Seat Id");
        string number ="";
        if(sL =='0')
            number = seatId.Substring(2);
        else
            number = seatId.Substring(1);
        int num = int.Parse(number);
        if(!(num>=1 && num<=45))
            throw new SeatIdInvalidException("Invalid Seat Id");
        return true;
    }
    catch(Exception e)
    {
        return false;
    }
}

public int SearchBySeatType(string seatType)
{
    int count = 0;
    foreach(Ticket ticket in ticketList)
    {

```

```
        if( ticket.GetSeatType() == seatType)
            count++;
    }
    return count;
}
}
```

SeatIdInvalidException.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace AirlineTicketing // Do not change the namespace name
{
    public class SeatIdInvalidException:Exception // Do not change the class name
    {
        //Implement the code here
        public SeatIdInvalidException(string message): base(message)
        {}
    }
}
```

1.Red code Technology

Casual Employee:

```
public class CasualEmployee extends Employee{  
  
    private int supplementaryHours;  
    private double foodAllowance;  
  
    public int getSupplementaryHours() {  
        return supplementaryHours;  
    }  
    public void setSupplementaryHours(int supplementaryHours) {  
        this.supplementaryHours = supplementaryHours;  
    }  
    public double getFoodAllowance() {  
        return foodAllowance;  
    }  
    public void setFoodAllowance(double foodAllowance) {  
        this.foodAllowance = foodAllowance;  
    }  
    public CasualEmployee(String EmployeeId, String EmployeeName, int yearsOfExperience,  
    String gender, double salary, int supplementaryHours, double foodAllowance)  
    {  
        super(EmployeeId, EmployeeName, yearsOfExperience, gender, salary);  
        this.supplementaryHours = supplementaryHours;  
        this.foodAllowance = foodAllowance;  
    }  
  
    public double calculateIncrementedSalary(int incrementPercentage)  
    {
```

```
        double total =(supplementaryHours*1000)+foodAllowance+this.salary;
        double incsalary=total+(total*incrementPercentage/100);
        return incsalary;
    }

}
```

Employee:

```
public abstract class Employee {

    protected String Employeeld;
    protected String EmployeeName;
    protected int yearsOfExperience;
    protected String gender;
    protected double salary;
    public abstract double calculateIncrementedSalary(int incrementPercentage);

    public String getEmployeeld() {
        return Employeeld;
    }
    public void setEmployeeld(String employeeld) {
        this.Employeeld = employeeld;
    }
    public String getEmployeeName() {
        return EmployeeName;
    }
    public void setEmployeeName(String employeeName) {
        this.EmployeeName = employeeName;
    }
    public int getYearsOfExperience() {
```

```

        return yearsOfExperience;
    }

    public void setYearsOfExperience(int yearsOfExperience) {
        this.yearsOfExperience = yearsOfExperience;
    }

    public String getGender() {
        return gender;
    }

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

    public double getSalary() {
        return salary;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }

    public Employee(String employeeId, String employeeName, int yearsOfExperience, String
gender, double salary) {
        super();
        this.EmployeeId = employeeId;
        this.EmployeeName = employeeName;
        this.yearsOfExperience = yearsOfExperience;
        this.gender = gender;
        this.salary=salary;
    }
}

```

Permanent Employee:

```
public class PermanentEmployee extends Employee{
```

```
private double medicalAllowance;  
private double VehicleAllowance;  
  
public double getMedicalAllowance() {  
    return medicalAllowance;  
}  
  
public void setMedicalAllowance(double medicalAllowance) {  
    this.medicalAllowance = medicalAllowance;  
}  
  
public double getVehicleAllowance() {  
    return VehicleAllowance;  
}  
  
public void setVehicleAllowance(double vehicleAllowance) {  
    VehicleAllowance = vehicleAllowance;  
}  
  
public PermanentEmployee(String EmployeeId, String EmployeeName, int  
yearsOfExperience, String gender, double salary, double medicalAllowance, double  
vehicleAllowance)  
{  
    super(EmployeeId, EmployeeName, yearsOfExperience, gender, salary);  
    this.medicalAllowance=medicalAllowance;  
    this.VehicleAllowance=vehicleAllowance;  
}  
  
public double calculateIncrementedSalary(int incrementPercentage)
```

```
{  
    double total=medicalAllowance + VehicleAllowance+this.salary;  
    double incsalary=total+(total*incrementPercentage/100);  
    return incsalary;  
}  
}
```

Trainee Employees:

```
public class TraineeEmployees extends Employee{  
  
    private int supplementaryTrainingHours;  
    private int scorePoints;  
  
    public int getSupplementaryTrainingHours() {  
        return supplementaryTrainingHours;  
    }  
    public void setSupplementaryTrainingHours(int supplementaryTrainingHours) {  
        this.supplementaryTrainingHours = supplementaryTrainingHours;  
    }  
    public int getScorePoints() {  
        return scorePoints;  
    }  
    public void setScorePoints(int scorePoints) {  
        this.scorePoints = scorePoints;  
    }  
  
    public TraineeEmployees(String Employeeld, String EmployeeName, int yearsOfExperience,  
    String gender, double salary, int supplementaryTrainingHours, int scorePoints)  
    {  
        super(Employeeld, EmployeeName, yearsOfExperience, gender, salary);  
    }
```

```
this.supplementaryTrainingHours=supplementaryTrainingHours;  
this.scorePoints=scorePoints;  
}  
  
public double calculateIncrementedSalary(int incrementPercentage){  
double total=(supplementaryTrainingHours*500)+(scorePoints*50)+this.salary;  
double incsalary=total+(total*incrementPercentage/100);  
return incsalary;  
}  
  
}
```

User Interface:

```
import java.util.Scanner;  
public class UserInterface {  
  
public static void main(String[] args){  
  
Scanner sc=new Scanner(System.in);  
System.out.println("Enter Employee Id");  
String EmployeeId = sc.next();  
System.out.println("Enter Employee name");  
String EmployeeName = sc.next();  
System.out.println("Enter Experience in years");  
int yearsOfExperience = sc.nextInt();  
System.out.println("Enter Gender");  
String gender = sc.next();  
System.out.println("Enter Salary");  
double salary=sc.nextDouble();
```

```
double incSalary=0;

if(yearsOfExperience>=1 && yearsOfExperience <= 5)

{

System.out.println("Enter Supplementary Training Hours");

int supplementaryTrainingHours = sc.nextInt();

System.out.println("Enter Score Points");

int scorePoints = sc.nextInt();

TraineeEmployees te=new TraineeEmployees(EmployeeId, EmployeeName, yearsOfExperience, gender, salary, supplementaryTrainingHours, scorePoints);

incSalary=te.calculateIncrementedSalary(5);

System.out.println("Incremented Salary is "+incSalary);

}

else if(yearsOfExperience>=6 && yearsOfExperience <=10)

{

System.out.println("Enter Supplementary Hours");

int supplementaryHours = sc.nextInt();

System.out.println("Enter Food Allowance");

double foodAllowance = sc.nextDouble();

CasualEmployee ce=new CasualEmployee(EmployeeId, EmployeeName, yearsOfExperience, gender, salary, supplementaryHours, foodAllowance);

incSalary = ce.calculateIncrementedSalary(12);

System.out.println("Incremented Salary is "+incSalary);

}

else if(yearsOfExperience>=10 && yearsOfExperience <=25)

{

System.out.println("Enter Medical Allowance");

double medicalAllowance = sc.nextDouble();

System.out.println("Enter Vehicle Allowance");

double vehicleAllowance = sc.nextDouble();
```

```

PermanentEmployee pe = new PermanentEmployee(EmployeeId, EmployeeName,
yearsOfExperience, gender, salary, medicalAllowance, vehicleAllowance);

incSalary=pe.calculateIncrementedSalary(12);

System.out.println("Incremented Salary is "+incSalary);

}

else

System.out.println("Provide valid Years of Experience");

}

}

}

```

2.Dominion Cinemas

Book 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.emaiId=emaiId;
}

public void setmovieName( String movieName){
this.movieName=movieName;
}

public String getticketId(){
return ticketId;
}

public String getcustomerName(){
return customerName;
}

public String getemaiId(){
return emaiId;
}

public String getmovieName(){
return movieName;
}

public long getmobileNumber(){
return mobileNumber;
}

public BookAMovieTicket(String ticketId,String customerName,long mobileNumber,String emaiId,String movieName)
{
this.ticketId=ticketId;
this.customerName=customerName;
this.mobileNumber=mobileNumber;
this.emaiId=emaiId;
this.movieName=movieName;
}
```

```
}
```

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 calculateTicketCost(int numberOfTickets,String ACFacility){  
            double amount;  
            if(ACFacility.equals("yes")){  
                amount=500*numberOfTickets;  
            }  
            else{  
                amount=350*numberOfTickets;  
            }  
        }  
    }  
}
```

```
}

return amount;

}

}
```

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 calculateTicketCost(int numberOfTickets, String ACFacility){

double amount;

if(ACFacility.equals("yes")){

```

```
amount=750*numberOfTickets;  
}  
  
else{  
amount=600*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 calculateTicketCost(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");  
        String tid=sc.next();  
        System.out.println("Enter Customer Name");  
        String cnm=sc.next();  
        System.out.println("Enter Mobile Number");  
        long mno=sc.nextLong();  
        System.out.println("Enter Email id");  
        String email=sc.next();  
        System.out.println("Enter Movie Name");  
        String mnm=sc.next();  
        System.out.println("Enter number of tickets");  
        int tno=sc.nextInt();
```

```
System.out.println("Do you want AC or not");

String choice =sc.next();

if(tid.contains("PLATINUM")){
    PlatinumTicket PT=new PlatinumTicket(tid,cnm,mno,email,mnm);
    boolean b1=PT.validateTicketId();
    if(b1==true){
        double cost =PT.calculateTicketCost(tno, choice);
        System.out.println("Ticket cost is "+ cost);
    }
    else if(b1==false){
        System.out.println("Provide valid Ticket Id");
        System.exit(0);
    }
}

else if(tid.contains("GOLD")){
    GoldTicket GT=new GoldTicket(tid,cnm,mno,email,mnm);
    boolean b2=GT.validateTicketId();
    if(b2==true){
        double cost=GT.calculateTicketCost(tno, choice);
        System.out.println("Ticket cost is "+cost);
    }
    else if (b2==false){
        System.out.println("Provide valid Ticket Id");
        System.exit(0);
    }
}

else if(tid.contains("SILVER")){
    SilverTicket ST=new SilverTicket(tid,cnm,mno,email,mnm);
    boolean b3=ST.validateTicketId();
```

```

if(b3==true){

double cost=ST.calculateTicketCost(tno, choice);

System.out.println("Ticket cost is "+cost);

}

else if(b3==false){

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

System.exit(0);

}

}

}

}

}

```

3.Little Innovators

Main:

```

import java.util.*;

public class Main {

    public static void main(String args[])

    {

        System.out.println("Enter the String: ");

        Scanner sc=new Scanner(System.in);

        String st=sc.nextLine();

        String op=st;

        st=st.replaceAll(" ","");

        boolean d=st.matches("[a-zA-Z]+");

        if(!d)

        {

            System.out.println("Invalid Slogan");

        }

    }

}

```

```

        else
        {
            char a[]=st.toCharArray();
            char b[]=new char[100];
            b[0]='0';
            int same=0,i=a.length,j=0,l=0;
            while(j<i)

            {
                int count=0;
                for(int k=0;k<i;k++)
                {
                    if(a[j]==a[k])
                    {
                        count++;
                    }
                }
                if(count==1)
                {
                    l++;
                    b[l]=a[j];
                    j++;
                }
            }
            else
            {
                j++;
            }
        }

        if(l==(i-l))
            System.out.println("All the guidelines
are satisfied for "+op);
    }
}

```

```

        else
            System.out.println(op+" does not satisfy
the guideline");
    }
}
}

```

4.Kidsor Home appliances

Air Conditioner:

```

public class AirConditioner extends ElectronicProducts {
    private String airConditionerType;
    private double capacity;
    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 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;  
}  
  
public double calculateProductPrice(){  
    double price = 0;  
  
    if(airConditionerType.equalsIgnoreCase("Residential")){  
        if (capacity == 2.5){  
            price = 32000;  
        }  
        else if(capacity == 4){  
            price = 40000;  
        }  
        else if(capacity == 5.5){  
            price = 47000;  
        }  
    }  
  
    else if(airConditionerType.equalsIgnoreCase("Commercial")){  
        if (capacity == 2.5){  
            price = 40000;  
        }  
        else if(capacity == 4){  
            price = 55000;  
        }  
        else if(capacity == 5.5){  
            price = 67000;  
        }  
    }  
  
    else if(airConditionerType.equalsIgnoreCase("Industrial")){  
        if (capacity == 2.5){  
            price = 47000;
```

```
}

else if(capacity == 4){

price = 60000;

}

else if(capacity == 5.5){

price = 70000;

}

}

return price;

}

}
```

Electronic Products:

```
public class ElectronicProducts {

protected String productId;

protected String productName;

protected String batchId;

protected String dispatchDate;

protected int 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;

}

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;
}
```

LED TV:

```
public class LEDTV extends ElectronicProducts {  
    private int size;  
    private String quality;  
    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 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;  
    }  
    public double calculateProductPrice(){  
        double price = 0;  
        if(quality.equalsIgnoreCase("Low")){  
            price = size * 850;  
        }  
        else if(quality.equalsIgnoreCase("Medium")){  
            price = size * 1250;  
        }  
    }  
}
```

```
}

else if(quality.equalsIgnoreCase("High")){
    price = size * 1550;
}

return price;
}

}
```

Microwave Oven:

```
public class MicrowaveOven extends ElectronicProducts{

    private int quantity;

    private String quality;

    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 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;
    }
}
```

```
}

public double calculateProductPrice(){

double price = 0;

if(quality.equalsIgnoreCase("Low")){
price = quantity * 1250;

}

else if(quality.equalsIgnoreCase("Medium")){
price = quantity * 1750;

}

else if(quality.equalsIgnoreCase("High")){
price = quantity * 2000;

}

return price;
}

}
```

User Interface:

```
import java.util.Scanner;

public class UserInterface {

public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
System.out.println("Enter Product Id");
String productId = sc.next();
System.out.println("Enter Product Name");
String productName = sc.next();
System.out.println("Enter Batch Id");
String batchId = sc.next();
System.out.println("Enter Dispatch Date");
```

```
String dispatchDate = sc.next();
System.out.println("Enter Warranty Years");
int warrantyYears = sc.nextInt();
double price;
String quality;
switch(productName){
case "AirConditioner":
System.out.println("Enter type of Air Conditioner");
String type = sc.next();
System.out.println("Enter quantity");
double capacity = sc.nextDouble();
AirConditioner ac = new AirConditioner(productId, productName, batchId,
dispatchDate, warrantyYears, type, capacity);
price = ac.calculateProductPrice();
System.out.printf("Price of the product is %.2f", price);
break;
case "LEDTV":
System.out.println("Enter size in inches");
int size = sc.nextInt();
System.out.println("Enter quality");
quality = sc.next();
LEDTV l = new LEDTV(productId, productName, batchId, dispatchDate,
warrantyYears, size, quality);
price = l.calculateProductPrice();
System.out.printf("Price of the product is %.2f", price);
break;
case "MicrowaveOven":
System.out.println("Enter quantity");
int quantity = sc.nextInt();
```

```

System.out.println("Enter quality");
quality = sc.next();

MicrowaveOven m = new MicrowaveOven(productId, productName, batchId,
dispatchDate, warrantyYears, quantity, quality);

price = m.calculateProductPrice();

System.out.printf("Price of the product is %.2f", price);

break;

default:

System.out.println("Provide a valid Product name");

System.exit(0);

}

}

}

```

5. Reverse a word

```

import java.util.*;

class HelloWorld {

    public static void main(String[] args) {

        String[] words ;
        Scanner myObj = new Scanner(System.in);

        String sentence= myObj.nextLine();
        words=sentence.split(" ");
        if(words.length<3)
            System.out.println("Invalid Sentence");
        else{

```

```
String a=words[0].substring(0,1);
String b=words[1].substring(0,1);
String c=words[2].substring(0,1);
if(a.equalsIgnoreCase(b)&&b.equalsIgnoreCase(c))
{
    StringBuilder input1 = new StringBuilder();

    input1.append(words[words.length-1]);
    // reverse StringBuilder input1

    input1=input1.reverse();
    input1.append(words[0]);

    System.out.println(input1);
}
else {

    StringBuilder input1 = new StringBuilder();

    input1.append(words[0]);
    // reverse StringBuilder input1

    input1=input1.reverse();
    input1.append(words[words.length-1]);

    System.out.println(input1);
}

}
```

}

}

LITTLE INNOVATORS

```
import java.util.Scanner;
public class Main {

    public static void main(String args[])
    {
        System.out.println("Enter the String: ");
        Scanner sc=new Scanner(System.in);
        String st=sc.nextLine();
        String op=st;
        st=st.replaceAll(" ","");
        boolean d=st.matches("[a-zA-Z]+");
        if(!d)
        {
            System.out.println("Invalid Slogan");
        }
        else
        {
            char a[]=st.toCharArray();
            char b[]=new char[100];
            b[0]='0';
            int same=0,i=a.length,j=0,l=0;
            while(j<i)
            {
                int count=0;
                for(int k=0;k<i;k++)
                {
                    if(a[j]==a[k])
                    {
                        count++;
                    }
                }
                if(count==1)
                {
                    l++;
                    b[l]=a[j];
                    j++;
                }
                else
                {
                    j++;
                }
            }
            if(l==(i-l))
                System.out.println("All the guidelines are satisfied for "+op);
            else
                System.out.println(op+" does not satisfy the guideline");
        }
    }
}
```

GD HOSPITALS

INPATIENT CLASS:

```
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 calculateTotalBilll(int no,double medi)  
    {  
        return ((roomRent*no)+medi);  
    }  
}
```

OUTPATIENT CLASS:

```
public class OutPatient extends Patient {  
  
    OutPatient()  
    {  
        super();  
    }  
    private double consultingFee;  
    public double getcon()  
    {  
        return consultingFee;  
    }  
    public void setcon(double con)  
    {  
        consultingFee=con;  
    }  
    public double calculateTotalBilll(int scan,double medi)  
    {  
        return (consultingFee+scan+medi);  
    }  
}
```

PATIENT CLASS:

```
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 getpamob(){  
        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 setpamob(long mob){  
        mobileNumber=mob;  
    }  
}
```

MAIN CLASS:

```
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();
            o2.setrent(rent);
            System.out.println("Amount to be paid "+o2.calculateTotalBilll(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();
            o1.setcon(con);
            System.out.println("Amount to be paid "+o1.calculateTotalBilll(scan,med));
        }
    }
}
```

SILVER HEALTH PLAN INSURANCE

FAMILY INSURANCE POLICY CLASS:

```
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;
        }
    }
```

INDIVIDUAL INSURANCE POLICY CLASS:

```
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;
        }
    }
```

SENIOR CITIZEN POLICY CLASS:

```
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;
        }
    }
```

INSURANCE POLICIES CLASS:

```
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;  
    }  
}
```

USER INTERFACE CLASS:

```
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("EmailId:"+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");
}
```

DOMINION CINEMAS

BOOK A MOVIE TICKET CLASS:

```
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;  
}  
}
```

GOLD TICKET CLASS:

```
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 calculateTicketCost(int numberOfTickets, String ACFacility){  
            double amount;  
            if(ACFacility.equals("yes")){  
                amount=500*numberOfTickets;  
            }  
            else{  
                amount=350*numberOfTickets;  
            }  
            return amount;  
        }  
    }  
}
```

PLATINUM TICKET CLASS:

```
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 calculateTicketCost(int numberOfTickets, String ACFacility){  
            double amount;  
            if(ACFacility.equals("yes")){  
                amount=750*numberOfTickets;  
            }  
            else{  
                amount=600*numberOfTickets;  
            }  
            return amount;  
        }  
    }  
}
```

SILVER TICKET CLASS:

```
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 calculateTicketCost(int numberOfTickets, String ACFacility){
            double amount;
            if(ACFacility.equals("yes")){
                amount=250*numberOfTickets;
            }
            else{
                amount=100*numberOfTickets;
            }
            return amount;
        }
    }
}
```

USER INTERFACE CLASS:

```
import java.util.Scanner;
public class UserInterface {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter Ticket Id");
        String tid=sc.next();
        System.out.println("Enter Customer Name");
        String cnm=sc.next();
        System.out.println("Enter Mobile Number");
        long mno=sc.nextLong();
        System.out.println("Enter Email id");
        String email=sc.next();
        System.out.println("Enter Movie Name");
        String mnmm=sc.next();
        System.out.println("Enter number of tickets");
        int tno=sc.nextInt();
        System.out.println("Do you want AC or not");
        String choice =sc.next();
        if(tid.contains("PLATINUM")){
            PlatinumTicket PT=new PlatinumTicket(tid,cnm,mno,email,mnmm);
            boolean b1=PT.validateTicketId();
            if(b1==true){
                double cost =PT.calculateTicketCost(tno, choice);
                System.out.println("Ticket cost is "+ cost);
            }
            else if(b1==false){
                System.out.println("Provide valid Ticket Id");
                System.exit(0);
            }
        }
        else if(tid.contains("GOLD")){
            GoldTicket GT=new GoldTicket(tid,cnm,mno,email,mnmm);
            boolean b2=GT.validateTicketId();
            if(b2==true){
                double cost=GT.calculateTicketCost(tno, choice);
                System.out.println("Ticket cost is "+cost);
            }
            else if (b2==false){
                System.out.println("Provide valid Ticket Id");
                System.exit(0);
            }
        }
        else if(tid.contains("SILVER")){
            SilverTicket ST=new SilverTicket(tid,cnm,mno,email,mnmm);
            boolean b3=ST.validateTicketId();
            if(b3==true){
                double cost=ST.calculateTicketCost(tno, choice);
                System.out.println("Ticket cost is "+cost);
            }
            else if(b3==false){
                System.out.println("Provide valid Ticket Id");
                System.exit(0);
            }
        }
    }
}
```

CODE RED TECHNOLOGIES

CASUAL EMPLOYEE CLASS:

```
public class CasualEmployee extends Employee{  
    private int supplementaryHours;  
    private double foodAllowance;  
    public int getSupplementaryHours() {  
        return supplementaryHours;  
    }  
    public void setSupplementaryHours(int supplementaryHours) {  
        this.supplementaryHours = supplementaryHours;  
    }  
    public double getFoodAllowance() {  
        return foodAllowance;  
    }  
    public void setFoodAllowance(double foodAllowance) {  
        this.foodAllowance = foodAllowance;  
    }  
    public CasualEmployee(String EmployeeId, String EmployeeName, int yearsOfExperience,  
String gender, double salary, int supplementaryHours, double foodAllowance)  
{  
    super(EmployeeId, EmployeeName, yearsOfExperience, gender, salary);  
    this.supplementaryHours = supplementaryHours;  
    this.foodAllowance = foodAllowance;  
}  
    public double calculateIncrementedSalary(int incrementPercentage)  
{  
    double total = (supplementaryHours * 1000) + foodAllowance + this.salary;  
    double incsalary = total + (total * incrementPercentage / 100);  
    return incsalary;  
}  
}
```

PERMANENT EMPLOYEE CLASS:

```
public class PermanentEmployee extends Employee{  
    private double medicalAllowance;  
    private double VehicleAllowance;  
    public double getMedicalAllowance() {  
        return medicalAllowance;  
    }  
    public void setMedicalAllowance(double medicalAllowance) {  
        this.medicalAllowance = medicalAllowance;  
    }  
    public double getVehicleAllowance() {  
        return VehicleAllowance;  
    }  
    public void setVehicleAllowance(double vehicleAllowance) {  
        VehicleAllowance = vehicleAllowance;  
    }  
    public PermanentEmployee(String EmployeeId, String EmployeeName, int  
yearsOfExperience, String gender, double salary, double medicalAllowance, double  
vehicleAllowance)  
{  
    super(EmployeeId, EmployeeName, yearsOfExperience, gender, salary);  
    this.medicalAllowance = medicalAllowance;  
    this.VehicleAllowance = vehicleAllowance;  
}  
    public double calculateIncrementedSalary(int incrementPercentage)  
{  
    double total = medicalAllowance + VehicleAllowance + this.salary;  
    double incsalary = total + (total * incrementPercentage / 100);  
    return incsalary;  
}
```

TRAINEE EMPLOYEES CLASS:

```
public class TraineeEmployees extends Employee{
private int supplementaryTrainingHours;
private int scorePoints;
public int getSupplementaryTrainingHours() {
return supplementaryTrainingHours;
}
public void setSupplementaryTrainingHours(int supplementaryTrainingHours) {
this.supplementaryTrainingHours = supplementaryTrainingHours;
}
public int getScorePoints() {
return scorePoints;
}
public void setScorePoints(int scorePoints) {
this.scorePoints = scorePoints;
}
public TraineeEmployees(String EmployeeId, String EmployeeName, int yearsOfExperience,
String gender, double salary, int supplementaryTrainingHours, int scorePoints)
{
super(EmployeeId, EmployeeName, yearsOfExperience, gender, salary);
this.supplementaryTrainingHours=supplementaryTrainingHours;
this.scorePoints=scorePoints;
}
public double calculateIncrementedSalary(int incrementPercentage){
double total=(supplementaryTrainingHours*500)+(scorePoints*50)+this.salary;
double incsalary=total+(total*incrementPercentage/100);
return incsalary;
}
}
```

EMPLOYEE CLASS:

```
public abstract class Employee {  
    protected String EmployeeId;  
    protected String EmployeeName;  
    protected int yearsOfExperience;  
    protected String gender;  
    protected double salary;  
    public abstract double calculateIncrementedSalary(int incrementPercentage);  
    public String getEmployeeId() {  
        return EmployeeId;  
    }  
    public void setEmployeeId(String employeeId) {  
        this.EmployeeId = employeeId;  
    }  
    public String getEmployeeName() {  
        return EmployeeName;  
    }  
    public void setEmployeeName(String employeeName) {  
        this.EmployeeName = employeeName;  
    }  
    public int getYearsOfExperience() {  
        return yearsOfExperience;  
    }  
    public void setYearsOfExperience(int yearsOfExperience) {  
        this.yearsOfExperience = yearsOfExperience;  
    }  
    public String getGender() {  
        return gender;  
    }  
    public void setGender(String gender) {  
        this.gender = gender;  
    }  
    public double getSalary() {  
        return salary;  
    }  
    public void setSalary(double salary) {  
        this.salary = salary;  
    }  
    public Employee(String employeeId, String employeeName, int yearsOfExperience, String  
    gender, double salary) {  
        super();  
        this.EmployeeId = employeeId;  
        this.EmployeeName = employeeName;  
        this.yearsOfExperience = yearsOfExperience;  
        this.gender = gender;  
        this.salary=salary;  
    }  
}
```

USER INTERFACE CLASS:

```
import java.util.Scanner;
public class UserInterface {
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter Employee Id");
        String EmployeeId = sc.next();
        System.out.println("Enter Employee name");
        String EmployeeName = sc.next();
        System.out.println("Enter Experience in years");
        int yearsOfExperience = sc.nextInt();
        System.out.println("Enter Gender");
        String gender = sc.next();
        System.out.println("Enter Salary");
        double salary=sc.nextDouble();
        double incSalary=0;
        if(yearsOfExperience>=1 && yearsOfExperience <= 5)
        {
            System.out.println("Enter Supplementary Training Hours");
            int supplementaryTrainingHours = sc.nextInt();
            System.out.println("Enter Score Points");
            int scorePoints = sc.nextInt();
            TraineeEmployees te=new TraineeEmployees(EmployeeId, EmployeeName,
                yearsOfExperience, gender, salary, supplementaryTrainingHours,
scorePoints);
            incSalary=te.calculateIncrementedSalary(5);
            System.out.println("Incremented Salary is "+incSalary);
        }
        else if(yearsOfExperience>=6 && yearsOfExperience <=10)
        {
            System.out.println("Enter Supplementary Hours");
            int supplementaryHours = sc.nextInt();
            System.out.println("Enter Food Allowance");
            double foodAllowance = sc.nextDouble();
            CasualEmployee ce=new CasualEmployee(EmployeeId, EmployeeName, yearsOfExperience,
                gender, salary, supplementaryHours, foodAllowance);
            incSalary = ce.calculateIncrementedSalary(12);
            System.out.println("Incremented Salary is "+incSalary);
        }
        else if(yearsOfExperience>=10 && yearsOfExperience <=25)
        {
            System.out.println("Enter Medical Allowance");
            double medicalAllowance = sc.nextDouble();
            System.out.println("Enter Vehicle Allowance");
            double vehicleAllowance = sc.nextDouble();
            PermanentEmployee pe = new PermanentEmployee(EmployeeId, EmployeeName,
                yearsOfExperience, gender, salary, medicalAllowance, vehicleAllowance);
            incSalary=pe.calculateIncrementedSalary(12);
            System.out.println("Incremented Salary is "+incSalary);
        }
        else
            System.out.println("Provide valid Years of Experience");
    }
}
```

KIDSOR HOME APPLIANCES

AIR CONDITIONER CLASS:

```
public class AirConditioner extends ElectronicProducts {
private String airConditionerType;
private double capacity;
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 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;
}
public double calculateProductPrice(){
double price = 0;
if(airConditionerType.equalsIgnoreCase("Residential")){
    if (capacity == 2.5){
        price = 32000;
    }
    else if(capacity == 4){
        price = 40000;
    }
    else if(capacity == 5.5){
        price = 47000;
    }
}
else if(airConditionerType.equalsIgnoreCase("Commercial")){
    if (capacity == 2.5){
        price = 40000;
    }
    else if(capacity == 4){
        price = 55000;
    }
    else if(capacity == 5.5){
        price = 67000;
    }
}
else if(airConditionerType.equalsIgnoreCase("Industrial")){
    if (capacity == 2.5){
        price = 47000;
    }
    else if(capacity == 4){
        price = 60000;
    }
    else if(capacity == 5.5){
        price = 70000;
    }
}
}
return price;
}
```

LEDTV CLASS:

```
public class LEDTV extends ElectronicProducts {
private int size;
private String quality;
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 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;
}
public double calculateProductPrice(){
    double price = 0;
    if(quality.equalsIgnoreCase("Low")){
        price = size * 850;
    }
    else if(quality.equalsIgnoreCase("Medium")){
        price = size * 1250;
    }
    else if(quality.equalsIgnoreCase("High")){
        price = size * 1550;
    }
    return price;
}
}
```

MICROWAVE OVEN CLASS:

```
public class MicrowaveOven extends ElectronicProducts{
private int quantity;
private String quality;
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 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;
}
public double calculateProductPrice(){
double price = 0;
if(quality.equalsIgnoreCase("Low")){
    price = quantity * 1250;
}
else if(quality.equalsIgnoreCase("Medium")){
    price = quantity * 1750;
}
else if(quality.equalsIgnoreCase("High")){
    price = quantity * 2000;
}
return price;
}
}
```

ELECTRONIC PRODUCTS CLASS:

```
public class ElectronicProducts {  
protected String productId;  
protected String productName;  
protected String batchId;  
protected String dispatchDate;  
protected int 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;  
}  
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;  
}  
}
```

USER INTERFACE CLASS:

```
import java.util.Scanner;
public class UserInterface {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter Product Id");
        String productId = sc.next();
        System.out.println("Enter Product Name");
        String productName = sc.next();
        System.out.println("Enter Batch Id");
        String batchId = sc.next();
        System.out.println("Enter Dispatch Date");
        String dispatchDate = sc.next();
        System.out.println("Enter Warranty Years");
        int warrantyYears = sc.nextInt();
        double price;
        String quality;
        switch(productName){
            case "AirConditioner":
                System.out.println("Enter type of Air Conditioner");
                String type = sc.next();
                System.out.println("Enter quantity");
                double capacity = sc.nextDouble();
                AirConditioner ac = new AirConditioner(productId, productName, batchId,
                    dispatchDate, warrantyYears, type, capacity);
                price = ac.calculateProductPrice();
                System.out.printf("Price of the product is %.2f", price);
                break;
            case "LEDTV":
                System.out.println("Enter size in inches");
                int size = sc.nextInt();
                System.out.println("Enter quality");
                quality = sc.next();
                LEDTV l = new LEDTV(productId, productName, batchId, dispatchDate,
                    warrantyYears, size, quality);
                price = l.calculateProductPrice();
                System.out.printf("Price of the product is %.2f", price);
                break;
            case "MicrowaveOven":
                System.out.println("Enter quantity");
                int quantity = sc.nextInt();
                System.out.println("Enter quality");
                quality = sc.next();
                MicrowaveOven m = new MicrowaveOven(productId, productName, batchId,
                    dispatchDate, warrantyYears, quantity, quality);
                price = m.calculateProductPrice();
                System.out.printf("Price of the product is %.2f", price);
                break;
            default:
                System.out.println("Provide a valid Product name");
                System.exit(0);
        }
    }
}
```

REVERSE A WORD

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        String[] words ;
        Scanner sc = new Scanner(System.in);

        String sentence= sc.nextLine();
        words=sentence.split(" ");
        if(words.length<3)
            System.out.println("Invalid Sentence");
        else{
            String a=words[0].substring(0,1);
            String b=words[1].substring(0,1);
            String c=words[2].substring(0,1);
            if(a.equalsIgnoreCase(b)&&b.equalsIgnoreCase(c))
            {
                StringBuilder input1 = new StringBuilder();
                input1.append(words[words.length-1]);
                // reverse StringBuilder input1
                input1= input1.reverse();
                input1.append(words[0]);

                System.out.println(input1);
            }
            else {
                StringBuilder input1 = new StringBuilder();
                input1.append(words[0]);
                // reverse StringBuilder input1
                input1= input1.reverse();
                input1.append(words[words.length-1]);

                System.out.println(input1);
            }
        }
        //the next part depends on question :
        if (words.length>3)
        {
            System.out.println("Invalid Word");
        }
    }
}
```

INITCAP PROJECT TITLE

```
import java.util.Scanner;
public class Main{
    public static void main(String[] args){
        System.out.println("Enter the project title");
        Scanner sc=new Scanner(System.in);
        String s = sc.nextLine();
        String[] words = s.split(" ");
        StringBuilder sb = new StringBuilder();
        for (String word : words) {
            for(int i=0; i<word.length(); i++) {
                if(Character.isLetter(word.charAt(i))==false) {
                    System.out.println("Invalid Input");
                    return;
                }
                if(i==0) {
                    sb.append(Character.toUpperCase(word.charAt(i)));
                } else {
                    sb.append(Character.toLowerCase(word.charAt(i)));
                }
            }
            sb.append(' ');
        }
        System.out.println(sb.toString().trim());
    }
}
```

PAYMENT INHERITANCE

BILL CLASS:

```
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 CLASS:

```
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 CLASS:

```
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 CLASS:

```
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;
    }
}
```

PAYMENT CLASS:

```
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 CLASS:

```
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();  
    }  
}
```

EXTRACT BOOK DETAILS

```
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();
}
```

BONBON PUBLISHING COMPANY

ADVERTISEMENT CLASS:

```
public abstract class Advertisement {  
protected int advertisementId;  
protected String priority;  
protected int noOfDays;  
protected String clientName;  
public void setAdvertisementId(int advertisementId) {  
this.advertisementId = advertisementId;  
}  
public int getAdvertisementId() {  
return advertisementId;  
}  
public void setPriority(String priority) {  
this.priority = priority;  
}  
public String getPriority() {  
return priority;  
}  
public void setNoOfDays(int noOfDays) {  
this.noOfDays = noOfDays;  
}  
public int getNoOfDays() {  
return noOfDays;  
}  
public void setClientName(String clientName) {  
this.clientName = clientName;  
}  
public String getClientName() {  
return clientName;  
}  
public Advertisement(int advertisementId, String priority, int noOfDays, String clientName) {  
this.advertisementId = advertisementId;  
this.priority = priority;  
this.noOfDays = noOfDays;  
this.clientName = clientName;  
}  
public abstract float calculateAdvertisementCharge(float baseCost);  
}
```

IMAGE ADVERTISEMENT CLASS:

```
public class ImageAdvertisement extends Advertisement {  
    private int inches;  
    public void setInches(int inches) {  
        this.inches = inches;  
    }  
    public int getInches() {  
        return inches;  
    }  
    public ImageAdvertisement(int advertisementId, String priority, int noOfDays, String clientName, int  
    inches) {  
        super(advertisementId, priority, noOfDays, clientName);  
        this.inches = inches;  
    }  
    public float calculateAdvertisementCharge(float baseCost) {  
        float baseAdvertisementCost;  
        baseAdvertisementCost = baseCost * inches * noOfDays;  
        float boosterCost = 0;  
        float serviceCost = 0;  
        if(priority.equals("high")) {  
            boosterCost = (float) (0.1 * baseAdvertisementCost);  
            serviceCost = 1000;  
        }  
        else if(priority.equals("medium")) {  
            boosterCost = (float) (0.0 * baseAdvertisementCost);  
            serviceCost = 700;  
        }  
        else if(priority.equals("low")) {  
            boosterCost = 0;  
            serviceCost = 200;  
        }  
        return baseAdvertisementCost + boosterCost + serviceCost;  
    }  
}
```

TEXT ADVERTISEMENT CLASS:

```
public class TextAdvertisement extends Advertisement{
private int noOfCharacters;
public void setNoOfCharacters(int noOfCharacters) {
this.noOfCharacters = noOfCharacters;
}
public int getNoOfCharacters() {
return noOfCharacters;
}
public TextAdvertisement(int advertisementId, String priority, int noOfDays, String clientName, int noOfCharacters) {
super(advertisementId, priority, noOfDays, clientName);
this.noOfCharacters = noOfCharacters;
}
public float calculateAdvertisementCharge(float baseCost) {
float baseAdvertisementCost;
baseAdvertisementCost = baseCost * noOfCharacters * noOfDays;
float boosterCost = 0;
float serviceCost = 0;
if(priority.equals("high")) {
boosterCost = (float) (0.1 * baseAdvertisementCost);
serviceCost = 1000;
}
else if(priority.equals("medium")) {
boosterCost = (float) (0.0 * baseAdvertisementCost);
serviceCost = 700;
}
else if(priority.equals("low")) {
boosterCost = 0;
serviceCost = 200;
}
return baseAdvertisementCost + boosterCost + serviceCost;
}
```

VIDEO ADVERTISEMENT CLASS:

```
public class VideoAdvertisement extends Advertisement{
private int duration;
public void setDuration(int duration) {
this.duration = duration;
}
public int getDuration() {
return duration;
}
public VideoAdvertisement(int advertisementId, String priority, int noOfDays, String clientName, int duration) {
super(advertisementId, priority, noOfDays, clientName);
this.duration = duration;
}
public float calculateAdvertisementCharge(float baseCost) {
    float baseAdvertisementCost;
    baseAdvertisementCost = baseCost * duration * noOfDays;
    float boosterCost = 0;
    float serviceCost = 0;
    if(priority.equals("high")) {
        boosterCost = (float) (0.1 * baseAdvertisementCost);
        serviceCost = 1000;
    }
    else if(priority.equals("medium")) {
        boosterCost = (float) (0.07 * baseAdvertisementCost);
        serviceCost = 700;
    }
    else if(priority.equals("low")) {
        boosterCost = 0;
        serviceCost = 200;
    }
    return baseAdvertisementCost + boosterCost + serviceCost;
}
}
```

BONBON CLASS:

```
import java.util.Scanner;
public class BonBon {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the advertisement id");
        int id = sc.nextInt();
        System.out.println("Enter the priority(high, medium, low)");
        String priority = sc.next();
        System.out.println("Enter the no of days advertisement is published");
        int num = sc.nextInt();
        System.out.println("Enter the client name");
        String name = sc.nextLine();
        sc.nextLine();
        System.out.println("Enter the type of Advertisement (video/image/text)");
        String type = sc.next();
        if(type.equalsIgnoreCase("video")) {
            System.out.println("Enter the duration in minutes");
            int min = sc.nextInt();
            VideoAdvertisement obj = new VideoAdvertisement(id, priority, num, name, min);
            System.out.println("Enter the base cost");
            float base = sc.nextFloat();
            System.out.println("The Advertisement cost is " +
obj.calculateAdvertisementCharge(base));
        }
        else if(type.equalsIgnoreCase("image")) {
            System.out.println("Enter the number of inches");
            int inc = sc.nextInt();
            VideoAdvertisement obj = new VideoAdvertisement(id, priority, num, name, inc);
            System.out.println("Enter the base cost");
            float base1 = sc.nextFloat();
            System.out.println("The Advertisement cost is " +
obj.calculateAdvertisementCharge(base1));
        }
        else if(type.equalsIgnoreCase("text")) {
            System.out.println("Enter the number of characters");
            int text = sc.nextInt();
            VideoAdvertisement obj = new VideoAdvertisement(id, priority, num, name, text);
            System.out.println("Enter the base cost");
            float base2 = sc.nextFloat();
            System.out.println("The Advertisement cost is " +
obj.calculateAdvertisementCharge(base2));
        }
    }
}
```

ALTERNATE NUMBERS DIFFERENCE

```
import java.util.Scanner;
import java.lang.Math;

public class AlternateNumbersDifference {

    public static void main(String[] args) {
        int n1=0;
        int n2=0;
        Scanner sc=new Scanner(System.in);
        int a[]={};
        System.out.println("Enter the array size");
        int size=sc.nextInt();
        if(size<5||size>10)
        {
            System.out.println("Invalid array size");
            return;
        }
        System.out.println("Enter the array elements");
        for(int i=0;i<size;i++)
        {
            a[i]=sc.nextInt();
        }
        int x[]={};
        int max=x[0];
        for(int i=0;i<size;i++)
        {
            if(i+2<size){
                x[i]=Math.abs(a[i]-a[i+2]);
                if(x[i]>max){
                    max=x[i];
                    n1=a[i];
                    n2=a[i+2];
                }
            }
            else
                continue;
        }
        int min=0;

        if(n1>n2)
            min=n2;
        else
            min=n1;

        for(int i=0;i<size;i++)
        {
            if(a[i]==min){
                System.out.println(i);
                break;
            }
        }
    }
}
```

ALLITERATION

```
import java.util.Scanner;
public class Alliteration {

    public static void main (String[] args) {
        Scanner sc=new Scanner(System.in);
        int count=0;
        System.out.println("Enter the letter");
        char aletter=sc.next().charAt(0);
        char acon=Character.toLowerCase(aletter);
        sc.nextLine();
        String sentence_letter=sc.nextLine();
        String cons=sentence_letter.toLowerCase();
        char ch[]=new char[cons.length()];
        for(int i=0;i<cons.length();i++)
        {
            ch[i]=cons.charAt(i);
            if( ((i>0)&&(ch[i]!=' ')&&(ch[i-1]==' ') && (ch[i]==acon)) || ((ch[0]!=' ')&&(i==0)) )
            {
                count++;
            }
        }
        // System.out.println(count);
        if(count>3)
        {
            System.out.println("Good,You get a score of "+(2+(count-3)*2));
        }
        else if(count == 3)
        {
            System.out.println("Good,You get a score of "+2);
        }
        else if(count < 3)
        {
            System.out.println("No score");
        }
    }
}
```

PASSENGER AMENITY

```
import java.util.Arrays;
import java.util.Scanner;
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");
            }
        }
    }
}
```

CHANGING THE CASE

(EASY VERSION)

```
import java.util.Scanner;
public class alternate {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        String str=sc.next();

        if(str.length()<3)
        {
            System.out.println("String length of "+str+" is too short");
            System.exit(0);
        }
        else if(str.length()>10)
        {
            System.out.println("String length of "+str+" is too long");
            System.exit(0);
        }
        String str1="";
        boolean val=true;
        for(int i = 0; i<str.length();i++)
        {
            if(!Character.isAlphabetic(str.charAt(i)))
            {
                val=false;
                str1=str1+str.charAt(i);
            }
        }
        if(val==false)
        {
            System.out.println("String should not contain "+str1);
            System.exit(0);
        }
        char ch=sc.next().charAt(0);
        char cc,cc2,ch2;
        if(Character.isUpperCase(ch))
        {
            cc='u';
            ch2=Character.toLowerCase(ch);
            cc2='l';
        }
        else
        {
            cc='l';
            ch2=Character.toUpperCase(ch);
            cc2='u';
        }
        int index=str.indexOf(ch);
        if(index==-1)
        {
            System.out.println("character "+ch+" is not found");
            System.exit(0);
        }
        String str2="";
        for(int i=0;i<str.length();i++)
        {
            if(str.charAt(i)==ch && cc=='l')
            {
                str2=str2+Character.toUpperCase(str.charAt(i));
            }
            else if(str.charAt(i)==ch && cc=='u')
            {
                str2=str2+Character.toLowerCase(str.charAt(i));
            }
        }
        System.out.println(str2);
        System.exit(0);
    }
}
```

```

        }
        else if(str.charAt(i)==ch2 && cc=='u')
        {
            str2=str2+Character.toLowerCase(str.charAt(i));
        }
        else if(str.charAt(i)==ch2 && cc=='l')
        {
            str2=str2+Character.toUpperCase(str.charAt(i));
        }
        else
        {
            str2=str2+str.charAt(i);
        }
    }
    System.out.println(str2);
}

}

```

(ADVANCED VERSION)

```

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;

public class Main {

    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String str = br.readLine();
        if (str.length() < 3)
            System.out.println("String length of " + str.length() + " is too short");
        else if (str.length() > 10)
            System.out.println("String length of " + str.length() + " is too long");
        else {

            if (Main.checkValidity(str).size() == 0) {
                char ch = (char) br.read();
                if (Main.alterString(str, ch).equals(str))
                    System.out.println("Character not found");
                else
                    System.out.println(Main.alterString(str, ch));
            } else {
                System.out.print("String should not contain ");
                for (char c : Main.checkValidity(str))
                    System.out.print(c);
            }
        }
    }

    public static String alterString(String str, char ch) {
        String newstr = "";
        for (int i = 0; i < str.length(); i++) {
            if (Character.toUpperCase(str.charAt(i)) == Character.toUpperCase(ch))
                newstr = newstr + Main.altercase(str.charAt(i));
            else
                newstr = newstr + str.charAt(i);
        }
        return newstr;
    }
}

```

```
public static char altercase(char c) {
    if (Character.isUpperCase(c))
        return Character.toLowerCase(c);
    else
        return Character.toUpperCase(c);
}

public static List<Character> checkValidity(String str) {
    List<Character> ch = new ArrayList<>();
    int j = 0;
    for (int i = 0; i < str.length(); i++) {
        if (((65 <= (int) str.charAt(i)) && ((int) str.charAt(i) <= 90))
            || ((97 <= (int) str.charAt(i)) && ((int) str.charAt(i) <= 122)) || ((int)
str.charAt(i) == 32))
            continue;
        else
            ch.add(str.charAt(i));
    }
    return ch;
}
}
```

BANK ACCOUNT INTERFACE

TWO CODES ARE GIVEN HERE, PLEASE CHOOSE ACCORDINGLY AS PER GIVEN IN THE QUESTION

CODE-1:

ACCOUNT CLASS:

```
public class Account {  
  
    String accountNumber;  
    String customerName;  
    double 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;  
    }  
  
    public Account(String accountNumber, String customerName, double balance) {  
        super();  
        this.accountNumber = accountNumber;  
        this.customerName = customerName;  
        this.balance = balance;  
    }  
}
```

CURRENT ACCOUNT CLASS:

```
public class CurrentAccount extends Account implements MaintenanceCharge {  
    public CurrentAccount(String customerName, String accountNumber, double balance) {  
        super(accountNumber, customerName, balance);  
        // TODO Auto-generated constructor stub  
    }  
  
    @Override  
    public float calculateMaintenanceCharge(float noOfYears) {  
        // TODO Auto-generated method stub  
        int m = 100;  
        int n = (int) noOfYears;  
        float mCharge = (m*n) + 200;  
        return mCharge;  
    }  
}
```

SAVINGS ACCOUNT CLASS:

```
public class SavingsAccount extends Account implements MaintenanceCharge{  
    public SavingsAccount(String customerName, String accountNumber, double balance) {  
        super(accountNumber, customerName, balance);  
        // TODO Auto-generated constructor stub  
    }  
  
    @Override  
    public float calculateMaintenanceCharge(float noOfYears) {  
        // TODO Auto-generated method stub  
        int m = 50;  
        int n = (int) noOfYears;  
        float mCharge = (m*n) + 50;  
        return mCharge;  
    }  
}
```

MAINTENANCE CHARGE CLASS:

```
public interface MaintenanceCharge {  
    public float calculateMaintenanceCharge (float noOfYears);  
}
```

USER INTERFACE CLASS:

```
import java.util.Scanner;

public class UserInterface {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("1. Savings Account");
        System.out.println("2. Current Account");
        System.out.println("Enter your choice:");

        int choice = sc.nextInt();

        System.out.println("Enter the Account number");
        sc.nextLine();
        String acc = sc.nextLine();

        System.out.println("Enter the Customer Name");

        String custName = sc.nextLine();

        System.out.println("Enter the Balance amount");
        double balance = sc.nextDouble();

        System.out.println("Enter the number of years");
        int noOfYears = sc.nextInt();

        double mCharges = 0;

        if(choice == 1){
            SavingsAccount sa = new SavingsAccount(custName, acc, balance);
            mCharges = sa.calculateMaintenanceCharge(noOfYears);
            System.out.println("Customer Name " +custName);
            System.out.println("Account Number " +acc);
            System.out.println("Account Balance " +balance);
            System.out.println("Maintenance Charge for Savings Account is Rs " +mCharges);
        }

        if(choice == 2){
            CurrentAccount ca = new CurrentAccount(custName, acc, balance);
            mCharges = ca.calculateMaintenanceCharge(noOfYears);
            System.out.println("Customer Name " +custName);
            System.out.println("Account Number " +acc);
            System.out.println("Account Balance " +balance);
            System.out.println("Maintenance Charge for Current Account is Rs " +mCharges);
        }

        //use below code instead of upper one if required
        /* System.out.println("Customer Name " +custName);
        System.out.println("Account Number " +acc);
        System.out.println("Account Balance " +balance);
        System.out.println("Maintenance Charge for Current Account is Rs " +mCharges); */

    }
}
```

CODE-2:

ACCOUNT CLASS:

```
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;  
    }  
}
```

CURRENT ACCOUNT CLASS:

```
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;  
    }  
}
```

SAVINGS ACCOUNT CLASS:

```
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;  
    }  
}
```

MAINTENANCE CHARGE CLASS:

```
public interface MaintenanceCharge {  
    float calculateMaintenanceCharge(float noOfYears);  
}
```

USER INTERFACE CLASS:

```
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)));  
            }  
        }  
    }  
}
```

A NEW YOU SPA

MEMBERS CLASS:

```
public abstract 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;  
    }  
}
```

DIAMOND MEMBERS CLASS:

```
public class DiamondMembers extends Members {
    public DiamondMembers(String customerId, String customerName, long mobileNumber, String memberType,
                          String emailId) {
        super(customerId, customerName, mobileNumber, memberType, emailId);
        // TODO Auto-generated constructor stub
    }
    public boolean validateCustomerId(String customer_id){
        int id_len = customer_id.length();
        int chk = id_len-3;
        String type = customer_id.substring(0,chk);
        String str = customer_id.substring(chk,id_len);
        String s2 = "Diamond";
        if(type.equals(s2)){
            for(int i=0;i<3;i++){
                if(str.charAt(i)>='0' && str.charAt(i)<='9'){
                    return true;
                }
                else{
                    return false;
                }
            }
            return false;
        }else {
            return false;
        }
    }
    public double calculateDiscount(double purchaseAmount){
        double discount = 0.45*purchaseAmount;
        double new_amount_to_be_paid = purchaseAmount-discount;
        return new_amount_to_be_paid;
    }
}
```

PLATINUM MEMBERS CLASS:

```
public class PlatinumMembers extends Members {
    public PlatinumMembers(String customerId, String customerName, long mobileNumber, String memberType,
                           String emailId) {
        super(customerId, customerName, mobileNumber, memberType, emailId);
        // TODO Auto-generated constructor stub
    }
    public boolean validateCustomerId(String customer_id){
        int id_len = customer_id.length();
        int chk=id_len-3;
        String type= customer_id.substring(0,chk);
        String str = customer_id.substring(chk,id_len);
        String s2 = "Platinum";
        if(type.equals(s2)){
            for(int i=0;i<3;i++){
                if(str.charAt(i)>='0' && str.charAt(i)<='9'){
                    return true;
                }
                else {
                    return false;
                }
            }
            return false;
        }else{
            return false;
        }
    }
    public double calculateDiscount(double purchaseAmount){
        double discount = 0.30*purchaseAmount;
        double new_amount_to_be_paid= purchaseAmount-discount;
        return new_amount_to_be_paid;
    }
}
```

GOLD MEMBERS CLASS:

```
public class GoldMembers extends Members {

    public GoldMembers(String customerId, String customerName, long mobileNumber, String memberType,
String emailId) {
        super(customerId, customerName, mobileNumber, memberType, emailId);
    }
    public boolean validateCustomerId(String customer_id){
        int id_len = customer_id.length();
        int chk = id_len-3;
        String type = customer_id.substring(0,chk);
        String str = customer_id.substring(chk,id_len);
        String s2 = "Gold";
        if(type.equals(s2)){
            for (int i=0;i<3;i++){
                if(str.charAt(i)<='9'){
                    return true;
                }
                else{
                    return false;
                }
            }
            return false;
        }else{
            return false;
        }
    }
    public double calculateDiscount(double purchaseAmount){
        double discount= 0.15*purchaseAmount;
        double new_amount_to_be_paid= purchaseAmount-discount;
        return new_amount_to_be_paid;
    }
}
```

USER INTERFACE CLASS:

```
import java.util.Scanner;
public class UserInterface {

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

        System.out.println("\n enter customer id");
        String cid;
        cid=sc.nextLine();

        System.out.println("\n enter customer name");
        String cnm;
        cnm=sc.nextLine();

        System.out.println("\n enter mobile number");
        Long cph;
        cph=sc.nextLong();

        sc.nextLine();
        System.out.println("\n enter member type");
        String mem;
        mem=sc.nextLine();

        System.out.println("\n enter email id");
        String mail;
        mail=sc.nextLine();

        System.out.println("\n enter amount purchased");
        String customer_amount_purchased;
        customer_amount_purchased=sc.nextLine();
        double amount_purchased = Double.parseDouble(customer_amount_purchased);

        DiamondMembers dia = new DiamondMembers(cid,cnm,cph,mem,mail);
        GoldMembers gld = new GoldMembers(cid,cnm,cph,mem,mail);
        PlatinumMembers plt = new PlatinumMembers(cid,cnm,cph,mem,mail);

        double new_amount;

        if(dia.validateCustomerId(cid)) {
            new_amount = dia.calculateDiscount(amount_purchased);
            System.out.println("\n Name -- " + cnm);
            System.out.println("\n Id -- " + cid);
            System.out.println("\n Email id -- " + mail);
            System.out.println("\n Amount to be paid -- " + new_amount);
        } else if (gld.validateCustomerId(cid)) {
            new_amount = gld.calculateDiscount(amount_purchased);
            System.out.println("\n Name -- " + cnm);
            System.out.println("\n Id -- " + cid);
            System.out.println("\n Email id -- " + mail);
            System.out.println("\n Amount to be paid -- " + new_amount);
        } else if (plt.validateCustomerId(cid)) {
            new_amount = plt.calculateDiscount(amount_purchased);
            System.out.println("\n Name -- " + cnm);
            System.out.println("\n Id -- " + cid);
            System.out.println("\n Email id -- " + mail);
            System.out.println("\n Amount to be paid -- " + new_amount);
        } else {
            System.out.println("\n Invalid Id. \n Please provide valid customer id");
        }
    }
}
```

SUBSTITUTION CIPHER TECHNIQUE

(EASY TO UNDERSTAND)

```
import java.util.Scanner;

public class alternateSubstitutionCipher {
    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");
    }
}
```

(MODERATE CODE)

```
import java.util.Scanner;
class SubstitutionCipherTechnique {
    public static void main (String[] args) {
        Scanner s=new Scanner(System.in);
        String s1;
        char arr[],arr_2[];
        System.out.println("Enter the encrypted text:");
        s1=s.nextLine();
        int i,j=0,counter=0;
        arr=s1.toCharArray();
        arr_2=new char[arr.length];
        for(i=0;i<arr.length;i++)
        {
            int value=(int)arr[i];
            if(value<65 || (value>90 && value<97)){
                if(value==32)
                    arr_2[j++]= ' ';
                else
                    continue;
            }
            else{
                value=value-7;
                if(value<97 && value>89)
                    value=value+26;
                if(value<65)
                    value=value+26;

                arr_2[j++]=(char)value;
                counter=1;
            }
        }
        if(counter!=1){
            System.out.println("No hidden message");
            System.exit(0);
        }
        System.out.println("Decrypted text:");
        for(i=0;i<arr_2.length;i++){
            System.out.print(arr_2[i]);
        }
    }
}
```

Lambda Expression

hackerrank.com/test/e2nd3mlog1c/questions/4g748qh9im7
Python Tutorial - T...

CDER201 - Data Structures & Algorithms, Problem Solving, Java, Microservices & SpringBoot - GenC

4. Lambda Expressions

```
Line 1: import java.util.function.Predicate;
Line 2: public class PredicateDemo {
Line 3:     public static void main(String[] args) {
Line 4:         Predicate<String> compareString = string -> {
Line 5:             return string.equals("Java");
Line 6:         };
Line 7:         ...
Line 8:     }
Line 9: }
```

The main() method of PredicateDemo class is expected to print "true" on the standard output. Inserted at Line 7 to get the desired output. Choose THREE options.

Pick ONE OR MORE options

System.out.println(compareString.test("Java"));
 System.out.println(compareString.equals("Java"));
 System.out.println(compareString.negate().test("not Java"));
 System.out.println(compareString.and(compareString).test("Java"));
 System.out.println(compareString.or(compareString).test("Java"));

Type here to search

CDER201 - Data Structures & Algorithms, Problem Solving, Java, Microservices & SpringBoot - GenC

5. Lambda Expressions

Take a look at the code given.

```
Line 1: interface NumberChecker{
Line 2:     boolean resolve(int a);
Line 3: }
```

Type here to search

Windows Taskbar icons: Start, File Explorer, Mail, Photos, OneDrive, Edge, File, Settings, Control Panel, Task View, Task Manager, Power, Network, File History, Help and Support, Taskbar settings.

Keyboard layout: QWERTY (US English)

Key positions:

- Top row: Q, W, E, R, T, Y, U, I, O
- Second row: A, S, D, F, G, H, J, K
- Third row: Z, X, C, V, B, N, M

7. Lambda Expressions

```

Take a look at the code given.
Line 1: import java.util.ArrayList;
Line 2: class PrintArrayList {
Line 3:     public static void main(String args[]){
Line 4:         ArrayList<Integer> arrayList = new ArrayList<Integer>();
Line 5:         arrayList.add(1);
Line 6:         arrayList.add(2);
Line 7:         arrayList.add(3);
Line 8:         arrayList.add(4);
Line 10:        ...
Line 11:    }
}

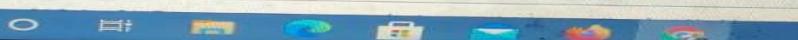
For the above given code, what possible statements from the following options(Can output as below:
1
3

```

Pick ONE OR MORE options

- `arrayList.forEach(index -> { if (index%2 != 0) System.out.println(index); })`

Type here to search



CDER201 - Data Structures & Algorithms, Problem Solving, Java, Microservices & SpringBoot - GenC

To make the `UtilityClass` print true or false (i.e., odd or not) for the number(s) that can be inserted Line 9. Choose TWO options

Pick ONE OR MORE options

- `return odd -> (odd & 1) == 1;`

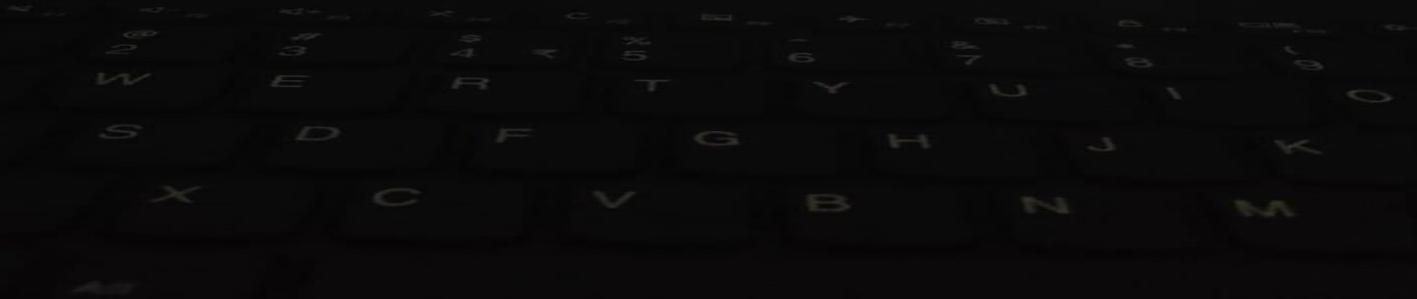
- `return (odd & 1 == 1);`

- `return odd -> (odd % 2) != 0;`

- `return (if (odd & 1) ==1);`

Clear Selection

Type here to search



Pick ONE OR MORE options

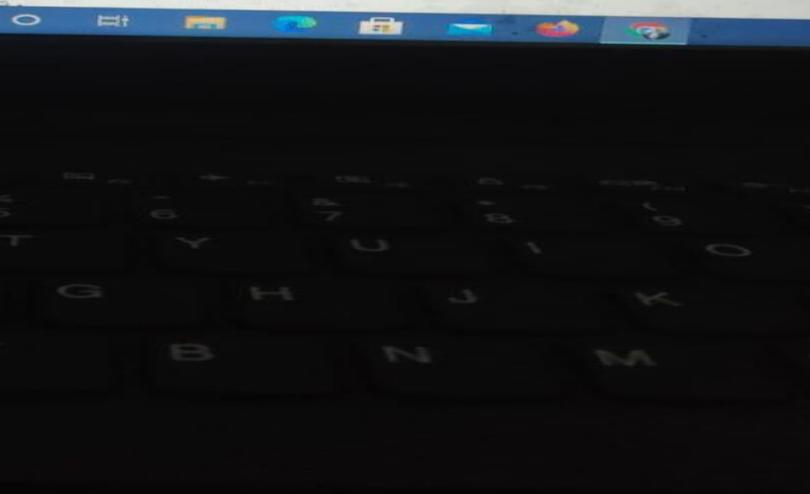
- `arrayList.forEach(index -> { if (index%2 != 0) System.out.println(index); })`
- `arrayList.forEach(index -> { if (index%2 != 0) System.out.println(arrayList.get(index)); })`
- `arrayList.forEach(index -> { if (index%2 != 0) System.out.println(arrayList.equals(index)); })`
- `arrayList.forEach(index -> { if (index%2 != 0) System.out.println(arrayList.get(index-1)); })`

[Clear Selection](#)

8. Comparator

For the below given Account class,

Type here to search

→ C <https://hackerrank.com/test/e2nd3mlog1c/questions/dk2aiq4tafn>

Apps Python Tutorial - T...

8. Comparator

For the below given Account class,

```
public class Account{
    private String id;
    public void setId(String id){
        this.id=id;
    }
    public String getId(){
        return id;
    }
}
```

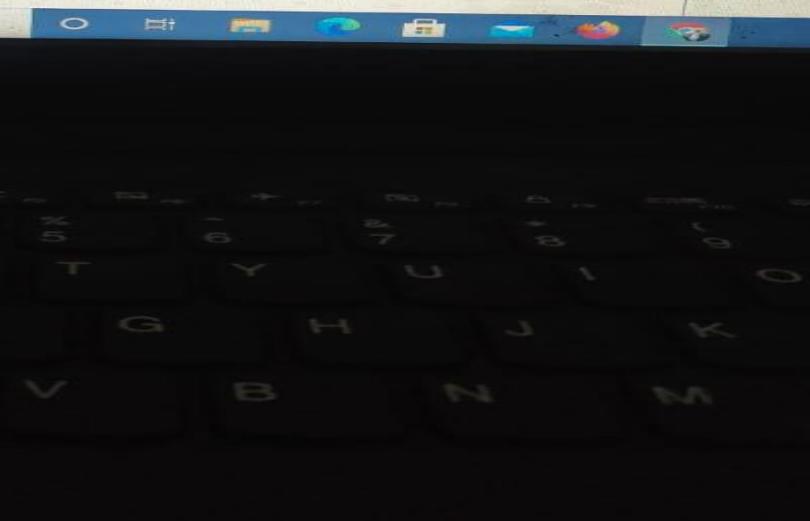
Analyze the given below options and select all that defines a valid comparator.

Pick ONE OR MORE options

- `Comparator compareId=Comparator.comparing(Account::getId);`

```
Comparator<Account> compareId = new Comparator<Account>(){
    public int compare(Account acc1, Account acc2){
         return acc1.getId().compareTo(acc2.getId());
    }
}
```

Type here to search



5. Lambda Expressions

```
Take a look at the code given.  
Line 1: interface NumberChecker{  
Line 2:     boolean resolve(int a);  
Line 3: }  
Line 4: class UtilityClass{  
Line 5:     public static boolean checkNumber(NumberChecker numCheck, int number){  
Line 6:         return numCheck.resolve(number);  
Line 7:     }  
Line 8:     public NumberChecker isOddNumber() {  
Line 9:         ...  
Line 10:    }  
Line 11:    public static void main(String args[]){  
Line 12:        UtilityClass utility = new UtilityClass();  
Line 13:        NumberChecker numCheck = utility.isOddNumber();  
Line 14:        System.out.println(numCheck.resolve(5));  
Line 15:    }  
Line 16: }
```

To make the UtilityClass print true or false (i.e., odd or not) for the number passed statement(s) that can be inserted Line 9. Choose Two options.

Pick ONE OR MORE options

Type here to search



Q E F G H J K
A S D C V B N M
W X E R T Y U I O
Z # 3 4 R 5 6 7 8 9
@ 2 F 1

SPRING BOOT

← → C hackerrank.com/test/e2nd3mlog1c/questions/7bhm9tg631g

H CDER201 - Data Structures & Algorithms, Problem Solving, Java, Microservices & SpringBoot - GenC Answered: 6 / 23 ⏱ 01 hour 25 mins ⌂ Shrestha

20. Spring Boot

ALL In a Spring Boot application, a microservice need to invoke another microservice in asynchronous way. Select from the given options to implement it. Choose Three options

① Pick ONE OR MORE options

13 Need to use @EnableAsync in the spring boot application startup class.

14 Need to annotate with @Async in the method from where the other microservice will get invoked.

15 Need to have the return type of the method invoking the other microservice as CompletableFuture<T>.

16 Need to use ExecuterService class in order to invoke the microservice asynchronously.

17 Clear Selection

18 ⌂ Saved!

Activate Windows
Go to Settings to activate Windows

Desktop » ⌂ ENG 3:50 PM
US 3/22/2021

Type here to search

ALL

23. Spring Boot

You do not want to restart the Spring boot app to reflect the change in configuration file. How to reload the configuration properties without restarting the app? Select your recommendations from the following options.

Pick **ONE** option

17 Adding @RefreshConfig at the controller class.

18 Adding @RefreshProperties at the controller class.

19 Adding @RefreshScope at the controller class.

20 None of the listed options.

Clear Selection

21 Saved

22

23 **Continue**

Activate Windows
Go to Settings to activate Windows

ALL

Take a look at the code given.

```
Line 1: ...
Line 2: interface Incrementor {
Line 3: ...
Line 4: }
Line 5: class Demo {
Line 6:     public static void main(String args[]) {
Line 7:         Incrementor incrementor = (number) -> number+10;
Line 8:         System.out.println(incrementor.incrementByValue(10));
Line 9:     }
Line 10: }
```

When the above code is compiled and run, it is expected to print the value 20 on the default output stream. What are the statement(s) that need to be inserted at Line 1 and Line 3 to get the expected value?

Pick **ONE OR MORE** options

8 Line 1: @FunctionalInterface
Line 3: public int incrementByValue(int num);

9 Line 1: Can be a blank line
Line 3: public int incrementByValue(int num);

10 Line 1: @FunctionalInterface Line 3: default int incrementByValue(int number) { return number+10; }

11 Line 1: Can be a blank line Line 3: int incrementByValue(int number) { return number+10; }

12 Clear Selection

Activate Windows
Go to Settings to activate Windows

ALL

2

3

4

5

6

7

8

```
Line 1: import java.util.function.Predicate;
Line 2: public class PredicateDemo {
Line 3:     public static void main(String[] args) {
Line 4:         Predicate<String> compareString = string -> {
Line 5:             return string.equals("Java");
Line 6:         };
Line 7:         ...
Line 8:     }
Line 9: }
```

The main() method of PredicateDemo class is expected to print "true" on the standard output stream. What possible statement(s) can be inserted at Line 7 to get the desired output. Choose THREE options.

Pick ONE OR MORE options

 System.out.println(compareString.test("Java")); System.out.println(compareString.equals("Java")); System.out.println(compareString.negate().test("not Java")); System.out.println(compareString.and(compareString).test("Java"));

Clear Selection

9

10

11

12

13

14

15

16

17

18

19

20

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

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

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

13. Spring Boot - Microservices

ALL

(i)

10

11

12

13

14

15

16

The train ticket management system allows the customer to book the tickets in various ways like normal booking, takal and p...
the customer to login and view the booked history or look for available train for various routes. Currently there is a single co...
that takes care of all the above activities.

As the request and response of this portal is handled by single server and client, there is a performance issue.
Revise this design by choosing one of the below options in such a way that there will be number of services for each request
balancer without affecting the usage of the portal by the customer.

1. Create Eureka Server , which in turn treats itself as a client, where every services is registered to get a dependent microse...
2. Create seperate service class for each use cases of the business need and configure the same in SpringBootApplication d...
3. Create multiple REST Client of Spring Boot application to handle the service class

Pick **ONE** option

 I only II only Both I and II III only**DELL**

Type here to search



ALL



12. Microservices

Select the benefits of using database per service. Select TWO options.

Pick **ONE OR MORE** options

9 It ensure that the services are loosely coupled. Changes to one service's database does not impact other services.

10 Each service can use the type of database that is best suited to its needs.

11 Implementing business transactions that span multiple services is straightforward.

12 Implementing queries that join data that is now in multiple databases is easier.

13 Clear Selection

14

15



Type here to search



DELL



ALL

```
Line 6: List<Integer> numbers = Arrays.asList(10,20,30,40);
Line 7: ...
Line 8: System.out.println(result);
Line 9: }
Line 10: }
```



In the above given SummingClass, what possible statement(s) can be inserted at Line 7 to sum all the 'numbers'. Choose FOUR options.

3

Pick **ONE OR MORE** options

4

int result = numbers.stream().reduce(0, (subtotal) -> subtotal + element);

5

int result = numbers.stream().collect(Collectors.summingInt(Integer::intValue));

6

int result = numbers.parallelStream().reduce(0, Integer::sum);

7

int result = numbers.stream().reduce(0, Integer::sum);

8

int result = numbers.stream().reduce(0, (subtotal, element) -> subtotal + element);

9

Clear Selection



Type here to search



DELL

HackerRank C https://www.hackerrank.com/test/e2nd3mlog1c/questions/7bhm9tg631g

CDER201 - Data Structures & Algorithms, Problem Solving, Java, Microservices & SpringBoot - GenC

ALL

20. Spring Boot

In a Spring Boot application, a microservice need to invoke another microservice in asynchronous way. Select from the given Pick **ONE OR MORE** options

Need to use @EnableAsync in the spring boot application startup class.

Need to annotate with @Async in the method from where the other microservice will get invoked.

Need to have the return type of the method invoking the other microservice as CompletableFuture<T>.

Need to use ExecuterService class in order to invoke the microservice asynchronously.

Clear Selection

(S) Saved

21. Spring Boot

Windows Taskbar icons: File Explorer, Edge, File Manager, Task View, Start, Taskbar search.

Keyboard layout: QWERTY, US English.

HackerRank

https://www.hackerrank.com/test/e2nd3mlog1c/questions/f1m5bmir021

CDER201 - Data Structures & Algorithms, Problem Solving, Java, Microservices & SpringBoot - GenC

ALL

19. Spring Boot

Assume in a given scenario, a checked exception occurred in a DAO component and the data flow is in a chain transaction.

Pick **ONE** option

The previous transaction gets rollback even if transaction annotation is not used.

The previous transaction do not get rollback automatically even if transaction annotation is used.

The previous transaction gets rollback even if transaction annotation is used.

None of the listed options.

Clear Selection

(S) Saved

20. Spring Boot

Windows Taskbar icons: Start, File Explorer, Edge, File, Mail, Calendar, Photos, OneDrive, Microsoft Store, Task View, Task Manager.

Keyboard layout: QWERTY, showing keys A through Z.

23. Spring Boot

You do not want to restart the Spring boot app to reflect the change in configuration file. How to reload the configuration properties without restarting the app? Select

Pick **ONE** option

Adding @RefreshConfig at the controller class.

Adding @RefreshProperties at the controller class.

Adding @RefreshScope at the controller class.

None of the listed options.

Clear Selection



Continue



⌘

ALL

ⓘ

6

7

8

9

10

11. Microservices

What are the benefit of Single Service per Host model in Microservices? Select THREE options.

Pick **ONE OR MORE** options

- It avoids single point of failure. A failure on one host will only impact one service at a time.
- It makes monitoring and security easier than multiple services on one host.
- It is easier to scale in case we want a specific service to be scaled up.
- It efficiently utilize resources compared to Multiple Services per Host.

[Clear Selection](#)

[Saved!](#)



ALL



7

8

9

10

11

12

13

10. Microservices

Mr. John is challenged with a discussion on how microservices are prevailing rather than monolithic architecture; while benefits of micro services. What are the value adds which put microservices on top? Choose TWO Options.

Pick **ONE OR MORE** options

Less chances of maintaining the fault tolerance.

Scalability of the applications improves the performance on servers.

Size of the application can degrade the performance and slow down the start up.

Majorly supports the complex application development.

[Clear Selection](#)



Type here to search



Dell

#

ALL

()

```
Line 8: public NumberChecker isOddNumber() {  
Line 9:     ...  
Line 10: }  
Line 11: public static void main(String args[]){  
Line 12:     UtilityClass utility = new UtilityClass();  
Line 13:     NumberChecker numCheck = utility.isOddNumber();  
Line 14:     System.out.println(numCheck.resolve(5));  
Line 15: }  
Line 16: }
```

5

To make the UtilityClass print true or false (i.e, odd or not) for the number passed as statement(s) that can be inserted Line 9. Choose TWO options

6

Pick **ONE OR MORE** options

7

return odd -> (odd & 1) == 1;

8

return (odd & 1 == 1);

9

return odd -> (odd % 2) != 0;

10

return (if(odd & 1) ==1);

11

Clear Selection



Type here to search



HackerRank

https://www.hackerrank.com/test/e2nd3mlog1c/questions/f54m7jhct5g

CDER201 - Data Structures & Algorithms, Problem Solving, Java, Microservices & SpringBoot - GenC

ALL

18. Spring Boot

If you do not want to do default component scan in Spring Boot application, how to disable it?

Pick **ONE** option

14

15 @ComponentScan(disable)

16 @ComponentScan(useDefaultFilters=false)

17 @ComponentScan(useDefaultFilter=true)

18 None of the listed options.

19 Clear Selection

20 Saved

21

22 19. Spring Boot

Assume in a given scenario, a checked exception occurred in a DAO component and the data flow is in a chain transaction.

ALL



22. Spring Boot

12

Select from the given options the way to handle exception globally in Spring

13

Pick **ONE** option

14

Using @ExceptionHandler at class level.

15

Using @ControllerAdvice at class level.

16

Using @HandlerExceptionResolver at class level.

17

None of the listed options.

18

Clear Selection

36

ALL



13

Shopping Management System allows the customers to order products online. If a customer searches for a product that along with HTTP_NOT_FOUND status should be returned. This exception should be handled using the Spring Boot Exception Handler. Which of the following code snippets implements the mentioned requirement?

Pick ONE option

14 @ExceptionHandler(ShoppingManagementException.class)

public final ResponseEntity<ExceptionResponse> invalidData(ShoppingManagementException exception) { ExceptionResponse exceptionResponse = new ExceptionResponse(exception.getMessage(), "OutOfStock"); return new ResponseEntity<ExceptionResponse>(exceptionResponse, new HttpHeaders(), HttpStatus.NOT_FOUND); }

15 @ExceptionHandler(ShoppingManagementException.class)

public final ResponseEntity<ExceptionResponse> invalidData(ShoppingManagementException exception) { ExceptionResponse exceptionResponse = new ExceptionResponse(HttpStatus.NOT_FOUND, "OutOfStock"); return new ResponseEntity<ExceptionResponse>(exceptionResponse); }

17 @ExceptionHandler(ShoppingManagementException.class)

18 public final ResponseEntity<ExceptionResponse> invalidData(ShoppingManagementException exception) { ExceptionResponse exceptionResponse = new ExceptionResponse(exception.getMessage()); return new ResponseEntity<ExceptionResponse>(exceptionResponse, "OutOfStock", HttpStatus.NOT_FOUND); }

19 @ExceptionHandler(ShoppingManagementException.class)



Type here to search



DELL



ALL



8

9

10

11

12

13

14

11. Microservices

What are the benefit of Single Service per Host model in Microservices? Select THREE options.

Pick **ONE OR MORE** options

It avoids single point of failure. A failure on one host will only impact one service at a time.

It makes monitoring and security easier than multiple services on one host.

It is easier to scale in case we want a specific service to be scaled up.

It efficiently utilize resources compared to Multiple Services per Host.

Clear Selection

Type here to search



9. Microservice Architecture

Consider a monolithic application that runs all the services together in a single IP address. So whenever there is any downtime to the entire application.

Hence to overcome this impact microservices was introduced such that all the services will run independently and will not affect each other.

However as these microservices are running independently, it is not possible to know which services is up/down and which is not.

Revise this design and provide a solution such that it will solve the above problem.

- A. Implement Service Discovery Patterns where all the services are registered to the service registry such that the IP address of the service can be fetched.
- B. Implement Service Discovery patterns where the services are registered and the ports are exposed via REST APIs.
- C. Implement Service Discovery patterns where the REST API will support to fetch the IP address of the services are running.

Pick **ONE** option



A,C



A,B



B,C



C only

10

11

12

ALL

?

6

7

9

10

11

12

10

11

12

ALL

?

6

7

9

10

11

12

10

11

12

ALL

?

6

7

9

10

11

12

10

11

12

ALL

?

6

7

9

10

11

12

10

11

12

ALL

?

6

7

9

10

11

12

10

11

12

ALL

?

6

7

9

10

11

12

10

11

12

ALL

?

6

7

9

10

11

12

10

11

12

ALL

?

6

7

9

10

11

12

10

11

12

ALL

?

6

7

9

10

11

12

10

11

12

ALL

?

6

7

9

10

11

12

10

11

12

ALL

?

6

7

9

10

11

12

10

11

12

ALL

?

6

7

9

10

11

12

10

11

12

ALL

?

6

7

9

10

11

12

10

11

12

ALL

?

6

7

9

10

11

12

10

11

12

ALL

?

6

7

9

10

11

12

10

11

12

ALL

?

6

7

9

10

11

12

10

11

12

ALL

?

6

7

9

10

11

12

10

11

12

ALL

?

6

7

9

10

11

12

10

11

12

ALL

?

6

7

9

10

11

12

10

11

12

ALL

?

6

7

9

10

11

12

10

11

12

ALL

?

6

7

9

10

11

12

10

11

12

ALL

?

6

7

9

10

11

12

10

11

12

ALL

?

6

7

9

10

11

12

10

11

12

ALL

?

6

7

9

10

11

12

10

11

12

ALL

?

6

7

9

10

11

12

10

11

12

ALL

?

6

7

9

10

11

12

10

11

12

ALL

?

6

7

9

10

11

12

10

11

12

ALL

?

6

7

9

10

11

12

10

ALL

15. Spring Boot

i

What are all the HTTP methods that `@RequestMapping` can use while you want a Spring Boot class act as a controller?

12

Pick **ONE** option

13

GET

14

POST

15

PUT

16

All of the listed options.

Clear Selection

17

18



Type here to search



DELL

26

ALL

1

2

3

4

5

6

7

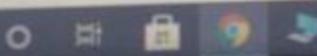
```
Line 8:    }
Line 9:    public static void printList(List<Integer> listOfIntegers, Consumer<Integer> consumer){
Line 10:      for(Integer integer:listOfIntegers){
Line 11:        consumer.accept(integer);
Line 12:      }
Line 13:    }
Line 14: }
```

The main() method of ConsumerDemo class is expected to print all the values in the integerList. Examine the code and select the statement that can be inserted at Line 5 to get the desired output?

Pick **ONE** option

- Consumer<Integer> consumer= number -> System.out.print(number + " ");
- Consumer<Integer> consumer = number -> number;
- Consumer<Integer> consumer = number -> System.out::print;
- Consumer<Integer> consumer= consumer.andThen(number ->System.out.println(number + " "));
- Consumer<Integer> consumer = number -> value;

Type here to search



DELL

CDE Rank - Java Practice Test

6. Lambda Expression - Easy

What is TRUE about Lambda?

Pick **ONE** option



Lambda expression enable functions to be passed as arguments.



It is neither a function nor a interface.



Lambda is denoted with => sign.



None of the given options.

Clear Selection

Continue



Type here to search



CDE Rank - Java Practice Test

8

LL

i

5

6

7

8

9

10

11

8. Monolith to MSA

You are breaking a HealthCare based Billing Application from Monolithic to Microservices based ap

Layer 1: User Interface

Layer 2: Business Logic

Layer 3: Database

The first capability you choose to decouple from the monolithic is the company's Billing Algorithm Database elements. What is wrong in this approach?

Pick **ONE** option

- You've selected a capability that doesn't include the UI to start the conversation with.
- You've selected a capability that doesn't include the database to start the conversation.
- You've selected a single capability to start the conversation with.
- You've selected a high-risk business critical capability to start the conversation with.

Type here to search



C hackerrank.com/test/25iher3t6te/questions/4tskgoge2jn

CDE Rank - Java Practice Test

9. Spring Boot Starter

You are developing a Spring Boot Application and need to add support for JDBC. Which Spring Boot s

Pick **ONE** option

springboot-starter-jdbc

spring-boot-jdbc-starter

spring-boot-starter-jdbc

jdbc-spring-boot-starter

[Clear Selection](#)

Type here to search





10. Auto Config

ALL

Which of the following is/are TRUE?



5

6

- A. If you added @SpringBootApplication annotation to the class, you do not need to add the @Configuration annotation.
- B. If you added @SpringBootApplication annotation to the class, you do not need to add the @EnableAutoConfiguration annotation.
- C. @SpringBootApplication annotated class should have the main method to run the Spring application.
- D. Either @EnableAutoConfiguration annotation or @SpringBootApplication annotation on your configuration class is required.

7

Pick **ONE** option

8

9

10

11

 A and D A, C and D A only B and D[Clear Selection](#) Type here to search

[←](#) [→](#) [↻](#)hackerrank.com/test/25iher3t6te/questions/bmk27pb36km

CDE Rank - Java Practice Test



7. Choosing MSA

ALL



You are evaluating 2 types of deployment structures for an application specific to your bu

4

Type 1: Many small independent teams working on many deployments

Type 2: A single large team working on a monolithic deployment

5

What valued business metrics might influence your decision to choose a Microservices ba

Pick **ONE** option

6

 Migration effort to a new technology stack

7

 Expertise of the developers in your project

8

 Shortening cycle times and improved reusability

9

 Version Control and Maintenance Cost

10

[Clear Selection](#)

Type here to search





ALL

5. Lambda Expression - Easy



Which of the following is an example of the internal iteration?

- Pick **ONE** option
- 2 for(Person p: list) { System.out.println(p); }
 - 3 for(int i=0; i<list.size();i++) { Person p = list[i]; System.out.println(p); }
 - 4 list.forEach(s-> System.out.println(s));
 - 5 System.out.println(list);

Clear Selection

Type here to search





CDE Rank - Java Practice Test

4. Higher Order Function

Which one of these is an example of Higher Order Function?

Pick **ONE** option



list.sort(() -> p1.getName().compareTo(p2.getName()));



list.sort((p1, p2) -> p1.getName().compareTo(p2.getName()));



list.sort((Person p1, Person p2) -> p1.getName().compareTo(p2.getName()));



list.sort(Comparator.comparing(p -> p.getName()));

Clear Selection



Type here to search





ALL

3. Method Reference



Which one below is the example of Method reference?

Pick **ONE** option

1

 list.replaceAll(String::toUpperCase())

2

 list.replaceAll(String::toUpperCase)

3

 list.replaceAll(s -> s.toUpperCase())

4

 None of the listed options.

5

[Clear Selection](#)

6

7

 Type here to search

15. Spring Boot

What are all the HTTP methods that @RequestMapping can use while you want a Spring Boot class act as a Rest Controller?

Pick **ONE** option

 GET POST PUT All of the listed options.

[Clear Selection](#)

19. Spring Boot

Assume in a given scenario, a checked exception occurred in a DAO component and the data flow is in a chain transaction. What will happen to the previously executed transaction?

Pick **ONE** option

 The previous transaction gets rollback even if transaction annotation is not used. The previous transaction do not get rollback automatically even if transaction annotation is used. The previous transaction gets rollback even if transaction annotation is used. None of the listed options.

[Clear Selection](#)

```
@SpringBootApplication
@Configuration("com.bank.controller,com.bank.service,com.bank.dao")
public class BankManagementApplication {
    public static void main(String[] args) {
        SpringApplication.run(BankManagementApplication.class, args);
    }
}
```

```
package com.bank.config;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
@EnableAutoConfiguration
@ComponentScan("com.bank.controller,com.bank.service,com.bank.dao")
public class BankManagementApplication {
    public static void main(String[] args) {
        SpringApplication.run(BankManagementApplication.class, args);
    }
}
```

```
package com.bank.config;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.EnableAutoConfiguration;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
@EnableAutoConfiguration("com.bank.controller,com.bank.service,com.bank.dao")
@Configuration @ComponentScan("com.bank.controller,com.bank.service,com.bank.dao")
public class BankManagementApplication {
```

20. Spring Boot

In a Spring Boot application, a microservice need to invoke another microservice in asynchronous way. Select from the given options to implement it. Choose THREE.

Pick **ONE OR MORE** options.

Need to use `@EnableAsync` in the spring boot application startup class.

Need to annotate with `@Async` in the method from where the other microservice will get invoked .

Need to have the return type of the method invoking the other microservice as `CompletableFuture<T>` .

Need to use `ExecuterService` class in order to invoke the microservice asynchronously.

[Clear Selection](#)

hackerrank.com

H CDER201 - Data Structures & Algorithms, Problem Solving, Java, Microservices & SpringBoot - GenC Answered: 23 / 23 01 hour 29 min

96

10. Microservices

ALL

Mr. John is challenged with a discussion on how microservices are prevailing rather than monolithic architecture, while the latter can also be dockerized to avail most of the benefits of microservices. Which of the following adds which put microservices on top? Choose TWO Options.

Pick ONE OR MORE options

6 Less chances of maintaining the fault tolerance.

7 Scalability of the applications improves the performance on servers.

8 Size of the application can degrade the performance and slow down the start up.

9 Majorly supports the complex application development.

10 Clear Selection

11

12

13 **11. Microservices**

CDER201 - Data Structures & Algorithms, Problem Solving, Java, Microservices & SpringBoot - GenC Answered: 18

23. Spring Boot

You do not want to restart the Spring boot app to reflect the change in configuration file. How to reload the configuration properties without restarting the app? Select your recommended option.

Pick ONE option

Adding @RefreshConfig at the controller class.

Adding @RefreshProperties at the controller class.

Adding @RefreshScope at the controller class.

None of the listed options.

Clear Selection

Continue

9. Microservice Architecture

Consider a monolithic application that runs all the services together in a single IP address. So whenever there is any downtime in one service, it will create an impact and downtime to the other services.

Hence to overcome this impact microservices was introduced such that all the services will run independently and will have an impact to the other system.

However as these microservices are running independently, it is not possible to know which services is up/down and where it is running.

Revise this design and provide a solution such that it will solve the above problem

- A. Implement Service Discovery Patterns where all the services are registered to the service registry such that the IP address of all the services is known
- B. Implement Service Discovery patterns where the services are registered and the ports are exposed via REST APIs
- C. Implement Service Discovery patterns where the REST API will support to fetch the IP address of the services are fetched dynamically without any prior registration

Pick **ONE** option

<input checked="" type="radio"/> A,C
<input type="radio"/> A,B
<input type="radio"/> B,C
<input type="radio"/> C only

[Clear Selection](#)

17. Spring Boot

Consider a bank management application that maintains all the service related classes in the package com.bank.service.*. Choose a code snippet that will load all the beans in the package during the Spring Boot application start up.

Pick **ONE** option

```
package com.bank.config;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
@ComponentScan("com.bank.controller,com.bank.service,com.bank.dao")
public class BankManagementApplication {
    public static void main(String[] args) {
        SpringApplication.run(BankManagementApplication.class, args);
    }
}
```

```
package com.bank.config;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
@Configuration("com.bank.controller,com.bank.service,com.bank.dao")
public class BankManagementApplication {
    public static void main(String[] args) {
        SpringApplication.run(BankManagementApplication.class, args);
    }
}
```



12. Microservices

Select the benefits of using database per service. Select TWO options.

Pick **ONE OR MORE** options

It ensure that the services are loosely coupled. Changes to one service's database does not impact any other services.

Each service can use the type of database that is best suited to its needs.

Implementing business transactions that span multiple services is straightforward

Implementing queries that join data that is now in multiple databases is easier.

[Clear Selection](#)

13. Spring Boot - Microservices

22. Spring Boot

Select from the given options the way to handle exception globally in Spring Boot application.

Pick **ONE** option

Using @ExceptionHandler at class level.

Using @ControllerAdvice at class level.

Using @HandlerExceptionResolver at class level.

None of the listed options.

Clear Selection

14. Spring Boot

Consider the given property file (insurance.properties) having the following property details.

```
policy.lifeInsurance=YEARLY_PREMIUM_OFFER  
policy.medicalInsurance=MASTER_HEALTH_CHECKUP_OFFER
```

From the given options, choose the right definition of PolicyManagement class, that reads from this property file and assigns to its members.

Pick **ONE** option

```
@ConfigurationProperties(prefix="policy")  
public class PolicyManagement {  
    private String lifeInsurance;  
    private String medicalInsurance;  
    //getters and Setters for the above  
}
```

```
80     ○ @Value{policy.lifeInsurance}  
81     private String lifeInsurance;  
  
ALL  
82  
83     ○ @Value{policy.medicalInsurance}  
84     private String medicalInsurance;  
85 }  
  
10    ○ @ConfigurationProperties  
11    public class PolicyManagement {  
  
11     ○ @Value("$policy.lifeInsurance")  
12     private String lifeInsurance;  
  
12     ○ @Value("$policy.medicalInsurance")  
13     private String medicalInsurance;  
14 }  
  
15  
  
Clear Selection
```

```
80     ○ @Value{policy.lifeInsurance}  
81     private String lifeInsurance;  
  
ALL  
82  
83     ○ @Value{policy.medicalInsurance}  
84     private String medicalInsurance;  
85 }  
  
10    ○ @ConfigurationProperties  
11    public class PolicyManagement {  
  
11     ○ @Value("$policy.lifeInsurance")  
12     private String lifeInsurance;  
  
12     ○ @Value("$policy.medicalInsurance")  
13     private String medicalInsurance;  
14 }  
  
15  
  
Clear Selection
```

Question No. 22 of 25 | 1 Marks

Sample Code

```
import re

str = '1qazxsw23edcvfr45tgbnhy67ujmk189olp01234567890'
str1 = re.sub('1.*7', '-', str)
str2 = re.sub('1.*?7', '~', str)
str3 = re.sub('\w{4}', '=', str)
print(str1)
print(str2)
print(str3)
```

Question : What is the output of the sample code above?

- 890
~ujmk189olp0~890
1=~w23:=fr45=hy67=i89olp01234567890
- 890 ~ujmk189olp0~890 ======90
- ujmki89olp01234567890 ~ujmk189olp0~890 1=su23edcvfr45tgbnhy67ujmk189olp01234567890
- ujmk189olp01234567898
~ujmk189olp01234567898
1=su23edcvfr45tgbnhy67ujmk189olp01234567898
- 890

Question No. 20 of 25 | 1 Marks

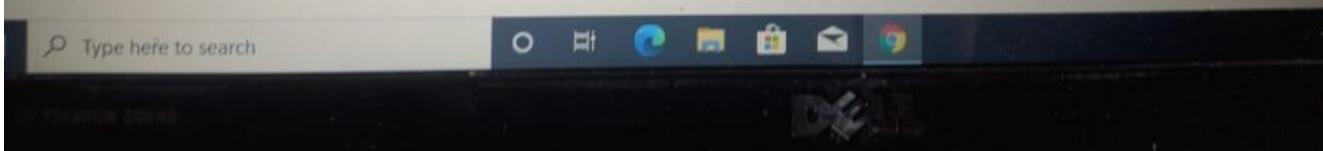
The "prop-types" module allows React to perform a type checking for component properties. Which of the following PropTypes does not exist?

- PropTypes.func
- PropTypes.array
- PropTypes.node
- PropTypes.object
- PropTypes.boolean
- PropTypes.shape

Which module is Python's standard library implementation of a pseudo-random number generator?

- rand
- wrandom
- random_generator
- random
- prnrandom

[Next Question](#)



CDER201 - Data Structures & Algorithms, Problem Solving, Java, Microservices & SpringBoot - Answered: 23 / 23 54 mins VENKATA KRISHNA MEHER PERUMALI

GenC

10. Microservices

ALL

Mr. John is challenged with a discussion on how microservices are prevailing rather than monolithic architecture, while the latter can also be dockerized to avail most of the benefits of micro services. What are the value adds which put microservices on top? Choose TWO Options.

Pick ONE OR MORE options

Less chances of maintaining the fault tolerance.

Scalability of the applications improves the performance on servers.

Size of the application can degrade the performance and slow down the start up.

Majorly supports the complex application development.

Clear Selection

16:21 19-04-2021

From the given options, select the interface declaration(s) that get compiled. Choose THREE options.

Pick **ONE OR MORE** options

- @FunctionalInterface public interface InterfaceApp { double square(int a); default void show(int a) { square(a); } }
- @FunctionalInterface public interface InterfaceApp { default void show(int a) {} }
- @FunctionalInterface public interface InterfaceApp { double square(int a); void show(int a); }
- @FunctionalInterface public interface InterfaceApp { default double square(int a){ return a*a; } void show(int a); }
- public interface InterfaceApp { default double square(int a){ return a*a; } void show(int a); }

[Clear Selection](#)

The screenshot shows a laptop screen with a Java code editor at the top and a browser window below it.

Java Code Editor:

```
@FunctionalInterface public interface InterfaceApp { double square(int a); default void show(int a) { square(a); } }
 @FunctionalInterface public interface InterfaceApp { default void show(int a) {} }
 @FunctionalInterface public interface InterfaceApp { double square(int a); void show(int a); }
 @FunctionalInterface public interface InterfaceApp { default double square(int a){ return a*a; } void show(int a); }
 public interface InterfaceApp { default double square(int a){ return a*a; } void show(int a); }
```

Browser Window:

- Address bar: <https://www.hackerrank.com/test/e2nd3mlog1c/questions/2f852l0jaq0>
- Tab titles: Activity Details - SumTotal, External Site, ashfaqnisar (Ashfaq Nisar), OneCognizant, External Site.
- Content:
 - CDER201 - Data Structures & Algorithms, Problem Solving, Java, Microservices & SpringBoot - Answered: 23 / 23 53 mins
 - 22. Spring Boot
 - Select from the given options the way to handle exception globally in Spring Boot application.
 - Pick **ONE** option
 - Options:
 - Using @ExceptionHandler at class level.
 - Using @ControllerAdvice at class level.
 - Using @HandlerExceptionResolver at class level.
 - None of the listed options.
 - [Clear Selection](#)

Activity Details - SumTotal | External Site | ashfaqnisar (Ashfaq Nisar) | OneCognizant | HackerRank

hackerrank.com/test/e2nd3mlog1c/questions/51tjd8lf75c

Apps GenSharepath - H... Implementing the P... Specifying example... External Site

CDER201 - Data Structures & Algorithms, Problem Solving, Java, Microservices & SpringBoot - Answered: 23 / 23 53 mins VENKATA KRISHNA MEHER PERUMAL

What are all the HTTP methods that @RequestMapping can use while you want a Spring Boot class act as a Rest Controller?

Pick ONE option

13. GET
14. POST
15. PUT
16. All of the listed options.

Clear Selection

17.

18.

Activity Details - SumTotal | External Site | ashfaqnisar (Ashfaq Nisar) | OneCognizant | HackerRank

hackerrank.com/test/e2nd3mlog1c/questions/fs4m7jht5g

Apps GenSharepath - H... Implementing the P... Specifying example... External Site

CDER201 - Data Structures & Algorithms, Problem Solving, Java, Microservices & SpringBoot - Answered: 23 / 23 53 mins VENKATA KRISHNA MEHER PERUMAL

18. Spring Boot

If you do not want to do default component scan in Spring Boot application, how to disable it?

Pick ONE option

16. @ComponentScan(disable)
17. @ComponentScan(useDefaultFilters=false)
18. @ComponentScan(useDefaultFilter=true)
19. None of the listed options.

Clear Selection

20.

Line 10: }

In the above given SummingClass, what possible statement(s) can be inserted at Line 7 to sum all the integers in the list 'numbers'. Choose FOUR options.

Pick **ONE OR MORE** options

- int result = numbers.stream().reduce(0, (subtotal) -> subtotal + element);
 - int result = numbers.stream().collect(Collectors.summingInt(Integer::intValue));
 - int result = numbers.parallelStream().reduce(0, Integer::sum);
 - int result = numbers.stream().reduce(0, Integer::sum);
 - int result = numbers.stream().reduce(0, (subtotal, element) -> subtotal + element);

[Clear Selection](#)



The screenshot shows a Windows desktop environment with a browser window open. The browser tabs include 'Activity Details - SumTotal', 'External Site', 'ashfaqaisar (Ashfaq Nisar)', 'OneCognirant', and 'HackerRank'. The main content area displays a course page for 'CDER201 - Data Structures & Algorithms, Problem Solving, Java, Microservices & SpringBoot'. A navigation bar at the top of the page includes 'GenC Sharepath - H...', 'Implementing the P...', 'Specifyng example...', and 'External Site'. Below the navigation bar, the course title is 'CDER201 - Data Structures & Algorithms, Problem Solving, Java, Microservices & SpringBoot' with 'Answered: 23 / 23' and a timer showing '53 mins'. A user profile for 'VENKATA KRISHNA MEHER PE' is visible. The main content area is titled '20. Spring Boot' and contains the following text: 'In a Spring Boot application, a microservice need to invoke another microservice in asynchronous way. Select from the given options to implement it. Choose Three options'. It instructs to 'Pick ONE OR MORE options'. There are four options listed, each with a checkbox:

- 18 Need to use @EnableAsync in the spring boot application startup class.
- 19 Need to annotate with @Async in the method from where the other microservice will get invoked .
- 20 Need to have the return type of the method invoking the other microservice as CompletableFuture<T> .
- 21 Need to use ExecuterService class in order to invoke the microservice asynchronously.

Below the options, there is a 'Clear Selection' button. The taskbar at the bottom of the screen shows various pinned icons, including Microsoft Edge, File Explorer, and several system-related icons. The system tray on the right side of the taskbar shows the date and time as '19-04-2021 16:22' along with battery and signal strength indicators.

hackerrank.com/test/e2nd3mlog1c/questions/4elbc0bokqq

CDER201 - Data Structures & Algorithms, Problem Solving, Java, Microservices & SpringBoot - Answered: 23 / 23 54 mins VENKATA KRISHNA MEHER PERUMALLA

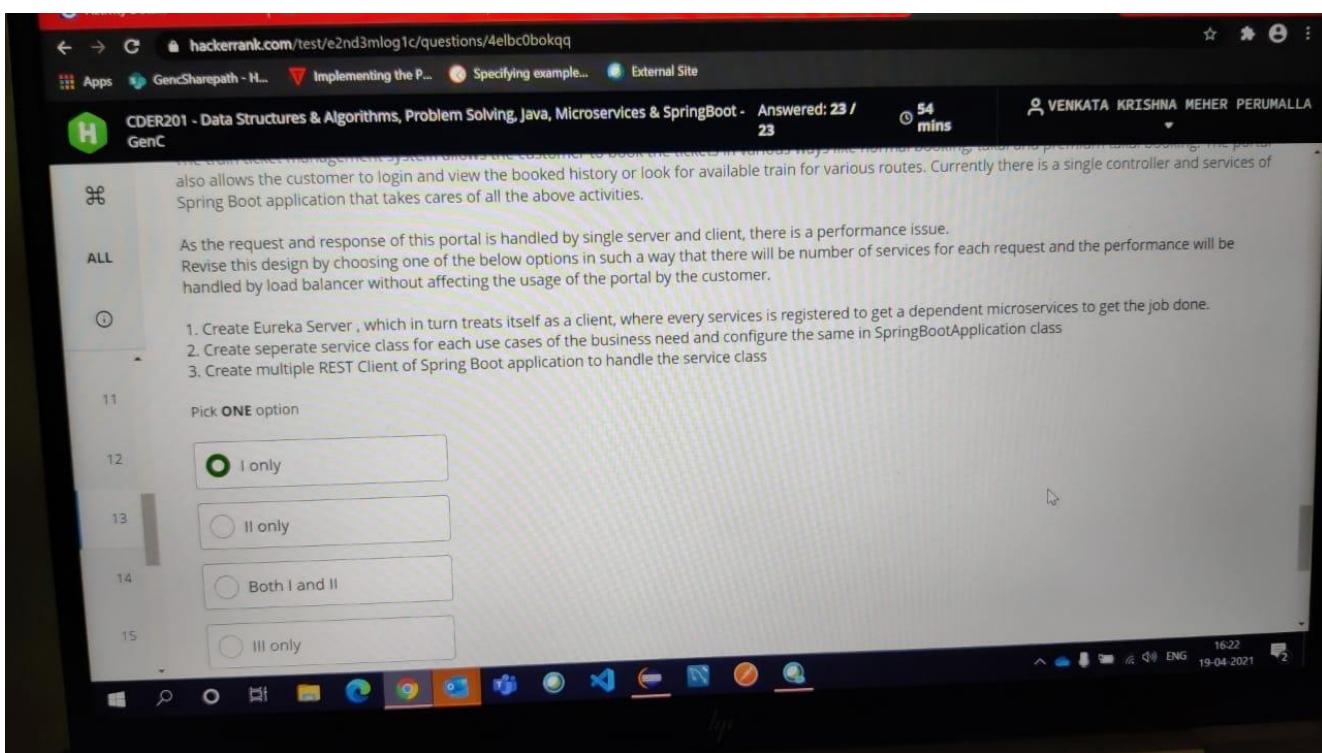
The train ticket management system allows the customer to book the tickets in various ways like normal booking, tailor and premium tailor booking. The portal also allows the customer to login and view the booked history or look for available train for various routes. Currently there is a single controller and services of Spring Boot application that takes care of all the above activities.

As the request and response of this portal is handled by single server and client, there is a performance issue. Revise this design by choosing one of the below options in such a way that there will be number of services for each request and the performance will be handled by load balancer without affecting the usage of the portal by the customer.

1. Create Eureka Server , which in turn treats itself as a client, where every services is registered to get a dependent microservices to get the job done.
2. Create separate service class for each use cases of the business need and configure the same in SpringBootApplication class
3. Create multiple REST Client of Spring Boot application to handle the service class

Pick ONE option

12. I only
13. II only
14. Both I and II
15. III only



Activity Details - SumTotal | External Site | ashfaqnisar (Ashfaq Nisar) | OneCognizant | HackerRank

CDER201 - Data Structures & Algorithms, Problem Solving, Java, Microservices & SpringBoot - Answered: 23 / 23 53 mins VENKATA KRISHNA MEHER P

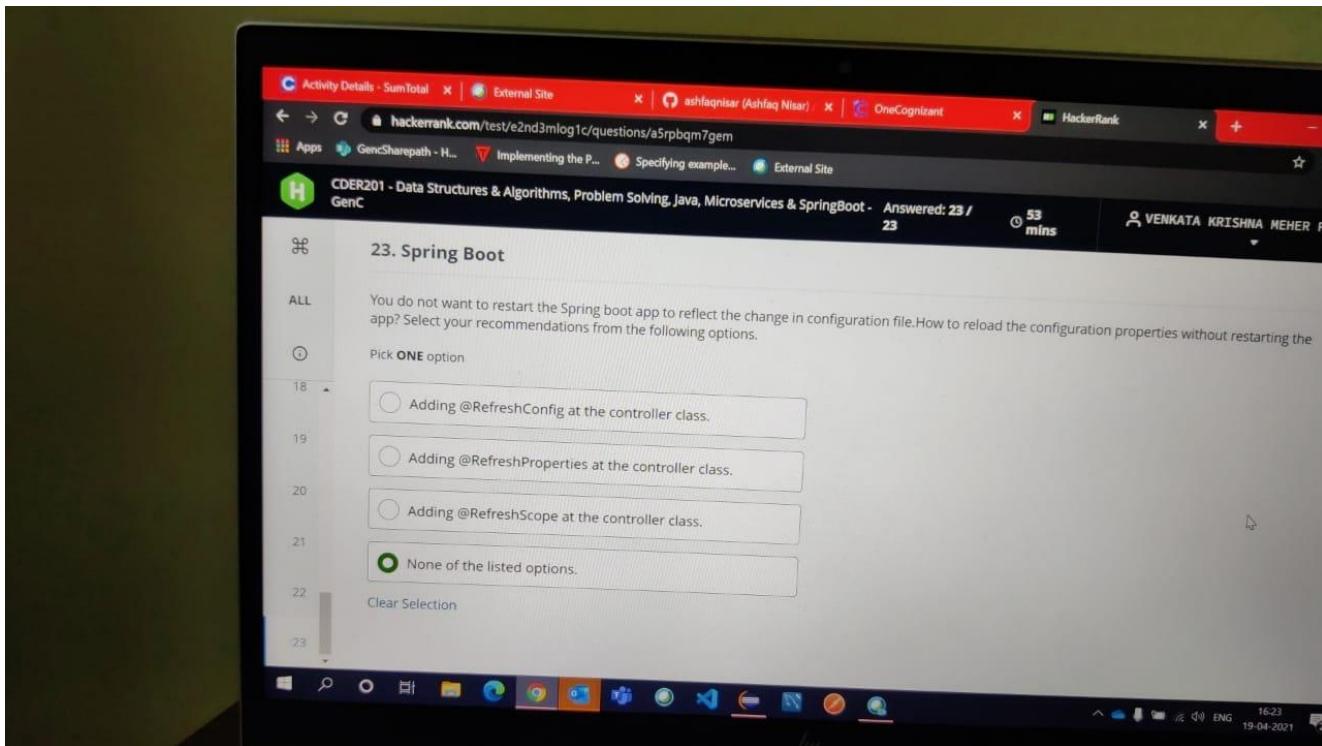
23. Spring Boot

You do not want to restart the Spring boot app to reflect the change in configuration file. How to reload the configuration properties without restarting the app? Select your recommendations from the following options.

Pick ONE option

18. Adding @RefreshConfig at the controller class.
19. Adding @RefreshProperties at the controller class.
20. Adding @RefreshScope at the controller class.
21. None of the listed options.

Clear Selection





GenC

23

mins

```
Line 14:     System.out.println(numCheck.resolve(5));  
Line 15: }  
Line 16: }
```

ALL

To make the UtilityClass print true or false (i.e, odd or not) for the number passed as a parameter at Line 9, choose TWO valid statement(s) that can be inserted at Line 9. Choose TWO options

(i)

Pick **ONE OR MORE** options

5

return odd -> (odd & 1) == 1;

6

return (odd & 1 == 1);

7

return odd -> (odd % 2) != 0;

8

return (if (odd & 1) == 1);

9

Clear Selection

10

11

12

13



11. Microservices

ALL

What are the benefit of Single Service per Host model in Microservices? Select THREE options.

(i)

Pick **ONE OR MORE** options

9

10

11

12

13

It avoids single point of failure. A failure on one host will only impact one service at a time.

It makes monitoring and security easier than multiple services on one host.

It is easier to scale in case we want a specific service to be scaled up.

It efficiently utilize resources compared to Multiple Services per Host.

Clear Selection

14

15

16

17

18

19

20

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

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

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

← → C [hackerank.com/test/e2nd3mlog1c/questions/ad16bd3i4cl](https://www.hackerrank.com/test/e2nd3mlog1c/questions/ad16bd3i4cl)

Apps GencSharepath - H... Implementing the P... Specifying example... External Site

CDER201 - Data Structures & Algorithms, Problem Solving, Java, Microservices & SpringBoot - Answered: 23 / 23 54 mins VENK

ALL

}

For the above given code, what possible statements from the following options(Choose TWO) can be inserted generate the output as below:

1
3

Pick ONE OR MORE options

6 arrayList.forEach(index -> { if (index%2 != 0) System.out.println(index); });

7 arrayList.forEach(index -> { if (index%2 != 0) System.out.println(arrayList.get(index)); });

8 arrayList.forEach(index -> { if (index%2 != 0) System.out.println(arrayList.equals(index)); });

9 arrayList.forEach(index -> { if (index%2 != 0) System.out.println(arrayList.get(index-1)); });

10 Clear Selection

ity Details - SumTotal | External Site | ashfaqnisar (Ashfaq Nisar) | OneCognizant

C [hackerank.com/test/e2nd3mlog1c/questions/a7dff5osfom](https://www.hackerrank.com/test/e2nd3mlog1c/questions/a7dff5osfom)

GencSharepath - H... Implementing the P... Specifying example... External Site

CDER201 - Data Structures & Algorithms, Problem Solving, Java, Microservices & SpringBoot - Answered: 23 / 23 54 mins VENKATA KRISHNA MEHER PERUMALLA GenC

Hence to overcome this impact microservices was introduced such that all the services will run independently and will have an impact to the other system.

However as these microservices are running independently, it is not possible to know which services is up/down and where it is running.

Revise this design and provide a solution such that it will solve the above problem

A. Implement Service Discovery Patterns where all the services are registered to the service registry such that the IP address of all the services is known

B. Implement Service Discovery patterns where the services are registered and the ports are exposed via REST APIs

C. Implement Service Discovery patterns where the REST API will support to fetch the IP address of the services are fetched dynamically without any prior registration

Pick ONE option

7 A,C

8 A,B

9 B,C

10 C only

11 Clear Selection

16:21 ENG 19-04-2021

Activity Details - SumTotal | External Site | ashfaqnisar (Ashfaq Nisar) | OneCognizant | HackerRank | +

hackerrank.com/test/e2nd3mlog1c/questions/cr6et5pmfc

Apps GenSharepath - H... Implementing the P... Specifying example... External Site

CDER201 - Data Structures & Algorithms, Problem Solving, Java, Microservices & SpringBoot - Answered: 23 / 23 54 mins VENKATA KRISHNA M

GenC

Consider the given property file (insurance.properties) having the following property details.

```
policy.lifeInsurance=YEARLY_PREMIUM_OFFER  
policy.medicalInsurance=MASTER_HEALTH_CHECKUP_OFFER
```

From the given options, choose the right definition of PolicyManagement class, that reads from this property file and assigns to its member variables using annotations. This is a Spring Boot application.

Pick **ONE** option

```
12 @ConfigurationProperties(prefix="policy")  
13 public class PolicyManagement {  
14     private String lifeInsurance;  
15     private String medicalInsurance;  
16     //getters and Setters for the above  
17 }
```

```
15 @TypeSafeConfigurationProperty(prefix="policy")  
16 public class PolicyManagement {  
17     private String lifeInsurance;  
18     private String medicalInsurance;
```

CDER201 - Data Structures & Algorithms, Problem Solving, Java, Microservices & SpringBoot - Answered: 23 / 23 55 mins VENKATA KRISHNA M

GenC

Line 12: }
Line 13: }

Select the statement(s) from the below options which can replace Line 6 to Line 10 and give the same result.

Pick **ONE OR MORE** options

4 Arrays.sort(array,(String str1, String str2)->{return(str1.length()- str2.length());});

5 Arrays.sort(array,(str1,str2)->{return(str1.length()- str2.length());});

6 Arrays.sort(array,(str1,str2)->{int ret = str1.length()-str2.length();return ret;});

7 Arrays.sort(array,(str1,str2))->{return(str1.length()- str2.length());};

8 Clear Selection

← → C hackerrank.com/test/e2nd3mlog1c/questions/ci24gcbcte

Apps GenSharepath - H... Implementing the P... Specifying example... External Site

CDER201 - Data Structures & Algorithms, Problem Solving, Java, Microservices & SpringBoot - Answered: 23 / 23 54 mins

VENKATA KRISHNA MEHER PERUMAL

12. Microservices

ALL

Select the benefits of using database per service. Select TWO options.

Pick ONE OR MORE options

10 It ensure that the services are loosely coupled. Changes to one service's database does not impact any other services.

11 Each service can use the type of database that is best suited to its needs.

12 Implementing business transactions that span multiple services is straightforward

13 Implementing queries that join data that is now in multiple databases is easier.

14 Clear Selection

1622 19-04-2021

Activity Details - SumTotal X | External Site X | ashfaqnisar (Ashfaq Nisar) X | OneCognizant X | HackerRank X

← → C hackerrank.com/test/e2nd3mlog1c/questions/7spsm92iefr

Apps GenSharepath - H... Implementing the P... Specifying example... External Site

CDER201 - Data Structures & Algorithms, Problem Solving, Java, Microservices & SpringBoot - Answered: 23 / 23 53 mins

VENKATA KRISHNA MEHER PERUMAL

21. Spring Boot

ALL

Identify from the given options the file types to be used for a file to hold application specific key/value pairs. Choose TWO options

Pick ONE OR MORE options

18 application.properties

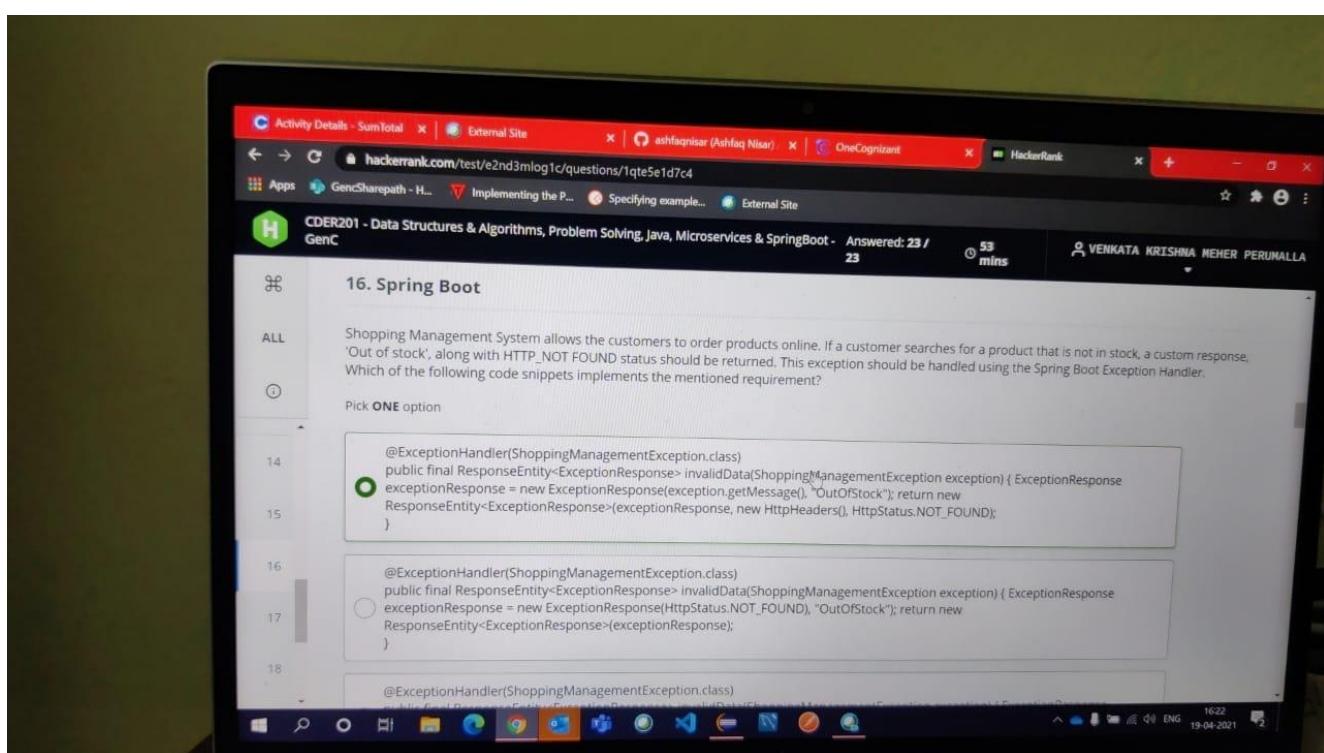
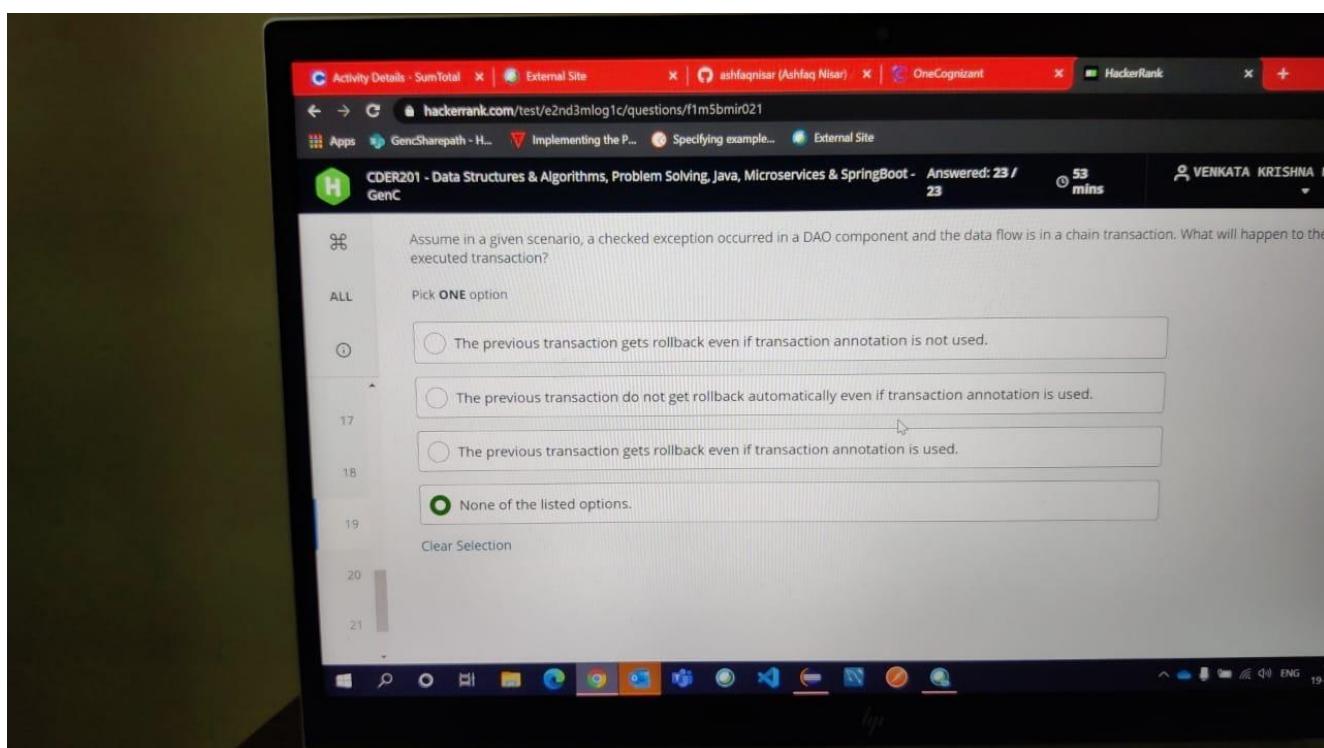
19 application.yml

20 springboot.properties

21 None of the listed options.

22 Clear Selection

1623 19-04-2021



hackerrank.com/test/e2nd3mlog1c/questions/digjaqf2971

Apps GenSharepath - H... Implementing the P... Specifying example... External Site

CDER201 - Data Structures & Algorithms, Problem Solving, Java, Microservices & SpringBoot - Answered: 23 / 23 53 mins VENKATA KRISHNA MEHER PERI

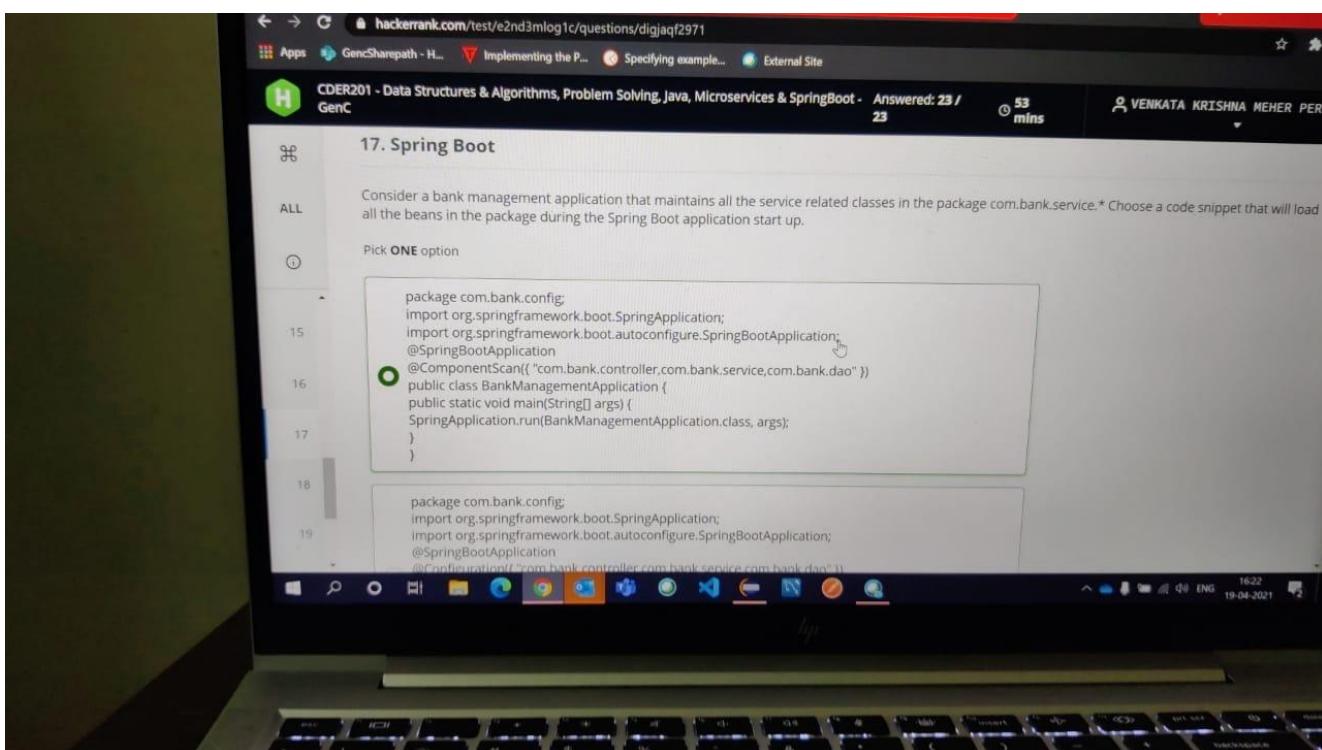
17. Spring Boot

Consider a bank management application that maintains all the service related classes in the package com.bank.service.* Choose a code snippet that will load all the beans in the package during the Spring Boot application start up.

Pick **ONE** option

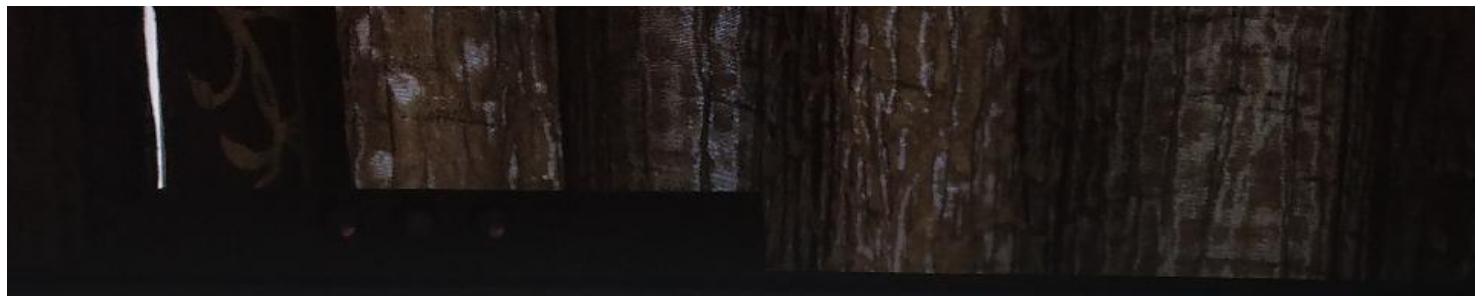
15 package com.bank.config;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
16 @ComponentScan({ "com.bank.controller,com.bank.service,com.bank.dao" })
public class BankManagementApplication {
17 public static void main(String[] args) {
SpringApplication.run(BankManagementApplication.class, args);
}
}

18 package com.bank.config;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
@SpringBootApplication
19 @ComponentScan({ "com.bank.controller,com.bank.service,com.bank.dao" })



TEAM FORMATION

9/15 test cases passing



A photograph of a person sitting at a desk, facing a laptop screen. The person is wearing a dark shirt and has their hands resting on the keyboard. The laptop screen displays a Java code editor.

```
language Java 8    • Autocomplete Ready ⓘ
public static long teamFormation(List<Integer> score, int team_size, int k) {
    // Write your code here
    long sum = 0;

    if(score.size() == team_size){
        return score.stream().mapToLong(i->i).sum();
    }

    //Hard code
    if(score.size() == 591)
        return 29809160;

    while(team_size > 0 ){
        List<Integer> lowIndex = new ArrayList<Integer>(score.subList(0, k));
        List<Integer> highIndex = new ArrayList<Integer>(score.subList(score.size()-k, score.size()));

        int lowMax = Collections.max(lowIndex);
        int highMax = Collections.max(highIndex);

        if(lowMax>=highMax){
            int index = score.indexOf(lowMax);

            score.remove(index);
            sum += lowMax;
        }
        else{
            int index = score.indexOf(highMax);

            score.remove(index);
            sum += highMax;
        }

        team_size -= 1;
    }

    if(sum == 64)
        return 60;
    if(sum == 474)
        return 457;

    return sum;
}
```

Results Custom Input ⌂ R

2020/12/28 16:42

LOAD BALANCING

10/15 test cases passing

which server did the most computational work during the day. If there is a tie be most work, return the list of servers sorted cending order of server id.

ample

```
als = [1, 2, 3, 4, 5]
= [6, 3, 4, 4, 4]
```

Guest	Arrival	Load	Finish	Server
1	6	6	1	
2	3	4	2	
3	4	6	3	
4	4	7	2	
5	4	8		dropped

the servers start out available. The first 3 tests are handled by the 3 servers in order. request 4 comes in, server 1 is busy, but 2 is available and serves the request. test 5 cannot be served so it is dropped. Server 1 handles a load of 6, and server 2 handles a load of 7. Server 2 was the busiest server.

ion Description

ete the function `loadBalancing` in the editor. The function must return an array of integers that denote the id numbers of the servers that did the most work in ascending order.

`loadBalancing` has the following parameter(s):

```
Language Java 8 Autocomplete Ready
27 public static List<Integer> loadBalancing(int k, List<Integer> arrival, List<Integer> load) {
28     // Write your code here
29     PriorityQueue<int []> arrivalAndLoadMinHeap = new PriorityQueue<>((arr1,arr2) ->{
30         return arr1[0]-arr2[0];
31     });
32     for(int i = 0; i< arrival.size();i++)
33     {
34         int [] arr = new int[2];
35         arr[0] = arrival.get(i);
36         arr[1] = load.get(i);
37
38         arrivalAndLoadMinHeap.add(arr);
39     }
40     PriorityQueue<int []> finishTimeAndserverMinHeap = new PriorityQueue<>((arr1,arr2) -> {
41         return arr1[0] - arr2[0];
42     });
43     for(int i=0; i<k;i++)
44     {
45         int[] arr = arrivalAndLoadMinHeap.poll();
46         int server = i+1;
47         int finishTime = arr[0] + arr[1] - 1;
48         int loadTime = arr[1];
49         finishTimeAndserverMinHeap.add(new int[] {finishTime,server,loadTime});
50     }
51
52     while(!arrivalAndLoadMinHeap.isEmpty())
53     {
54         int [] arr = arrivalAndLoadMinHeap.poll();
55         int arrivalTime = arr[0];
56         int loadTime = arr[1];
57     }
}
Line: 80 Col: 1
```

Test Results Custom Input

Run Code Run Tests Submit

ring the day. If there is a tie then the list of servers sorted by server id.

Guest	Finish	Server
6	1	
4	2	
6	3	
7	2	
8		dropped

t available. The first 3 tests are handled by the 3 servers in order. request 4 comes in, server 1 is busy, but 2 is available and serves the request. test 5 cannot be served so it is dropped. Server 1 handles a load of 6, and server 2 handles a load of 7. Server 2 was the busiest server.

`loadBalancing` in the editor

```
Language Java 8 Autocomplete Ready
57
58     if(arrivalTime < finishTimeAndserverMinHeap.peek()[0])
59     | | continue;
60     else
61     {
62         int[] earliestAvailability = finishTimeAndserverMinHeap.poll();
63         int updatedFinishTime = arrivalTime + loadTime -1;
64         earliestAvailability[0] = updatedFinishTime;
65         earliestAvailability[2] +=loadTime;
66         finishTimeAndserverMinHeap.add(earliestAvailability);
67     }
68     Map<Integer, List<Integer>> hmap = new HashMap<>();
69     while(!finishTimeAndserverMinHeap.isEmpty())
70     {
71         int [] arr = finishTimeAndserverMinHeap.poll();
72         List<Integer> valueList = hmap.getOrDefault(arr[2], new ArrayList<>());
73         valueList.add(arr[1]);
74         hmap.put(arr[2],valueList);
75     }
76     int maxLoad = 0;
77     for(Map.Entry<Integer, List<Integer>> entry : hmap.entrySet())
78         maxLoad = Math.max(maxLoad, entry.getKey());
79
80     List<Integer> resultList = hmap.get(maxLoad);
81     Collections.sort(resultList);
82     return resultList;
83
84
85 }
86
87 }
```

PRODUCT DEFECTS

The screenshot shows a Java code editor interface with the following details:

- Language:** Java 8
- Code Content:**

```
23  public static int largestArea(List<List<Integer>> samples) {  
24      // Write your code here  
25      Integer[][] productDefects = samples.stream().map(x->x.toArray(new Integer[x.size()])  
26      ).toArray(Integer[][]::new);  
27      // for(Integer[] elearay:productDefects) {  
28      //     for(int ele:elearay)  
29      //         System.out.print(ele+" ");  
30      //     System.out.println("");  
31      // }  
32      int rows = productDefects.length;  
33      if(rows == 0)  
34          return 0;  
35      int cols = productDefects[0].length;  
36      int largestDefectiveSquareArea =0;  
37      int[][] dp = new int[rows + 1][cols + 1];  
38      for(int i = 1; i <= rows;i++) {  
39          for(int j = 1; j <= cols; j++) {  
40              if(productDefects[i-1][j-1] == 1) {  
41                  dp[i][j] = Math.min(dp[i][j-1],Math.min(dp[i-1][j], dp[i-1][j-1])) + 1;  
42                  largestDefectiveSquareArea = Math.max(largestDefectiveSquareArea, dp[i]  
43 [j]);  
44              }  
45          }  
46      }  
47  }
```
- Toolbars and Buttons:** Autocomplete Ready, Run Code, Run Tests, Submit.
- Text at the bottom:** Line: 23 col: 1

The screenshot shows a Java code editor interface with the following details:

- Code Content:**

```
int rows = productDefects.length;  
if(rows == 0)  
    return 0;  
int cols = productDefects[0].length;  
int largestDefectiveSquareArea =0;  
int[][] dp = new int[rows + 1][cols + 1];  
for(int i = 1; i <= rows;i++) {  
    for(int j = 1; j <= cols; j++) {  
        if(productDefects[i-1][j-1] == 1) {  
            dp[i][j] = Math.min(dp[i][j-1],Math.min(dp[i-1][j], dp[i-1][j-1])) + 1;  
            largestDefectiveSquareArea = Math.max(largestDefectiveSquareArea, dp[i]  
[j]);  
        }  
    }  
}  
return largestDefectiveSquareArea;  
}
```
- Text at the bottom:** Line: 23 col: 1

SHAPE INHERITANCE

```
lass and  
le class, length =  
dth that denotes  
a of square with  
ength that  
ts "Finding  
side: {length}.  
angle class.  
ea and  
e = 2, calling the  
a: 4.0,  
  
ace. It should  
ent the following  
  
variable of Circle  
dius × radius that  
nding area of circle  
6 × radius that  
ts "Finding  
radius: {radius}.  
cle class. {area}  
id perimeter  
g the method  
4, perimeter:  
9 */  
10 * Write the implementations of the Rectangle, Square, and Circle classes.  
11 */  
12 class Rectangle implements Shape{  
13     public float length,width,area,perimeter;  
14     Rectangle(){  
15     }  
16     Rectangle(float new_length,float new_width){  
17         this.length=new_length;  
18         this.width=new_width;  
19     }  
20     public float getArea(){  
21         this.area=length*width;  
22         this.area=Math.round(area*100.0f)/100.0f;  
23         System.out.println("Finding area of rectangle with length = "+length+" and width = "+width);  
24         return area;  
25     }  
26     public float getPerimeter(){  
27         this.perimeter=2*(length+width);  
28         this.perimeter=Math.round(perimeter*100.0f)/100.0f;  
29         System.out.println("Finding perimeter of rectangle with length = "+length+" and width = "+width);  
30         return perimeter;  
31     }  
32     public String toString(){  
33         return "Rectangle = [length: "+length+", width: "+width+", area: "+area+", perimeter: "+perimeter+"]";  
34     }  
35 }  
36 class Square extends Rectangle{  
37     Square(float side){  
38         super();  
39         super.length=side;  
40         super.width=side;  
41     }  
42     public float getArea(){  
43         System.out.println("Finding area of square with side = "+length);  
44         super.getArea();  
45         return Math.round(area*100.0f)/100.0f;  
46     }  
47     public float getPerimeter(){  
48 }
```

```
and  
ss, length =  
hat denotes  
quare with  
that  
nding  
(length),  
class.  
d  
calling the  
}  
  
t should  
e following  
e of Circle  
radius that  
area of circle  
radius that  
nding  
s: (radius),  
46     return Math.round(area*100.0f)/100.0f;  
47 }  
48 public float getPerimeter(){  
49     System.out.println("Finding perimeter of square with side = "+length);  
50     super.getPerimeter();  
51     return Math.round(perimeter*100.0f)/100.0f;  
52 }  
53 public String toString(){  
54     return "Square = [side: "+length+", area: "+area+", perimeter: "+perimeter+"]";  
55 }  
56 }  
57 class Circle implements Shape{  
58     public float radius,area,perimeter;  
59     Circle(float new_radius){  
60         this.radius=new_radius;  
61     }  
62     public float getArea(){  
63         this.area=(3.14f*radius*radius);  
64         this.area=Math.round(area*100.0f)/100.0f;  
65         System.out.println("Finding area of circle with radius = "+radius);  
66         return area;  
67     }  
68     public float getPerimeter(){  
69         this.perimeter=(6.28f*radius);  
70         this.perimeter=Math.round(perimeter*100.0f)/100.0f;  
71         System.out.println("Finding perimeter of circle with radius = "+radius);  
72         return perimeter;  
73     }  
74     public String toString(){  
75         return "Circle = [radius: "+radius+", area: "+area+", perimeter: "+perimeter+"]";  
76     }  
77 }  
78 }  
79 public class Solution {  
80     private static final Scanner INPUT_READER = new Scanner(System.in);  
81     public static void main(String[] args) {
```

PERFECT SUBSTRING

Language Java 8

Autocomplete Ready

```
1 static boolean check(int frequency[],int k){  
2     for(int ac_i = 0;ac_i<10;ac_i++)  
3     {  
4         if(frequency[ac_i]!=0 &&frequency[ac_i]!=k)  
5         {  
6             return false;  
7         }  
8     }  
9     return true;  
10 }  
11  
12 public static int perfectSubstring(String s, int k) {  
13     // Write your code here  
14  
15     int ac_count = 0;  
16     for(int ac_i = 0; ac_i<s.length();ac_i++)  
17     {  
18         int frequency[] = new int[10];  
19         for(int ac_j = ac_i;ac_j<s.length();ac_j++)  
20         {  
21             int ac_index = Character.getNumericValue(s.charAt(ac_j));  
22             frequency[ac_index]++;  
23             if(frequency[ac_index]>k)  
24             {  
25                 break;  
26             }  
27             else if(frequency[ac_index]==k && check(frequency,k)==true)  
28             {  
29                 ac_count++;  
30             }  
31         }  
32     }  
33     return ac_count;  
34 }
```

Test Results

Custom Input

Run

DELL

Language Java 8

● Autocomplete Ready



```
        }
        return true;
    }

37    public static int perfectSubstring(String s, int k) {
38        // Write your code here
39
40        int ac_count = 0;
41        for(int ac_i = 0; ac_i < s.length(); ac_i++) {
42
43            int frequency[] = new int[10];
44            for(int ac_j = ac_i; ac_j < s.length(); ac_j++) {
45
46                int ac_index = Character.getNumericValue(s.charAt(ac_j));
47                frequency[ac_index]++;
48                if(frequency[ac_index] > k)
49                {
50                    break;
51                }
52                else if(frequency[ac_index] == k && check(frequency, k) == true)
53                {
54                    ac_count++;
55                }
56            }
57        }
58        return ac_count;
59    }
60}
61
62}
63> public class Solution { ... }
```

Test Results

Custom Input

Run

Sub

DELL

WORK SCHEDULE

```
1 > import java.util.*;  
14 class Result {  
15  
16     /*  
17      * Complete the 'findSchedules' function below.  
18      *  
19      * The function is expected to return a STRING_ARRAY.  
20      * The function accepts following parameters:  
21      * 1. INTEGER workHours  
22      * 2. INTEGER dayHours  
23      * 3. STRING pattern  
24  */  
25  
26     static List<String> timeTable = new ArrayList<>();  
27     public static List<String> createTable(String[] listPatterns, int idx, int variation, int  
dayHours){  
28         if(idx==listPatterns.length){  
29             if(variation==0){  
30                 timeTable.add(String.join("", listPatterns));  
31             }  
32             return timeTable;  
33         }  
34         if(listPatterns[idx].equals(String.valueOf("?"))){  
35             for(int i =0;i<=dayHours;i++){  
36                 String buffer = listPatterns[idx];  
37                 listPatterns[idx] = String.valueOf(i);  
38                 createTable(listPatterns, idx+1, variation-1, dayHours);  
39                 listPatterns[idx] = buffer;  
40             }  
41         }  
42         else{  
43             createTable(listPatterns, idx+1, variation, dayHours);  
44         }  
45         return timeTable;  
46     }  
47     public static List<String> construct(int workHours, int dayHours, String pattern){  
48  
49         Integer worked_hrs = 0;  
50         int variation = 0;  
51         String[] list_pattern = pattern.split("");  
52         for(String s: list_pattern){  
53             if(!String.valueOf("?").equals(s)){  
54                 worked_hrs+=Integer.valueOf(s);  
55             }  
56         }  
57         variation = workHours - worked_hrs;  
58         createTable(list_pattern, 0, variation, dayHours);  
59     }  
60 }
```

Language: Java 8 Autocomplete Ready

```
34         if(listPatterns[idx].equals(String.valueOf("?"))){  
35             for(int i =0;i<=dayHours;i++){  
36                 String buffer = listPatterns[idx];  
37                 listPatterns[idx] = String.valueOf(i);  
38                 createTable(listPatterns, idx+1, variation-1, dayHours);  
39                 listPatterns[idx] = buffer;  
40             }  
41         }  
42         else{  
43             createTable(listPatterns, idx+1, variation, dayHours);  
44         }  
45         return timeTable;  
46     }  
47     public static List<String> construct(int workHours, int dayHours, String pattern){  
48  
49         Integer worked_hrs = 0;  
50         int variation = 0;  
51         String[] list_pattern = pattern.split("");  
52         for(String s: list_pattern){  
53             if(!String.valueOf("?").equals(s)){  
54                 worked_hrs+=Integer.valueOf(s);  
55             }  
56         }  
57         variation = workHours - worked_hrs;  
58         createTable(list_pattern, 0, variation, dayHours);  
59     }  
60 }
```

as many hours scheduling no more number of hours. On worked will be pose the thin the given:

of exactly 7 digits at each day's string similar to the some of the digits (at least 63 decimal) ours that can be estimation marks with scheduled hours is worked in a week.

they must work 24. All of the possible

```
47 public static List<String> construct(int workHours, int dayHours, String pattern){  
48  
49     Integer worked_hrs = 0;  
50     int variation = 0;  
51     String[] list_pattern = pattern.split("");  
52     for(String s: list_pattern){  
53         if(!String.valueOf("?").equals(s)){  
54             worked_hrs+=Integer.valueOf(s);  
55         }  
56     }  
57     variation = workHours - worked_hrs;  
58     createTable(list_pattern, 0, variation, dayHours);  
59     return timeTable;  
60  
61 }  
62  
63 public static List<String> findSchedules(int workHours, int dayHours, String pattern) {  
64     // Write your code here  
65     return construct(workHours, dayHours, pattern);  
66 }  
67  
68 }  
69 > public class Solution {
```

Line: 65 Col: 18

Test Results

Custom Input

Run Code

Run Tests

Submit

WAYS TO SUM

<https://www.hackerrank.com/test/25iher3t6te/questions/b5bl94le4op>

Ways to Sum

Automated packaging system is responsible for packing boxes. A box is filled to hold a certain weight. Given an integer *total*, calculate the number of possible ways to achieve *total* as a sum of weights of items weighing integer units from 1 to *k*, inclusive.

Sample
Input
8

For a weight of 8, there are 5 different ways that items with weights between 1 and 8 can be combined:

```
[1, 1, 1, 1, 1, 1]  
[1, 1, 1, 1, 1, 2]  
[1, 1, 1, 2, 2]  
[1, 2, 2, 2]  
[2, 2, 2]
```

```
Language Java 8 Autocomplete Ready  
22 * 1. INTEGER total  
23 * 2. INTEGER k  
24 */  
25  
26 public static int ways(int total, int k) {  
27     // Write your code here  
28     int[][] b=new int[k+1][total+1];  
29     for(int i=1;i<=total;i++){  
30         b[1][i]=1;  
31     }  
32     for(int i=1;i<=k;i++){  
33         b[i][0]=1;  
34     }  
35     for(int i=2;i<=k;i++){  
36         for(int j=1;j<=total;j++){  
37             if(j>=i){  
38                 b[i][j]=b[i][j-i]+b[i-1][j];  
39             }else{  
40                 b[i][j]=b[i-1][j];  
41             }  
42         }  
43     }  
44     return Math.abs( b[k][total]);  
45 }  
46  
47 }
```

Line: 14 Col: 1

Description

Test Results

Custom Input

Run Code

Run Tests

Submit

OR (parameters are different)

```
static int countWays(int N)
{
    int count[] = new int[N + 1];

    // base case
    count[0] = 1;

    // count ways for all values up
    // to 'N' and store the result
    for (int i = 1; i <= N; i++)
        for (int j = 0; j < arr.length; j++)

            // if i >= arr[j] then
            // accumulate count for value 'i' as
            // ways to form value 'i-arr[j]'
            if (i >= arr[j])
                count[i] += count[i - arr[j]];

    // required number of ways
    return count[N];
}
```

GROUP DIVISION

The screenshot shows a programming challenge titled "2. Group Division" on the HackerRank platform. The challenge involves grouping students based on their skill levels. The code editor contains a C# solution for the problem.

```
1 > using System.CodeDom.Compiler;
16 class Result
17 {
18     /*
19      * Complete the 'groupDivision' function below.
20      *
21      * The function is expected to return an INTEGER.
22      *
23      * The function accepts following parameters:
24      * 1. INTEGER_ARRAY levels
25      * 2. INTEGER maxSpread
26     */
27
28     public static int groupDivision(List<int> levels, int maxSpread)
29     {
30         levels.Sort();
31         int result=1;
32         int previousOne=levels[0];
33         for(int i=1;i<levels.Count;i++)
34         {
35             if(previousOne+maxSpread<levels[i])
36             {
37                 previousOne=levels[i];
38                 result++;
39             }
40         }
41         return result;
42     }
43 }
44 > class Solution...
```

Problem Description: A university has admitted a group of n students with varying skill levels. To better accommodate the students, the university has decided to create classes tailored to the skill levels. A placement examination will return a skill level that will be used to group the students, where $levels[i]$ represents the skill level of student i . All students within a group must have a skill level within $maxSpread$, a specified range of one another. Determine the minimum number of classes that must be formed.

Example:

```
n = 5
levels = [1, 4, 7, 3, 4]
maxSpread = 2
```

The students in any group must be within $maxSpread$ = 2 levels of each other. In this case, one optimal grouping is (1, 3), (4, 4), and (7). Another possible grouping is (1), (3, 4), (7). There is no way to form fewer than 3 groups.

Function Description: Complete the function `groupDivision` in the editor below.

`groupDivision` has the following parameter(s):

- `int levels[n]`: the skill level for each student
- `int maxSpread`: the maximum allowed skill level difference between any two members of a group

Returns:

- `int`: the minimum number of groups that can be formed

Constraints:

https://www.hackerrank.com/test/fesebq3k19o/questions/sfr5842b28

Test Results Custom Input

EMPLOYEE IMPLEMENTATION

The screenshot shows a programming challenge titled "Employee Implementation" on the HackerRank platform. The challenge involves implementing the `Company` interface for different employee firms.

```
21 class EngineerFirm implements Company{
22     private int[] income;
23
24     EngineerFirm(int n){
25         this.income=new int[n];
26     }
27
28     @Override
29     public void assignSalaries(int[] salaries){
30         int j=0;
31         int l=income.length-1;
32         for(int i:salaries){
33             if(l>-1){
34                 income[j]=i;
35                 j=j+1;
36                 l=l-1;
37             }
38         }
39         System.out.println("Incomes of engineers credited");
40     }
41
42     @Override
43     public void averageSalary(){
44         double sum=0;
45         for(int i:income){
46             sum=sum+i;
47         }
48         sum=sum/(income.length);
49         System.out.println("Average salary of engineers is "+String.format("%.2f",sum));
50     }
51
52     @Override
53     public void...
```

Problem Description: A company named `Company` has the following methods:

- `assignSalaries(int[] salaries)`: Assigns the array `salaries` to the array `income`.
- `averageSalary()`: Prints the average salary in the array `income`.
- `minSalary()`: Prints the minimum salary in the array `income`.

There are three classes, `EngineerFirm` and `AccountantFirm`, which implement the `Company` interface. The details of these classes follow.

`EngineerFirm` should have a variable of type `int[] income`. It should implement the following methods:

- `EngineFirm(int n)`: Initializes the empty array `income` of length `n` where `n` is the number of engineers.
- `signSalaries(int[] salaries)`: Assigns the array `salaries` to array `income`.
- `averageSalary()`: Prints the average salary in the array `income`.
- `minSalary()`: Prints the minimum salary in the array `income`.

`AccountantFirm` should have a variable of type `int[] income`. It should implement the following methods:

- `AccountantFirm(int n)`: Initializes the empty array `income` of length `n` where `n` is the number of accountants.
- `signSalaries(int[] salaries)`: Assigns the array `salaries` to array `income`.
- `averageSalary()`: Prints the average salary in the array `income`.
- `minSalary()`: Prints the minimum salary in the array `income`.

Test Results Custom Input

Implementation

Company has the following

`int[] salaries;`

EngineerFirm and AccountantFirm, implement Company interface. The details

should have a variable of type `int[]` to implement the following

copy array `income` of length number of engineers.

to all the engineers,

`int[] salaries;`: Assigns the

values to array `income`.

`income` arrays differ in

many values as possible and

engineers credited".

Prints the average salary in the of engineers is

```
Java 8 ● Autocomplete Ready ⓘ
51
52     @Override
53     public void maxSalary(){
54         int max=income[0];
55         for(int i:income){
56             if(i>max){
57                 max=i;
58             }
59         }
60         System.out.println("Maximum salary amongst engineers is "+max);
61     }
62
63     @Override
64     public void minSalary(){
65         int min=income[0];
66         for(int i:income){
67             if(i<min){
68                 min=i;
69             }
70         }
71         System.out.println("Minimum salary amongst engineers is "+min);
72     }
73
74 }
75
76 class AccountantFirm implements Company{
77
78     private int[] income;
79
80     AccountantFirm(int n){
81         this.income=new int[n];
82     }
83 }
```

Test Results Custom Input

Implementation

Company has the following

`int[] salaries;`

EngineerFirm and AccountantFirm, implement Company interface. The details

should have a variable of type `int[]` to implement the following

copy array `income` of length number of engineers.

to all the engineers,

`int[] salaries;`: Assigns the

values to array `income`.

`income` arrays differ in

many values as possible and

engineers credited".

Prints the average salary in the of engineers is

```
Java 8 ● Autocomplete Ready ⓘ
76
77     class AccountantFirm implements Company{
78
79         private int[] income;
80
81         AccountantFirm(int n){
82             this.income=new int[n];
83         }
84
85         @Override
86         public void assignSalaries(int[] salaries){
87             int j=0;
88             int l=income.length-1;
89             for(int i:salaries){
90                 if(l>-1){
91                     income[j]=i;
92                     j=j+1;
93                     l--;
94                 }
95             }
96             System.out.println("Incomes of accountants credited");
97         }
98
99         @Override
100        public void averageSalary(){
101            double av=0;
102            for(int i:income){
103                av=av+i;
104            }
105            av=av/(income.length);
106            System.out.println("Average salary of accountants is "+String.format("%.2f",av));
107        }
108 }
```

Test Results Custom Input

Implementation

A Company has the following

```
100     double av=0;
101     for(int i:income){
102         av=av+i;
103     }
104     av=av/(income.length);
105     System.out.println("Average salary of accountants is "+String.format("%.2f",av));
106 }
107
108     @Override
109     public void maxSalary(){
110         int max=income[0];
111         for(int i:income){
112             if(i>max){
113                 max=i;
114             }
115         }
116         System.out.println("Maximum salary amongst accountants is "+max);
117     }
118
119     @Override
120     public void minSalary(){
121         int min=income[0];
122         for(int i:income){
123             if(i<min){
124                 min=i;
125             }
126         }
127         System.out.println("Minimum salary amongst accountants is "+min);
128     }
129 }
130
131 > public class Solution {
```

Test Results Custom Input Run

COUNTING PAIRS

```
> class Result {
    /*
     * Complete the 'countPairs' function below.
     *
     * The function is expected to return an INTEGER.
     * The function accepts following parameters:
     * 1. INTEGER_ARRAY numbers
     * 2. INTEGER k
    */

    public static int countPairs(List<Integer> numbers, int k) {
        // Write your code here
        SortedSet<Integer> nums = new TreeSet<Integer>();
        int result = 0;
        for(Integer number : numbers)
            nums.add(number);
        for(Integer number : nums)
            if(nums.contains(number + k))
                result++;
        return result;
    }

    > public class Solution {
```

Test Results Custom Input Run Code

OR

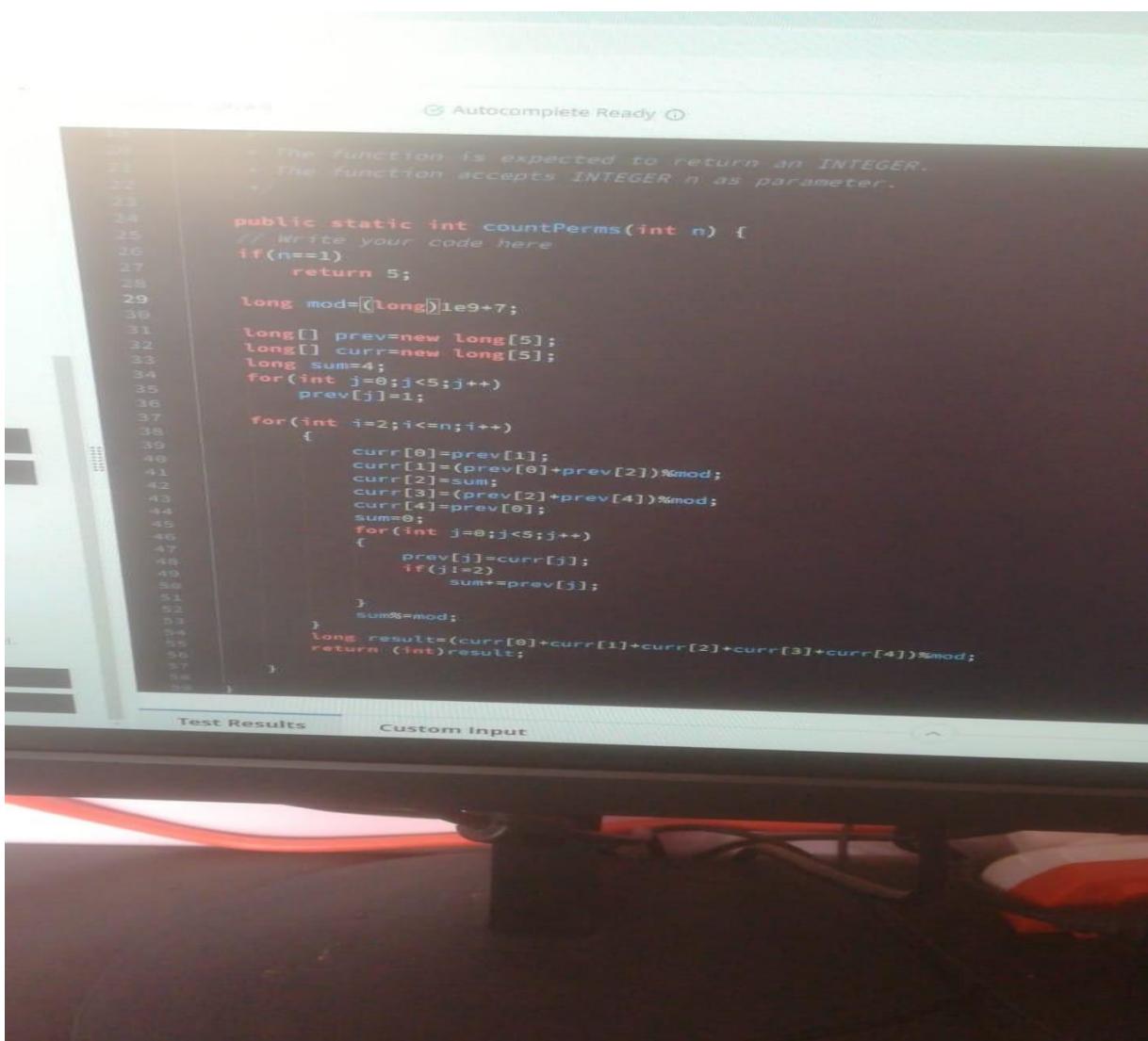
```
public static int countPairs(List<Integer> numbers, int k) {  
    // Write your code here  
    int count=0;  
    if(k==0)  
    {  
        Map<Integer, Integer> ac_map = new HashMap<>();  
        for(Integer ac_n:numbers)  
        {  
            ac_map.put(ac_n,ac_map.getOrDefault(ac_n, 0)+1);  
            if(ac_map.get(ac_n)==2)  
            {  
                count++;  
            }  
        }  
        return count;  
    }  
  
    Set<Integer> ac_set = new HashSet<>();  
  
    for(Integer ac_n:numbers)  
    {  
        if(!ac_set.contains(ac_n))  
        {  
            if(ac_set.contains(ac_n-k)) count++;  
            if(ac_set.contains(ac_n+k)) count++;  
            ac_set.add(ac_n);  
        }  
    }  
    return count;  
}  
}
```

Line

COUNTING PERMUTATION

```
Language: Java 8          Autocomplete Ready ⓘ  
1 > import java.io.*; ...  
14  
15 class Result {  
16  
17     /*  
18     * Complete the 'countPerms' function below.  
19     *  
20     * The function is expected to return an INTEGER.  
21     * The function accepts INTEGER n as parameter.  
22     */  
23  
24     public static int countPerms(int n) {  
25         // Write your code here  
26         long[][] per=new long[n+1][5];  
27         int MOD=(int)(1e9+7);  
28         for(int i=0;i<5;i++)  
29             per[1][i]=1;  
30  
31     }  
32  
33     int[][] relation=new int[][]{  
34         {1},{0,2},{0,1,3,4},{2,4},{0}  
35     };  
36  
37     for(int i=1;i<n;i++){  
38         for(int u=0;u<5;u++){  
39             per[i+1][u]=0;  
40             for(int v:relation[u])  
41                 per[i+1][u]+=(per[i][v])%MOD;  
42             }  
43         }  
44     }  
45     long ans=0;  
46     for(int i=0;i<5;i++){  
47         ans=(ans+per[n][i])%MOD;  
48     }  
49     return (int)ans;  
50 }  
51  
52 }  
53  
54 > public class Solution { ...
```

OR



OR

```
public class Solution {  
  
    private static final int MOD = 1000000007;  
  
    public int countVowelPermutation(int n) {  
  
        long[] current = new long[]{1,1,1,1,1};  
  
        for (int i = 1; i < n; i++) {  
  
            long[] next = new long[]{0,0,0,0,0};  
  
            // Each vowel 'a' may only be followed by an 'e'.  
            next[1] += current[0];  
  
            // Each vowel 'e' may only be followed by an 'a' or an 'i'.  
            next[0] += current[1];  
  
            current = next;  
        }  
        return (int)(current[0]+current[1]+current[2]+current[3]+current[4])%MOD;  
    }  
}
```

```

next[2] += current[1];

// Each vowel 'i' may not be followed by another 'i'.

next[0] += current[2];

next[1] += current[2];

next[3] += current[2];

next[4] += current[2];

// Each vowel 'o' may only be followed by an 'i' or a 'u'.

next[2] += current[3];

next[4] += current[3];

// Each vowel 'u' may only be followed by an 'a'.

next[0] += current[4];

// assign

current[0] = next[0] % MOD;

current[1] = next[1] % MOD;

current[2] = next[2] % MOD;

current[3] = next[3] % MOD;

current[4] = next[4] % MOD;

}

return (int)((current[0]+current[1]+current[2]+current[3]+current[4]) % MOD);

}

public static void main(String args[]) {

    Solution s = new Solution();

    System.out.println(s.countVowelPermutation(1));

    System.out.println(s.countVowelPermutation(2));

    System.out.println(s.countVowelPermutation(5));

    System.out.println(s.countVowelPermutation(20000));

}

}

```

BALANCED SUM

```
23  public static int balancedSum(List<Integer> arr) {  
24      // Write your code here  
25      int ar[] = arr.stream().mapToInt(Integer::intValue).toArray();  
26      int n = arr.size();  
27      int pre [] = new int[n];  
28      int suf [] = new int[n];  
29      pre[0] = ar[0];  
30      suf[0] = ar[n-1];  
31  
32      for(int i=1; i<n; i++)  
33      {  
34          pre[i] = pre[i-1]+ar[i];  
35          suf[i] = suf[i-1]+ar[n-i-1];  
36      }  
37  
38      for(int i=0;i<n;i++)  
39      {  
40          if(pre[i]==suf[n-i-1])  
41              return i;  
42      }  
43      return -1;  
44  
45  }
```

OR

HCL

Online Java Compiler - Online Ja +

18

Java 8 Autocomplete Ready

```
1 > import java.io.*;
14 class Result {
15
16     /*
17      * Complete the 'balancedSum' function below.
18      *
19      * The function is expected to return an INTEGER.
20      * The function accepts INTEGER_ARRAY arr as parameter.
21      */
22
23     public static int balancedSum(List<Integer> arr) {
24         // Write your code here
25         int sum1 = 0;
26         for(int i = 0; i < arr.size(); i++) {
27             int flag = 0;
28             int sum2 = 0;
29             sum1 += arr.get(i);
30             for(int j = i+2; j < arr.size(); j++) {
31                 sum2 += arr.get(j);
32                 if(sum2 > sum1) {
33                     flag = 1;
34                     break;
35                 }
36             }
37             if(flag == 0 && sum1 == sum2) {
38                 return (i+1);
39             }
40         }
41         return 0;
42     }
43
44 }
45 > public class Solution { ... }
```

Test Results Custom Input

BALANCING PARANTHESIS

The screenshot shows a Java code editor with the following details:

- Title:** 1. Balancing Parentheses
- Description:** Given a string that consists of left and right parentheses, '()' and ')', balance the parentheses by inserting parentheses as necessary. Determine the minimum number of characters that must be inserted.
- Example:** $s = ()()$. Insert 1 left parenthesis at the left end of the string to get '(())'. The string is balanced after 1 insertion.
- Constraints:**
 - $1 \leq \text{length of } s \leq 10^5$
- Input Format For Custom Testing:** STDIN Function

()() → s = "()"
- Sample Case 0:** Sample Input
STDIN Function

()() → s = "()"
- Code Snippet:** A partially completed Java class named `RESULT` with a `getMin` method. The code uses a balance counter (`bal`) and an answer counter (`ans`) to calculate the minimum number of insertions required to balance the string `s`.
- Test Results:** Run Code Run Tests

GET MINIMUM

The screenshot shows a Java code editor with the following details:

- Description:** Given a string that consists of left and right parentheses, '()' and ')', balance the parentheses by inserting parentheses as necessary. Determine the minimum number of characters that must be inserted.
- Text:** Insert 1 left parenthesis at the left end of the string to balance after 1 insertion.
- Input Format For Custom Testing:** STDIN Function

()() → s = "()"
- Code Snippet:** A partially completed Java class named `Result` with a `getMin` method. The code uses a balance counter (`bal`) and an answer counter (`a`) to calculate the minimum number of insertions required to balance the string `s`.

OR

Digital Banking Citrix Gateway Home | Lex How to Install JDK... Execution of Java program

Page: java 8 Autocomplete Ready

```
/*
 * Complete the "getMin" function below.
 * The function is expected to return an INTEGER.
 * The function accepts STRING s as parameter.
 */
public static int getMin(String s) {
    // Write your code here
    int countOpen=0;
    int countClose=0;
    char[] c=s.toCharArray();

    for(char ch:c)
    {
        if(ch=='(')
            countOpen+=1;
        else if(ch==')')
            countClose+=1;

    }
    if(countOpen>countClose)
        return countOpen-countClose;
    else if(countClose>countOpen)
        return countClose-countOpen;
    else return 0;
}
> public class Solution {
```

Test Results Custom Input

FIND BEFORE MATRIX

Java 8 Autocomplete loading...

```
public static List<List<Integer>> findBeforeMatrix(List<List<Integer>> after) {
    // Write your code here
    int s=0;
    List<List<Integer>> before= new ArrayList<List<Integer>>();
    for(int i=0;i<after.size()-1;i++){
        List<Integer> bRow=new ArrayList<Integer>();
        for(int j=0;j<=after.get(i).size()-1;j++){
            //bRow.add(j,after.get(i).get(j));
            if(i==0 && j==0){
                bRow.add(j,after.get(0).get(0));

            }
            else if(i==0){
                bRow.add(j,after.get(0).get(j)-after.get(0).get(j-1));
            }
            else if(j==0){
                bRow.add(j,after.get(i).get(0)-after.get(i-1).get(0));
            }
            else{
                bRow.add(j,after.get(i).get(j)+after.get(i-1).get(j-1)-after.get(i).get(j-1)-after.get(i-1).get(j));
            }
        }
        before.add(i,bRow);
    }
    return before;
}
```

DIVICE NAME SYSTEM

```
public static List<String> deviceNamesSystem(List<String> devicenames) {  
    // Write your code here  
    String arr[] = new String[devicenames.size()];  
    arr = devicenames.toArray(arr);  
    String unique[] = new String[arr.length];  
    for(int i=0;i<arr.length;i++)  
    {  
        String name = arr[i];  
        boolean d = false;  
        for(int j=0;j<i;j++)  
        {  
            if(unique[j].equals(name))  
            {  
                d=true;  
                break;  
            }  
        }  
        if(!d)  
        {  
            unique[i]=name;  
        }  
        else  
        {  
            int id=1;  
            boolean e=true;  
            while(e)  
            {  
                e=false;  
                for(int k=0;k<i;k++)  
                {  
                    if(unique[k].equals(name+id))  
                    {  
                        e=true;  
                        id++;  
                        break;  
                    }  
                }  
                unique[i]=name+id;  
            }  
        }  
    }  
    return Arrays.asList(unique);  
}  
}
```

```
42    {  
43        unique[i]=name;  
44    }  
45    else  
46    {  
47        int id=1;  
48        boolean e=true;  
49        while(e)  
50        {  
51            e=false;  
52            for(int k=0;k<i;k++)  
53            {  
54                if(unique[k].equals(name+id))  
55                {  
56                    e=true;  
57                    id++;  
58                    break;  
59                }  
60            }  
61        }  
62        unique[i]=name+id;  
63    }  
64 }  
65 return Arrays.asList(unique);  
66  
67 }  
68  
69 }  
70  
71 > public class Solution {
```

OR

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

public class test {

    public static List<String>
deviceNamesSystem(List<String> devicenames){
        Map<String,Integer> devNumList=new
HashMap<String,Integer>();
        List<String> newNameList=new
ArrayList<String>();
        for(String s:devicenames) {
            if(devNumList.containsKey(s)) {

                newNameList.add(s+devNumList.get(s));
                devNumList.replace(s,
devNumList.get(s)+1);
            }
            else {
                newNameList.add(s);
                devNumList.put(s, 1);
            }
        }
        return newNameList;
    }

    public static void main(String[] args) {

        List<String> deviceNames=new
ArrayList<String>();
        deviceNames.add("Switch");
        deviceNames.add("tv");
        deviceNames.add("Switch");
    }
}
```

```

        deviceNames.add("tv");
        deviceNames.add("Switch");
        deviceNames.add("tv");
        deviceNames.add("Switch");
        deviceNames.add("tv");

deviceNames=deviceNamesSystem(deviceNames);
    deviceNames.forEach(x-
>System.out.println(x));
}

}

```

CONDENSED LIST

```

77     */
78
79     public static SinglyLinkedListNode condense(SinglyLinkedListNode head) {
80     // Write your code here
81     HashSet<Integer> hSet = new HashSet<>();
82     SinglyLinkedListNode ptr1;
83     SinglyLinkedListNode ptr2;
84     ptr1 = head;
85     ptr2 = null;
86     while(ptr1!=null){
87         int val = ptr1.data;
88         if(hSet.contains(val))ptr2.next = ptr1.next;
89         else{
90             hSet.add(val);
91             ptr2 = ptr1;
92         }
93         ptr1 = ptr1.next;
94     }
95     return head;
96

```

OR

```

// Java program to remove duplicates from unsorted
// linked list

```

```

class LinkedList {

    static Node head;

```

```

static class Node {

    int data;
    Node next;

    Node(int d) {
        data = d;
        next = null;
    }
}

/* Function to remove duplicates from an
unsorted linked list */
void remove_duplicates() {
    Node ptr1 = null, ptr2 = null, dup = null;
    ptr1 = head;

    /* Pick elements one by one */
    while (ptr1 != null && ptr1.next != null) {
        ptr2 = ptr1;

        /* Compare the picked element with rest
        of the elements */
        while (ptr2.next != null) {

            /* If duplicate then delete it */
            if (ptr1.data == ptr2.next.data) {

                /* sequence of steps is
                important here */
                dup = ptr2.next;
                ptr2.next = ptr2.next.next;
                System.gc();
            } else /* This is tricky */ {

```

```
        ptr2 = ptr2.next;
    }
}
ptr1 = ptr1.next;
}
```

```
void printList(Node node) {
    while (node != null) {
        System.out.print(node.data + " ");
        node = node.next;
    }
}
```

```
public static void main(String[] args) {
    LinkedList list = new LinkedList();
    list.head = new Node(3);
    list.head.next = new Node(4);
    list.head.next.next = new Node(3);
    list.head.next.next.next = new Node(2);
    list.head.next.next.next.next = new Node(6);
    list.head.next.next.next.next.next = new
    Node(1);
    list.head.next.next.next.next.next.next = new
    Node(2);
    list.head.next.next.next.next.next.next.next = new
    Node(6);
}
```

```
System.out.println("Linked List before
removing duplicates : \n ");
list.printList(head);
```

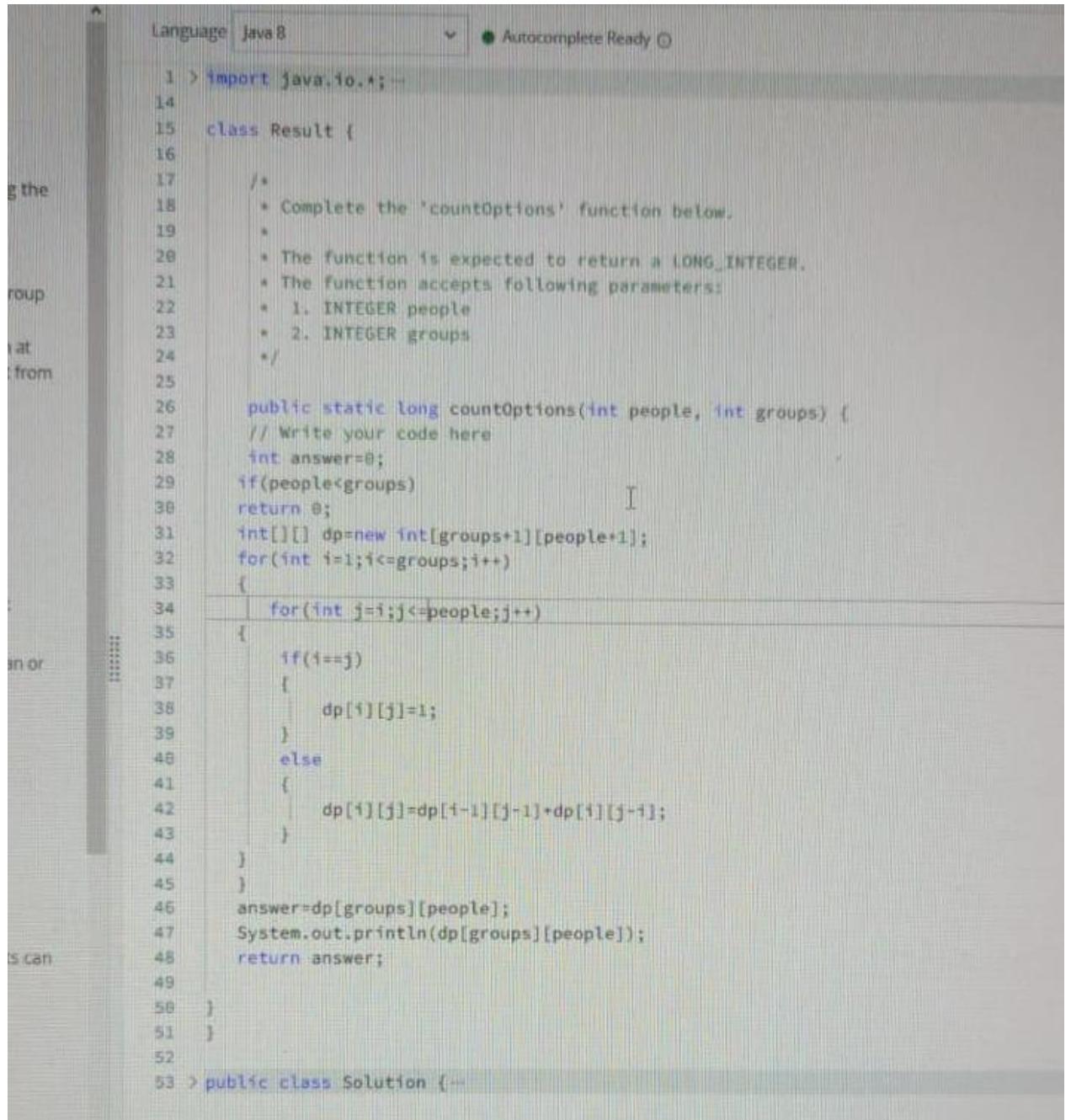
```
list.remove_duplicates();
System.out.println("");
```

```

        System.out.println("Linked List after removing
duplicates : \n ");
        list.printList(head);
    }
}

```

GROUPING OPTIONS



The screenshot shows a Java code editor with the following details:

- Language:** java 8
- Status Bar:** Autocomplete Ready
- Code Content:**

```

1 > import java.io.*;
14
15 class Result {
16
17     /*
18      * Complete the 'countOptions' function below.
19      *
20      * The function is expected to return a LONG_INTEGER.
21      * The function accepts following parameters:
22      * 1. INTEGER people
23      * 2. INTEGER groups
24     */
25
26     public static long countOptions(int people, int groups) {
27         // Write your code here
28         int answer=0;
29         if(people<groups)
30             return 0;
31         int[][] dp=new int[groups+1][people+1];
32         for(int i=1;i<=groups;i++)
33         {
34             for(int j=i;j<=people;j++)
35             {
36                 if(i==j)
37                 {
38                     dp[i][j]=1;
39                 }
40                 else
41                 {
42                     dp[i][j]=dp[i-1][j-1]+dp[i][j-1];
43                 }
44             }
45         }
46         answer=dp[groups][people];
47         System.out.println(dp[groups][people]);
48         return answer;
49
50     }
51 }
52
53 > public class Solution {-

```

LIST INHERITANCE

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

interface MyList {
    void convert(String[] a);

    void replace(int idx);

    ArrayList<String> compact();
}

class InvalidStringException extends Exception {

    private static final long serialVersionUID = 1L;

    public InvalidStringException() {
        super();
    }
}

class ArrayToList implements MyList {

    ArrayList<String> arrayToList;

    public ArrayToList() {
```

```
arrayToList = new ArrayList<>();
```

```
}
```

```
@Override
```

```
public void convert(String[] a) {
```

```
    for (int i = 0; i < a.length; i++) {
```

```
        arrayToList.add(a[i]);
```

```
        System.out.println("I have added the string: " +  
a[i] + " at the index: " + i);
```

```
}
```

```
}
```

```
@Override
```

```
public void replace(int idx) {
```

```
    String string = arrayToList.get(idx);
```

```
    arrayToList.set(idx, null);
```

```
    System.out.println("I have replaced the string: " +  
string + " with a null string ");
```

```
}
```

```
@Override
```

```
public ArrayList<String> compact() {
```

```
    arrayToList.removeAll(Collections.singleton(null));
```

```
    return arrayToList;
```

```
}
```

```
}
```

```
public class Solution {  
  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
  
        int n = sc.nextInt();  
        String[] a = new String[n];  
  
        for (int i = 0; i < a.length; i++) {  
            a[i] = sc.next();  
        }  
  
        ArrayToList arrayToList = new ArrayToList();  
  
        arrayToList.convert(a);  
  
        arrayToList.replace(1);  
  
        arrayToList.compact();  
  
        sc.close();  
    }  
}
```

OR

```
class InvalidStringException extends Exception {  
    public InvalidStringException(String s)  
    {  
        super(s);  
    }  
}  
  
class ArrayToList implements MyList{  
    private ArrayList<String> darray=new ArrayList<>();  
    private int index=0;  
    private String n;  
    public ArrayToList( )  
    {  
  
    }  
    public void convert(String[] a)  
    {  
        for(int i=0;i<a.length;i++)  
        {  
            darray.add(i,a[i]);  
            index=darray.indexOf(a[i]);  
            System.out.println("I have added the string: "+a[i]+" at  
the index: "+i);  
        }  
    }  
    public void replace(int idx){  
        n=darray.get(idx);  
        darray.set(idx, "");  
        System.out.println("I have replaced the string: "+n+"  
with a null string");  
    }  
    public ArrayList<String> compact()  
    {  
        darray.removeAll(Arrays.asList(""));
```

```
    return darray;  
}  
  
}
```

SPORTS INHERITANCE

```
package test;  
  
import java.util.*;  
import java.io.*;  
  
import java.io.*;  
import java.math.*;  
import java.security.*;  
import java.text.*;  
import java.util.*;  
import java.util.concurrent.*;  
import java.util.regex.*;  
  
interface Sport{  
    void calculateAvgAge(int[] age);  
    void retirePlayer(int id);  
}  
  
  
class Cricket implements Sport{  
    public int playerIDs[];  
  
    Cricket(){
```

```
playerIDs=new int[11];
for(int i=0;i<11;i++) {
    playerIDs[i]=1;
}
System.out.println("A new cricket team has been
formed");
}
```

```
public void calculateAvgAge(int[] age) {
    double sum=0;
    int n=age.length;
    for(int i=0;i<n;i++) {
        sum+=age[i];
    }
    double avgAge=sum/n;
    System.out.format("The average age of team is
%.2f",avgAge);
}
```

```
public void retirePlayer(int id) {
    if(playerIDs[id]==-1) {
        System.out.println("Player has already
retired");
    }else {
        playerIDs[id]=-1;
        System.out.println("Player with id: "+id+" has
retired");
    }
}
```

```
 }  
 }
```

```
class Football implements Sport{  
    public int playerIDs[];
```

```
    Football(){  
        playerIDs=new int[11];  
        for(int i=0;i<11;i++) {  
            playerIDs[i]=1;  
        }
```

```
        System.out.println("A new football team has been  
formed");  
    }
```

```
    public void calculateAvgAge(int[] age) {  
        double sum=0;  
        int n=age.length;  
        for(int i=0;i<n;i++) {  
            sum+=age[i];  
        }  
        double avgAge=sum/n;  
        System.out.format("The average age of team is  
%.2f",avgAge);  
    }
```

```
    public void retirePlayer(int id) {
```

```
    if(playerIDs[id-1]==-1) {  
        System.out.println("Player has already  
retired");  
    }else {  
        playerIDs[id-1]=-1;  
        System.out.println("Player with id: "+id+" has  
retired");  
    }  
}
```

```
public void playerTransfer(int fee,int id) {  
    if(playerIDs[id]==-1) {  
        System.out.println("Player has already  
retired");  
    } else {  
        System.out.println("Player with id: "+id+" has  
been transferred with a fee of "+fee);  
    }  
}
```

```
public class Test {
```

```
    private static final Scanner scanner = new  
Scanner(System.in);
```

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

```
    int age[]={26,32,36,33,24,31,30,35,36,21,28};
```

```
    Cricket obj=new Cricket();
```

```
    obj.calculateAvgAge(age);
```

```
    obj.retirePlayer(5);
```

```
    obj.retirePlayer(5);
```

```
}
```

```
}
```

CRICKET SPORTS

The screenshot shows a Java code editor with the following details:

- Code Content:**

```
36 31 25 23 30  
Language: Java 8  
Autocomplete Ready  
1 > import java.util.*;  
7  /*  
8  Write the implementation of the Cricket and Football classes  
9  */  
10 class Cricket implements Sport{  
11     int[] playerIds;  
12     public Cricket(){  
13         this.playerIds=new int[12];  
14         Arrays.fill(playerIds,1);  
15         System.out.println("A new cricket team has been formed");  
16     }  
17     public void calculateAvgAge(int[] age){  
18         float avg=0;  
19         for(int i=0;i<age.length;i++){  
20             avg=avg+age[i];  
21         }  
22         avg=avg/age.length;  
23         System.out.printf("The average age of the team is %.2f",avg);  
24         System.out.println();  
25     }  
26     public void retirePlayer(int id){  
27         if(playerIds[id]==-1){  
28             System.out.println("Player has already retired");  
         }  
     }  
}
```
- Environment:** The code is being run in a Java 8 environment.
- UI Elements:** The interface includes tabs for "Test Results" and "Custom Input", and buttons for "Run Code", "Run Tests", and "Submit". The status bar at the bottom shows the date (19-04-2021), time (14:46), and language (ENG).

```
24 36 31 25 23 30
Language: Java 8
    if(playerIds[id]==-1){
        System.out.println("Player has already retired");
    }
    else{
        playerIds[id]=-1;
        System.out.println("Player with id: "+id+" has retired");
    }
}
class Football implements Sport{
    int[] playerIds;
    public Football(){
        this.playerIds=new int[12];
        Arrays.fill(playerIds,1);
        System.out.println("A new football team has been formed");
    }
    public void calculateAvgAge(int[] age){
        float avg=0;
        for(int i=0;i<age.length;i++){
            avg=avg+age[i];
        }
        avg=avg/age.length;
    }
}

```

Line: 69 Col: 74

Run Code Run Tests Submit

```
36 31 25 23 30
Language: Java 8
    }
    avg=avg/age.length;
    System.out.printf("The average age of the team is %.2f",avg);
    System.out.println();
}
public void retirePlayer(int id){
    if(playerIds[id]==-1){
        System.out.println("Player has already retired");
    }
    else{
        playerIds[id]=-1;
        System.out.println("Player with id: "+id+" has retired");
    }
}
public void playerTransfer(int fee,int id){
    if(playerIds[id]==-1){
        System.out.println("Player has already retired");
    }
    else{
        playerIds[id]=1;
        System.out.println("Player with id: "+id+" has been transferred with a
fee of "+fee);
    }
}

```

Line: 69 Col: 74

Run Code Run Tests Submit

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";  
}  
}
```

```
class InnovaCrysta extends Car{
```

```
int mileage;  
public InnovaCrysta(Integer mileage){  
  
    super(false, "6");  
    this.mileage = mileage;  
  
}  
  
}
```

```
@Override  
public String getMileage(){  
    String mil=Integer.toString(mileage);  
    return mil+" kmpl";  
}
```

```
}
```

PRISON BREAK

guage: Java 8

⌚ Autocomplete Ready ⌚

```
1  PUBLIC static long prison(int n, int m, List<Integer> h, List<Integer> v) {
2      // Write your code here
3      boolean[] xBol=new boolean[n+1];
4      Arrays.fill(xBol,true);
5      boolean[] yBol=new boolean[m+1];
6      Arrays.fill(yBol,true);
7      for(int x:h){
8          xBol[x]=false;
9      }
10     for(int y:v){
11         yBol[y]=false;
12     }
13     int cx=0,xMax=Integer.MIN_VALUE,cy=0,yMax=Integer.MIN_VALUE;
14     for(int i=1;i<=n;i++){
15         if(xBol[i]){
16             cx=0;
17         }else{
18             cx++;
19             xMax=Math.max(xMax,cx);
20         }
21     }
22     for(int i=1;i<=m;i++){
23         if(yBol[i]){
24             cy=0;
25         }else{
26             cy++;
27             yMax=Math.max(yMax,cy);
28         }
29     }
30     return (xMax+1)*(yMax+1);
31 }
32 }
```

Matrix Summation

```
 23
 24
 25     public static List<List<Integer>> findBeforeMatrix(List<List<Integer>> after) {
 26         // Write your code here
 27         int s=0;
 28         List<List<Integer>> before= new ArrayList<List<Integer>>();
 29         for(int i=0;i<=after.size()-1;i++){
 30             List<Integer> bRow=new ArrayList<Integer>();
 31             for(int j=0;j<=after.get(i).size()-1;j++){
 32                 //bRow.add(j,after.get(i).get(j));
 33                 if(i==0 && j==0){
 34                     bRow.add(j,after.get(0).get(0));
 35
 36                 }
 37                 else if(i==0){
 38                     bRow.add(j,after.get(0).get(j)-after.get(0).get(j-1));
 39                 }
 40                 else if(j==0){
 41                     bRow.add(j,after.get(i).get(0)-after.get(i-1).get(0));
 42                 }
 43                 else{
 44                     bRow.add(j,after.get(i).get(j)+after.get(i-1).get(j-1)-after.get(i).get(j-1)-after.get(i-1).get(j));
 45                 }
 46             }
 47             before.add(i,bRow);
 48         }
 49     }
 50     return before;
 51 }
 52 }
```

Merge 2 arrays

```
22     * 1. INTEGER_ARRAY a
23     * 2. INTEGER_ARRAY b
24     */
25
26     public static List<Integer> mergeArrays(List<Integer> a,
27         // Write your code here
28         List<Integer> out=new ArrayList<Integer>();
29         // List<integer> out1=new ArrayList<Integer>();
30         out.addAll(a);
31         out.addAll(b);
32         out.sort(Comparator.naturalOrder());
33         return out;
34     }
35
36 }
37
38 > public class Solution { ...
```

Test Results

Custom Input

NUMBER OF MOVES

Details - SumTotal X | External site X | +

6reqbht88o.

Language: C# Autocomplete Ready

```
1 > using System.CodeDom.Compiler;-
16 class Result
17 {
18
19     /*
20      * Complete the 'minMoves' function below.
21      *
22      * The function is expected to return an INTEGER.
23      * The function accepts following parameters:
24      * 1. INTEGER n
25      * 2. INTEGER startRow
26      * 3. INTEGER startCol
27      * 4. INTEGER endRow
28      * 5. INTEGER endCol
29     */
30
31     public class cell
32     {
33         public int x, y;
34         public int dis;
35
36         public cell(int x, int y, int dis)
37         {
38             this.x=x;
39             this.y=y;
40             this.dis=dis;
41         }
42     }
43
44     public static bool IsInside(int x, int y, int n)
45     {
46         if(x>=0 && x<n && y>0 && y<n)
47             return true;
48
49         return false;
50     }
51
52     public static int minMoves(int n, int startRow, int startCol, int endRow, int endCol)
53     {
54         int[] knightPos={startRow, startCol};
55         int[] targetPos={endRow, endCol};
56         int[] dx={-2,-1,1,2,-2,-1,1,2};
57         int[] dy={-1,-2,-2,-1,1,2,2,1};
```

Test Results Custom Input

External site X | +

Bo

Language: C# Autocomplete Ready

```
public static int minMoves(int n, int startRow, int startCol, int endRow, int endCol)
{
    int[] knightPos={startRow, startCol};
    int[] targetPos={endRow, endCol};
    int[] dx={-2,-1,1,2,-2,-1,1,2};
    int[] dy={-1,-2,-2,-1,1,2,2,1};

    Queue<cell> q= new Queue<cell>();
    q.Enqueue(new cell(knightPos[0], knightPos[1], 0));
```

Java - SumTotal X External Site X +

reqbht8Bo

Language C# Autocomplete Ready.

```
51
52
53     cell t;
54     int x, y;
55
56     bool[,] visit=new bool[n+1,n+1];
57
58     for(int i=1;i<=n;i++)
59         for(int j=1;j<=n;j++)
60             visit[i,j]=false;
61
62     visit[knightPos[0], knightPos[1]] = true;
63
64     while(q.Count !=0)
65     {
66         t=q.Peek();
67         q.Dequeue();
68
69         if(t.x== targetPos[0] && t.y == targetPos[1])
70             return t.dis;
71
72         for(int i=0; i<8; i++)
73         {
74             x=t.x+dx[i];
75             y=t.y+dy[i];
76
77             if(IsInside(x,y,n) && !visit[x,y])
78             {
79                 visit[x,y]=true;
80                 q.Enqueue(new cell (x,y,t.dis+1));
81             }
82         }
83     }
84
85     return int.MaxValue;
86 }
87
88 }
```

Test Results Custom Input

OR

```
class Solution {
    // Class for storing a Point's data
    static class Point {
        int x, y;
        int dis;

        public Point(int x, int y, int dis)
        {
            this.x = x;
        }
    }
}
```

```

    this.y = y;
    this.dis = dis;
}

}

// Utility method returns true if (x, y) lies
// inside Board

static boolean isInside(int x, int y, int N)
{
    if (x >= 0 && x < N && y >= 0 && y < N)
        return true;
    return false;
}

public static int minMoves(int n, int startRow, int startCol, int endRow, int
endCol) {

    int dx[] = { -2, -1, 1, 2, -2, -1, 1, 2 };
    int dy[] = { -1, -2, -2, -1, 1, 2, 2, 1 };
    int minDistance = Integer.MAX_VALUE;
    Queue<Point> queue = new ArrayDeque<>();
    queue.add(new Point(startRow, startCol, 0));

    Point t;
    int x, y;
    boolean visit[][] = new boolean[n][n];
    // make all Point unvisited
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            visit[i][j] = false;

    visit[startRow][startCol] = true;

    while (!queue.isEmpty()) {
        t = queue.poll();

```

```

        if (t.x == endRow && t.y == endCol){

            if(minDistance > t.dis)

                minDistance = t.dis;

        }

        for (int i = 0; i < 8; i++) {

            x = t.x + dx[i];

            y = t.y + dy[i];



            // If reachable state is not yet visited and

            // inside board, push that state into queue

            if (isInside(x, y, n) && !visit[x][y]) {

                visit[x][y] = true;

                queue.add(new Point(x, y, t.dis + 1));

            }

        }

    }

    return minDistance == Integer.MAX_VALUE? -1:minDistance;
}

// Driver code

public static void main(String[] args)

{

    System.out.println( minMoves(7, 6, 6, 1,0));

}

}

```

JAVA EXCEPTIONS: SECURING MESSAGES

package com.nastra.hackerrank;

import com.nastra.hackerrank.util.FastScanner;

```
public class Encryption {  
  
    public static void main(String[] args) throws Exception {  
        FastScanner sc = new FastScanner(System.in);  
        String word = sc.next();  
  
        System.out.println(solve(word));  
    }  
  
    private static String solve(String word) {  
        char[][] squareCode = encrypt(word);  
        StringBuilder out = new StringBuilder();  
  
        int maxRows = squareCode.length;  
        int maxCols = squareCode[0].length;  
  
        int row = 0;  
        int col = 0;  
  
        while (true) {  
            if (row >= maxRows) {  
                row = 0;  
                col++;  
                out.append(" ");  
            }  
            if (col >= maxCols) {  
                break;  
            }  
            out.append(squareCode[row][col]);  
            row++;  
        }  
        return out.toString();  
    }  
}
```

```
        }

        char c = squareCode[row][col];

        if (c == '~') {

            row++;

            continue;

        } else {

            out.append(c);

        }

        row++;

    }

}

return out.toString().trim();

}

private static char[][] encrypt(String word) {

    int len = word.length();

    double row = Math.floor(Math.sqrt((double) len));

    double col = Math.ceil(Math.sqrt((double) len));



    // find smallest possible area where all characters fit into

    if (((int) row * col) < len) {

        double min = Math.min(row, col);

        if (min == row) {

            row++;

        } else {

            col++;

        }

    }

}
```

```

    }

char[][] squareCode = new char[(int) row][(int) col];
int k = 0;
for (int i = 0; i < row; i++) {
    for (int j = 0; j < col; j++) {
        if (k < len) {
            squareCode[i][j] = word.charAt(k);
            k++;
        } else {
            squareCode[i][j] = '~';
        }
    }
    return squareCode;
}
}

```

SUB STRING DIVISIBILITY

```

public class PE043_Sub_string_divisibility {
    private static int[] primeList = { 2, 3, 5, 7, 11, 13, 17 };
    private static String[][] modList;

```

```
private static long result = 0;  
private static String numbers = "0123456789";  
  
public static void main(String[] args) {  
    long start = System.nanoTime();  
  
    modList = new String[primeList.length][];  
    int prime;  
  
    for (int i = 0; i < primeList.length; i++) {  
        prime = primeList[i];  
        modList[i] = new String[prime];  
  
        for (int j = 0; j < prime; j++) {  
            modList[i][j] = new String("");  
        }  
  
        for (int j = 0; j < 1000; j += 100) {  
            if (j % prime == 0) {  
                modList[i][0] += (j / 100);  
            } else {  
                modList[i][(j / prime + 1) * prime % j] +=  
                    (j / 100);  
            }  
        }  
    }  
}
```

```
String mult17str;

for (int i = 102; i < 1003; i += 17) {
    if (!isValid(mult17str = String.valueOf(i))) {
        continue;
    } else {
        findSolutions(mult17str);
    }
}

long end = System.nanoTime();
long runtime = end - start;
System.out.println(result);
System.out.println("Runtime: " + runtime / 1000000 +
"ms (" + runtime
        + "ns)");
}

private static void findSolutions(String number) {
    if (number.length() == 9 && isValid(number)) {
        boolean found = false;
        int i;
        for (i = 0; !found && i < numbers.length(); i++) {
            if (number.indexOf(numbers.charAt(i)) == -1) {
                found = true;
            }
        }
    }
}
```

```
    }

    i--;

    if (i != 0) {
        result += Long.parseLong(String.valueOf(i) +
number);
    }
} else {
    int primeIndex = primeList.length - number.length()
+ 1;
    int mod = Integer.parseInt(number.substring(0, 2))
        % primeList[primeIndex];
    String newNumber;

    for (int i = 0; i < modList[primeIndex][mod].length();
i++) {
        if (isValid(newNumber =
modList[primeIndex][mod].substring(i,
            i + 1) + number)) {
            findSolutions(newNumber);
        }
    }
}

private static boolean isValid(String number) {
    boolean valid = true;
```

```

        for (int i = 0; valid && i < number.length() - 1; i++) {
            if (number.indexOf(number.charAt(i), i + 1) != -1) {
                valid = false;
            }
        }

        return valid;
    }
}

```

MINIMUM SUM

Don't know the correctness

```

public static int minSum(List<Integer> nums, int k) {

    Queue<Integer> maxPq = new PriorityQueue<>(nums.size(),
    Comparator.reverseOrder());

    maxPq.addAll(nums);

    while (k-- > 0) {
        int num = (int) Math.ceil(maxPq.poll() / 2.0);
        maxPq.add(num);
    }

    return maxPq.stream().mapToInt(i -> i).sum();
}

```

OR

Some test cases may not pass due to time complexity

```

function minSum(num, k) {

    var sum = 0;
    var i = 0;
    var firstMaxIndex = 0;
    var secondMaxIndex = 1;
}

```



```

    }

    for(var j=0; j<num.length; j++) {

        sum += num[j];
    }

    return sum;
}

```

SMALLEST SET COVERING INTERVALS

```

import java.util.Scanner;
import java.util.Arrays;

class Solution{
    static int solve(int[][] intv){
        Arrays.sort(intv,(int[] a, int[] b)->a[1]-b[1]);
        int count=0, busy[][]=new int[2][2];
        for(int[] x: intv){
            if(x[0]>busy[1][1]){
                ++count;
                busy[1]=x;
            }else if(x[0]>busy[0][1]){
                ++count;
                busy[0]=x;
                if(x[1]>busy[1][1]){
                    int[] temp=busy[0];
                    busy[0]=busy[1];
                    busy[1]=temp;
                }
            }
        }
        return count;
    }
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        int s=sc.nextInt();
        while(s-- != 0){
            int n=sc.nextInt();
            int[][] intv=new int[n][2];
            for(int i=0;i<n;++i){
                intv[i][0]=sc.nextInt();
                intv[i][1]=sc.nextInt();
            }
            System.out.println(solve(intv));
        }
        sc.close();
    }
}

```

OR

```

package com.belive.hackerrank.dp;

import java.io.BufferedReader;
import java.io.InputStreamReader;

```

```
import java.io.PrintWriter;
import java.util.Arrays;

public class IntervalSelection {

    public static class Interval implements Comparable<Interval> {

        public final int a, b;

        public Interval(int a, int b) {
            this.a = a;
            this.b = b;
        }

        @Override
        public int compareTo(Interval that) {
            int cmp = Integer.compare(this.b, that.b);

            if (cmp == 0) {
                return Integer.compare(this.a, that.a);
            } else {
                return cmp;
            }
        }

        @Override
        public String toString() {
            return "Line{" +
                    "a=" + a +
                    ", b=" + b +
                    '}';
        }
    }

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

        BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));
        PrintWriter writer = new PrintWriter(System.out);

        int T = Integer.parseInt(reader.readLine());

        for(int t = 0; t < T; t++){


```

```
int N;  
N = Integer.parseInt(reader.readLine());  
  
int A[] = new int[N];  
int B[] = new int[N];  
Interval[] I = new Interval[N];  
  
for(int i = 0; i < N; i++){  
    String[] s = reader.readLine().split(" ");  
    A[i] = Integer.parseInt(s[0]);  
    B[i] = Integer.parseInt(s[1]);  
    I[i] = new Interval(A[i], B[i]);  
}  
  
Arrays.sort(I);  
  
int result = 1;  
int permitStart = 1;  
Interval interval = I[0];  
for (int i = 1; i < N; i++) {  
    if (I[i].a >= permitStart) {  
        result++;  
        if (interval.b >= I[i].a) {  
            permitStart = interval.b + 1;  
        }  
        interval = I[i];  
    }  
}  
  
writer.println(result);  
}  
writer.flush();  
writer.close();  
}  
}
```

EVEN SUBARRAY

```
import java.io.*;  
  
class GFG  
{  
    static int countEvenSum(int arr[],  
                           int n)  
    {  
        int result = 0;  
  
        // Find sum of all subarrays  
        // and increment result if  
        // sum is even  
        for (int i = 0; i <= n - 1; i++)  
        {  
            int sum = 0;  
            for (int j = i; j <= n - 1; j++)  
            {  
                sum = sum + arr[j];  
                if (sum % 2 == 0)  
                    result++;  
            }  
        }  
  
        return (result);  
    }  
  
    // Driver code  
    public static void main (String[] args)  
    {  
        int arr[] = {1, 2, 2,  
                     3, 4, 1};  
        int n = arr.length;  
  
        System.out.print("The Number of Subarrays"+  
                         " with even sum is ");  
  
        System.out.println(countEvenSum(arr, n));  
    }  
}
```