

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        System.out.println("Enter the amount to be withdrawn:");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 setIsEligible(LocalDate now) {
        isEligible = joiningDate.until(now, ChronoUnit.YEARS) >= 5;
    }

    public boolean getIsEligible() {
        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_USERNA
            ME"),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);
}

```

GD HOSPITAL

```

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 calculateTotalBill1(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.calculateTotalBill1(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.calculateTotalBill1(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 calculateTotalBill1(int scan,double med)
    {
        return (consultingFee+scan+med);
    }
}
}

```

```

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 getpacer(){
        return gender;
    }
    public long getpanob(){
        return mobileNumber;
    }

    public void setpaid(String id){
        patientId=id;
    }
    public void setpaname(String name){
        patientname=name;
    }
    public void setpacer(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 succesfully via cheque";

}

else if(obj instanceof Cash ) {

Cash cash = (Cash) obj;

if(cash.payAmount())

message = "Payment done succesfully via cash";

}

else if(obj instanceof Credit ) {

Credit card = (Credit) obj;

if(card.payAmount())

message = "Payment done succesfully via creditcard. Remainig 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(chèque));
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.equalsIgno
reCase("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("pilgrimage")){
            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.next();
        primary.setTheatreid(theatreid);
        sc.nextLine();
        System.out.println("Enter the theatre name");
        String theatrename = sc.next();
        primary.setTheatreName(theatrename);
        sc.nextLine();
        System.out.println("Enter the location");
        String location = sc.next();
        primary.setLocation(location);
        sc.nextLine();
        System.out.println("Entrer the no of screens");
        int screens = sc.nextInt();
        primary.setNoOfScreen(screens);
        System.out.println("Ente 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.next();
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("Entrer the no of screens");
screens = sc.nextInt();
secondary.setNoOfScreen(screens);
System.out.println("Ente 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();
}
}

```

Fixed deposit

```

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();
    }
}
```

Group -1

1. AirVoice - Registration

Grade settings: Maximum grade: 100

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 }
45
```

Main.java

```
1 import java.util.Scanner;
2
3 public class Main {
4
```

```

5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7      Scanner sc=new Scanner(System.in);
8      Customer c=new Customer();
9      System.out.println("Enter the Name:");
10     String name=(sc.nextLine());
11     System.out.println("Enter the ContactNumber:");
12     long no=sc.nextLong();
13     sc.nextLine();
14     System.out.println("Enter the EmailId:");
15     String mail=sc.nextLine();
16
17     System.out.println("Enter the Age:");
18     int age=sc.nextInt();
19     c.setCustomerName(name);
20     c.setContactNumber(no);
21     c.setEmailId(mail);
22     c.setAge(age);
23     System.out.println("Name:"+c.getCustomerName());
24     System.out.println("ContactNumber:"+c.getContactNumber());
25     System.out.println("EmailId:"+c.getEmailId());
26     System.out.println("Age:"+c.getAge());
27
28
29
30     }
31
32 }

```

Grade

Reviewed on Monday, 7 February 2022, 4:45 PM by Automatic grade

Grade 100 / 100

Assessment report

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

=====

2. Payment - Inheritance

Grade settings: Maximum grade: 100

Run: Yes **Evaluate:** Yes

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

Payment Status

Roy is a wholesale cloth dealer who sells cloth material to the local tailors on monthly installments. At the end of each month, he collects the installment amount from all his customers.

Some of his customers pay by Cheque, some pay by Cash and some by Credit Card. He wants to automate this payment process.

Help him to do this by writing a java program.

Requirement 1: Make Payment

The application needs to verify the payment process and display the status report of payment by getting the inputs like due amount, payment mode and data specific to the payment mode from the user and calculate the balance amount.

Component Specification: Payment Class (Parent Class)

Component Name	Type(Class)	Attributes	Methods	Responsibilities
Make payment for EMI amount	Payment	int dueAmount	Include a public getter and setter method	
Make payment for EMI amount	Payment		public boolean payAmount()	The boolean payAmount() method should return true if there is no due to be paid, else return false.

Note:

- The attributes of Payment class should be private.
- The payment can be of three types: Cheque, Cash, Credit Card.

Component Specification: Cheque class (Needs to be a child of Payment class)

Component Name	Type(Class)	Attributes	Methods	Responsibilities
	Cheque	String chequeNo int chequeAmount Date dateOfIssue	Include a public getter and setter method for all the attributes.	

Make payment for EMI amount	Cheque		public boolean payAmount()	This is an overridden method of the parent class. It should return true if the cheque is valid and the amount is valid. Else return false.
-----------------------------	--------	--	----------------------------	--

Note:

- The cheque is valid for 6 months from the date of issue.
- Assume the current date is 01-01-2020 in dd-MM-yyyy format.
- The chequeAmount is valid if it is greater than or equal to the dueAmount.

Component Specification: Cash class (Needs to be a child of Payment class)

Component Name	Type(Class)	Attributes	Methods	Responsibilities
Make payment for EMI amount	Cash	int cashAmount	Include a public getter and setter method for the attribute.	
Make payment for EMI amount	Cash		public boolean payAmount()	This is an overridden method of the parent class. It should return true if the cashAmount is greater than or equal to the dueAmount. Else return false.

Component Specification: Credit class (Needs to be a child of Payment class)

Component Name	Type (Class)	Attributes	Methods	Responsibilities
Make payment for EMI amount	Credit	int creditCardNo String cardType int creditCardAmount	Include a public getter and setter method for all the attributes.	

Make payment for EMI amount	Credit		public boolean payAmount()	This is an overridden method of the parent class. It should deduct the dueAmount and service tax from the creditCardAmount and return true if the credit card payment was done successfully. Else return false.
-----------------------------	--------	--	----------------------------	---

Note:

- The payment can be done if the credit card amount is greater than or equal to the sum of due amount and service tax. Else payment cannot be made.
- The cardType can be “silver” or “gold” or “platinum”. Set the creditCardAmount based on the cardType.
- Also service tax is calculated on dueAmount based on cardType.

Credit Card Type	Credit Card Amount	Service Tax
silver	10000	2% of the due amount
gold	50000	5% of the due amount
platinum	100000	10% of the due amount

- The boolean payAmount() method should deduct the due amount and the service tax amount from a credit card. If the creditCardAmount is less than the dueAmount+serviceTax, then the payment cannot be made.
- The balance in credit card amount after a successful payment should be updated in the creditCardAmount by deducting the sum of dueAmount and serviceTax from creditCardAmount itself.

Component Specification: Bill class

Component Name	Type(Class)	Attributes	Methods	Responsibilities
Payment Status Report	Bill		public String processPayment (Payment obj)	This method should return a message based on the status of the payment made.

Note:

- If the payment is successful, processPayment method should return a message “Payment done successfully via cash” or “Payment done successfully via cheque” or “Payment done successfully via creditcard. Remaining amount in your <<cardType>> card is <<balance in CreditCardAmount>>”
- If the payment is a failure, then return a message “Payment not done and your due amount is <<dueAmount>>”

Create a **public class Main** with the main method to test the application.

Note:

- Assume the current date as 01-01-2020 in dd-MM-yyyy format.
- 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.
- Adhere to the sample input and output.

Sample Input 1:

Enter the due amount:

3000

Enter the mode of payment(cheque/cash/credit):

cash

Enter the cash amount:

2000

Sample Output 1:

Payment not done and your due amount is 3000

Sample Input 2:

Enter the due amount:

3000

Enter the mode of payment(cheque/cash/credit):

cash

Enter the cash amount:

3000

Sample Output 2:

Payment done successfully via cash

Sample Input 3:

Enter the due amount:

3000

Enter the mode of payment(ch cheque/cash/credit):

cheque

Enter the cheque number:

123

Enter the cheque amount:

3000

Enter the date of issue:

21-08-2019

Sample Output 3:

Payment done successfully via cheque

Sample Input 4:

Enter the due amount:

3000

Enter the mode of payment(ch cheque/cash/credit):

credit

Enter the credit card number:

234

Enter the card type(silver,gold,platinum):

silver

Sample Output 4:

Payment done successfully via credit card. Remaining amount in your silver card is 6940

Automatic evaluation[+]

Main.java

```
1 import java.text.ParseException;
2 import java.text.SimpleDateFormat;
3 import java.util.Date;
4 import java.util.Scanner;
5 public class Main {
6
7     public static void main(String[] args) {
8
9         Scanner sc=new Scanner(System.in);
10        System.out.println("Enter the due amount:");
11        int dueAmount=sc.nextInt();
12
13        System.out.println("Enter the mode of payment(cheque/cash/credit):");
14        String mode=sc.next();
15        Bill b = new Bill();
16        if(mode.equals("cheque"))
17        {
18            System.out.println("enter the cheque number:");
19            String chequeNumber=sc.next();
20            System.out.println("enter the cheque amount:");
21            int chequeAmount=sc.nextInt();
22            System.out.println("enter the date of issue:");
23            String date=sc.next();
24            SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy");
25            Date dateOfIssue=null;
26            try
27            {
28                dateOfIssue = dateFormat.parse(date);
29            }
30            catch (ParseException e)
31            {
32
33            }
34            Cheque cheque= new Cheque();
35            cheque.setChequeNo(chequeNumber);
36            cheque.setChequeAmount(chequeAmount);
37            cheque.setDateOfIssue(dateOfIssue);
38            cheque.setDueAmount(dueAmount);
39            System.out.println(b.processPayment(cheque));
40        }
41        else if(mode.equals("cash"))
42        {
43            System.out.println("enter the cash amount:");
44            int CashAmount=sc.nextInt();
45            Cash cash=new Cash();
46            cash.setCashAmount(CashAmount);
47            cash.setDueAmount(dueAmount);
48            System.out.println(b.processPayment(cash));
49        }
50        else if(mode.equals("credit"))
51        {
52            System.out.println("enter the credit card number:");
53            int creditCardNumber=sc.nextInt();
54            System.out.println("enter the card type:");
55            String cardType=sc.next();
56
```

```

57         Credit credit=new Credit();
58         credit.setCreditCardNo(creditCardNumber);
59         credit.setCardType(cardType);
60         credit.setDueAmount(dueAmount);
61         System.out.println(b.processPayment(credit));
62     }
63 }
64 }

```

Payment.java

```

1 public class Payment {
2     private int dueAmount;
3
4     public boolean payAmount()
5     {
6         if(dueAmount == 0)
7             return true;
8         else
9             return false;
10    }
11
12    public int getDueAmount() {
13        return dueAmount;
14    }
15
16    public void setDueAmount(int dueAmount) {
17        this.dueAmount = dueAmount;
18    }
19 }

```

Cheque.java

```

1 import java.text.ParseException;
2 import java.text.SimpleDateFormat;
3 import java.util.Date;
4 public class Cheque extends Payment {
5     String chequeNo;
6     int chequeAmount;
7     Date dateOfIssue;
8     public String getChequeNo() {
9         return chequeNo;
10    }
11    public void setChequeNo(String chequeNo) {
12        this.chequeNo = chequeNo;
13    }
14    public int getChequeAmount() {
15        return chequeAmount;
16    }
17    public void setChequeAmount(int chequeAmount) {
18        this.chequeAmount = chequeAmount;
19    }
20    public Date getDateOfIssue() {
21        return dateOfIssue;
22    }
23    public void setDateOfIssue(Date dateOfIssue) {
24        this.dateOfIssue = dateOfIssue;
25    }
26
27    @Override
28    public boolean payAmount()
29    {
30        SimpleDateFormat format = new SimpleDateFormat("dd-MM-yyyy");
31        Date today = new Date();
32        try
33        {
34            today = format.parse("01-01-2020");

```

```

35         }
36         catch (ParseException e)
37         {
38             return false;
39         }
40         long diff = today.getTime()-dateOfIssue.getTime();
41         int day = (int) Math.abs(diff/(1000*60*60*24));
42         int month = day/30;
43         if(month <=6)
44         {
45
46             if(chequeAmount>=getDueAmount())
47             {
48                 return true;
49             }
50             else
51                 return false;
52
53         }
54         else
55             return false;
56     }
57
58 }
59 }

```

Cash.java

```

1 public class Cash extends Payment {
2     int cashAmount;
3
4     public int getCashAmount() {
5         return cashAmount;
6     }
7
8     public void setCashAmount(int cashAmount) {
9         this.cashAmount = cashAmount;
10    }
11
12    @Override
13    public boolean payAmount()
14    {
15        if(cashAmount>=getDueAmount())
16            return true;
17        else
18            return false;
19    }
20
21
22 }

```

Credit.java

```

1 public class Credit extends Payment {
2     int creditCardNo;
3     String cardType;
4     int creditCardAmount;
5     public int getCreditCardNo() {
6         return creditCardNo;
7     }
8     public void setCreditCardNo(int creditCardNo) {
9         this.creditCardNo = creditCardNo;
10    }
11    public String getCardType() {
12        return cardType;
13    }
14    public void setCardType(String cardType) {

```

```

15         this.cardType = cardType;
16     }
17     public int getCreditCardAmount() {
18         return creditCardAmount;
19     }
20     public void setCreditCardAmount(int creditCardAmount) {
21         this.creditCardAmount = creditCardAmount;
22     }
23
24
25     @Override
26     public boolean payAmount()
27     {
28         int netAmount = 0;
29         if(cardType.equals("silver"))
30         {
31             netAmount = (int) (getDueAmount()*1.02);
32             creditCardAmount = 10000;
33         }
34         else if(cardType.equals("gold"))
35         {
36             netAmount = (int) (getDueAmount()*1.05);
37             creditCardAmount = 50000;
38         }
39         else if(cardType.equals("platinum"))
40         {
41             netAmount = (int) (int) (getDueAmount()*1.1);
42             creditCardAmount = 100000;
43         }
44
45         if(creditCardAmount>=netAmount)
46         {
47             creditCardAmount = creditCardAmount - netAmount;
48             return true;
49         }
50         else
51             return false;
52     }
53
54
55 }

```

Bill.java

```

1 public class Bill {
2     public String processPayment(Payment obj)
3     {
4         String res="";
5         if(obj instanceof Cheque)
6         {
7             if(obj.payAmount())
8                 res = "Payment done successfully via cheque";
9             else
10                 res = "Payment not done and your due amount is
"+obj.getDueAmount();
11         }
12         else if(obj instanceof Cash)
13         {
14             if(obj.payAmount())
15                 res = "Payment done successfully via cash";
16             else
17                 res = "Payment not done and your due amount is
"+obj.getDueAmount();
18         }
19         else if(obj instanceof Credit)
20         {
21             Credit c = (Credit) obj;

```



```

22                                     if(obj.payAmount())
23                                     res = "Payment done successfully via credit card. Remaining
amount in your "+c.getCardType()+" card is "+c.getCreditCardAmount();
24                                     else
25                                     res = "Payment not done and your due amount is
"+obj.getDueAmount();
26                                     }
27                                     return res;
28     }
29 }

```

Grade

Reviewed on Wednesday, 1 December 2021, 10:08 PM by Automatic grade

Grade 100 / 100

Assessment report

TEST CASE PASSED

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

3.Power Progress

Grade settings: Maximum grade: 100

Run: Yes **Evaluate:** Yes

Automatic grade: Yes

Andrews taught exponential multiplication to his daughter and gave her two inputs.

Assume, the first input as M and the second input as N. He asked her to find the sequential power of M until N times. For Instance, consider M as 3 and N as 5. Therefore, 5 times the power is incremented gradually from 1 to 5 such that, $3^1=3$, $3^2=9$, $3^3=27$, $3^4=81$, $3^5=243$. The input numbers should be greater than zero Else print "<Input> is an invalid". The first Input must be less than the second Input, Else print "<first input> is not less than <second input>".

Write a Java program to implement this process programmatically and display the output in sequential order. (3^3 means $3*3*3$).

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.

Adhere to the code template, if provided.

Kindly do not use System.exit() in the code.

Sample Input 1:

3

5

Sample Output 1:

3 9 27 81 243

Explanation: Assume the first input as 3 and second input as 5. The output is to be displayed are based on the sequential power incrementation. i.e., $3(3)$ $9(3*3)$ $27(3*3*3)$ $81(3*3*3*3)$ $243(3*3*3*3*3)$

Sample Input 2:

-3

Sample Output 2:

-3 is an invalid

Sample Input 3:

3

0

Sample Output 3:

0 is an invalid

Sample Input 4:

4

2

Sample Output 4:

4 is not less than 2

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

Main.java

```
1 import java.util.*;
2 public class Main
3 {
4     public static void main(String[] args)
5     {
6         Scanner sc=new Scanner(System.in);
7         //Fill the code
8         int m=sc.nextInt();
9         if(m<=0){
10             System.out.println(""+m+" is an invalid");
11             return;
12         }
13         int n=sc.nextInt();
14         if(n<=0){
15             System.out.println(""+n+" is an invalid");
16             return;
17         }
18         if(m>=n){
19             System.out.println(""+m+" is not less than "+n);
20             return;
21         }
22         for(int i=1;i<=n;i++){
```

```

23      System.out.print((int)Math.pow(m,i)+"");
24      }
25  }
26 }

```

Grade

Reviewed on Monday, 7 February 2022, 4:46 PM by Automatic grade

Grade 100 / 100

Assessment report

TEST CASE PASSED

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

4. ZeeZee bank

Grade settings: Maximum grade: 100

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 intialize 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)	<p>This method takes the amount to be deposited as an argument</p> <p>This method should perform the deposit, by adding the deposited amount to the balanceAmount</p>

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)	<p>This method should take the amount to be withdrawn as an argument.</p> <p>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.</p>

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:1 6500.00

Enter the amount to be withdrawn:

18500

Insufficient balance

Available balance is:1 6500.00

Automatic evaluation[+]

Main.java

```
1 import java.text.DecimalFormat;
2 import java.util.Scanner;
3 import java.util.Scanner;
4
5
6 public class Main{
7     static Account ac=new Account(0, 0);
8     public static void main (String[] args) {
9         Scanner sc=new Scanner(System.in);
10        System.out.println("Enter the account number:");
11        ac.setAccountNumber(sc.nextLong());
12        System.out.println("Enter the available amount in the account:");
13        ac.setBalanceAmount(sc.nextDouble());
14        System.out.println("Enter the amount to be deposited:");
15        ac.deposit(sc.nextDouble());
16        System.out.printf("Available balance is:%.2f",ac.getBalanceAmount());
17        System.out.println();
18        System.out.println("Enter the amount to be withdrawn:");
19        ac.withdraw(sc.nextDouble());
20        System.out.printf("Available balance is:%.2f",ac.getBalanceAmount());
21        //Fill the code
22    }
23 }
24
25
26
```

Account.java

```
1
2 public class Account {
3     long accountNumber;
4     double balanceAmount;
5
6
7     public Account(long accno, double bal){
```

```

8    super();
9    this.accountNumber=accno;
10   this.balanceAmount=bal;
11 }
12 public long getAccountNumber(){
13     return accountNumber;
14 }
15 public void setAccountNumber(long accno){
16     this.accountNumber=accno;
17 }
18 public double getBalanceAmount(){
19     return balanceAmount;
20 }
21 public void setBalanceAmount(double bal) {
22     this.balanceAmount=bal;
23 }
24 public void deposit(double depositAmt){
25     float total=(float)(balanceAmount+depositAmt);
26     balanceAmount=total;
27 }
28 public boolean withdraw(double withdrawAmt){
29     float total;
30     if(withdrawAmt>balanceAmount){
31         System.out.println("Insufficient balance");
32
33         return false;
34     }else{
35         total=(float)(balanceAmount-withdrawAmt);
36         setBalanceAmount(total);
37         return true;
38     }
39 }
40 }

```

Grade

Reviewed on Monday, 7 February 2022, 4:47 PM by Automatic grade

Grade 100 / 100

Assessment report

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

5. Reverse a word

Grade settings: Maximum grade: 100

Run: Yes **Evaluate:** Yes

Automatic grade: Yes

Reverse a word

Rita and Brigitha want to play a game. That game is to check the first letter of each word in a given sentence (Case Insensitive). If it is equal, then reverse the last word and concatenate the first word. Else reverse the first word and concatenate the last word. Create a Java application and help them to play the game

Note:

- Sentence must contain at least 3 words else print "Invalid Sentence" and terminate the program
- Each word must contain alphabet only else print "Invalid Word" and terminate the program
- Check the first letter of each word in a given sentence (Case Insensitive). If it is equal, then reverse the last word and concatenate the first word and print. Else reverse the first word and concatenate the last word and print.
- Print the output without any space.

Please do not use `System.exit(0)` to terminate the program

Sample Input 1:

Sea sells seashells

Sample Output 1:

sllehsaesSea

Sample Input 2:

Sam is away from Australia for a couple of days

Sample Output 2:

maSdays

Sample Input 3:

Welcome home

Sample Output 3:

Invalid Sentence

Sample Input 4:

Friendly fire fighting fr@gs.

Sample Output 4:

Invalid Word

Automatic evaluation[+]

Main.java

```
1 import java.util.Scanner;
2 import java.lang.String.*;
3 import java.util.*;
4 public class Main{
5     public static void main(String[] args){
6         String[] words;
7         Scanner read =new Scanner(System.in);
8         String sentence=read.nextLine();
9         words=sentence.split(" ");
10        if(words.length<3)
11            System.out.println("Invalid Sentence");
12        else{
13            String a=words[0].substring(0,1);
14            String b=words[1].substring(0,1);
15            String c=words[2].substring(0,1);
16            if(a.equalsIgnoreCase(b)&&b.equalsIgnoreCase(c))
17                {
18                    StringBuilder k= new StringBuilder();
19                    k.append(words[words.length-1]);
20                    k=k.reverse();
21                    k.append(words[0]);
22                    System.out.println(k);
23                }
24            else{
25                StringBuilder k = new StringBuilder();
26                k.append(words[0]);
27                k=k.reverse();
28                k.append(words[words.length-1]);
29                System.out.println(k);
30            }
31        }
32    }
33 }
```

Grade

Reviewed on Monday, 7 February 2022, 5:12 PM by Automatic grade

Grade 90 / 100

Assessment report

Fail 1 -- test5_CheckForTheSentenceContainsOtherThanAlphabets::
\$Expected output:"[Invalid Word]" Actual output:"[tahwme]"\$
[\[+\]Grading and Feedback](#)

6. Dominion cinemas

Grade settings: Maximum grade: 100

Run: Yes **Evaluate:** Yes

Automatic grade: Yes

Dominion cinema is a famous theatre in the city. It has different types of seat tiers – Platinum, Gold and Silver. So far the management was manually calculating the ticket cost for all their customers which proved very hectic and time consuming. Going forward they want to calculate ticket cost using their main computer. Assist them in calculating and retrieving the amount to be paid by the Customer.

Requirements 1: Calculation of Ticket Cost

The application needs to calculate the ticket cost to be paid by the Customer according to the seat tier.

Component Specification: BookAMovieTicket Class (Parent Class)

Component Name	Type(Class)	Attributes	Methods	Responsibilities
Calculation of Ticket cost	BookAMovieTicket	String ticketId String customerName long mobileNumber String emailId String movieName	Public getter and setter method for all the attributes and 5 argument constructor in the given order - ticketId, customerName, mobileNumber, emailId, movieName are provided as a part of the code skeleton.	

Note:

- The attributes of the BookAMovieTicket class should be protected.

Component Specification: GoldTicket class (Needs to be a child of BookAMovieTicket class)

Component Name	Type(Class)	Attributes	Methods	Responsibilities
Calculation of Ticket cost	GoldTicket		Include a public 5 argument constructor in the given order - ticketId, customerName, mobileNumber, emailId, movieName.	
Validate Ticket Id	GoldTicket		public boolean validateTicketId ()	This method should validate the Ticket Id, Ticket Id should contain a string “GOLD” followed by 3 digits. If the ticket id is valid this method should return true else it should return false.
Calculation of Ticket cost	GoldTicket		public double calculateTicketCost (int numberOfTickets, String ACFacility)	This method should calculate the ticket cost according to the seat tier and return the same.

Component Specification: PlatinumTicket class (Needs to be a child of the BookAMovieTicket class)

Component Name	Type(Class)	Attributes	Methods	Responsibilities
Calculation of Ticket cost	PlatinumTicket		Include a public 5 argument constructor in the given order - ticketId, customerName, mobileNumber, emailId, movieName.	
Validate Ticket Id	PlatinumTicket		public boolean validateTicketId()	This method should validate the Ticket Id, Ticket Id should contain a string “PLATINUM” followed by 3 digits. If the ticket id is

				valid this method should return true else it should return false.
Calculation of Ticket cost	PlatinumTicket		calculateTicketCost(int numberOfTickets, String ACFacility)	This method should calculate the ticket cost according to the seat tier and return the same.

Component Specification: SilverTicket class (Needs to be a child of the BookAMovieTicket class)

Component Name	Type(Class)	Attributes	Methods	Responsibilities
Calculation of Ticket cost	SilverTicket		Include a public 5 argument constructor in the given order - ticketId, customerName, mobileNumber, emailId, movieName.	
Validate Ticket Id	SilverTicket		public boolean validateTicketId()	This method should validate the Ticket Id, Ticket Id should contain a string "SILVER" followed by 3 digits. If the ticket id is valid this method should return true else it should return false.
Calculation of Ticket cost	SilverTicket		calculateTicketCost(int numberOfTickets, String ACFacility)	This method should calculate the ticket cost according to the seat tier and return the same.

Note:

- The classes GoldTicket, PlatinumTicket and SilverTicket should be concrete classes.

Ticket cost according to the seat tier without AC facilities.

Seat Tier	Silver	Gold	Platinum
-----------	--------	------	----------

Without AC Facility	100	350	600
With AC Facility	250	500	750

Amount is calculated based on the seat tier,

Amount = ticketCost * numberOfTickets

Use a **public class UserInterface** with the main method to test the application. In the main method call the validateTicketId() method, if the method returns true display the amount else display "**Provide valid Ticket Id**".

Note:

- **Display the amount to be paid to 2 decimal places.**
- **Use the System.out.printf method.**
- 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 Ticket Id

SILVER490

Enter Customer name

Venkat

Enter Mobile number

9012894578

Enter Email Id

venkat@gmail.com

Enter Movie name

Avengers

Enter number of tickets

8

Do you want AC or not

yes // Case insensitive

Ticket cost is 2000.00

Sample Input 2:

Enter Ticket Id

ACN450

Enter Customer name

Kamal

Enter Mobile number

9078561093

Enter Email Id

kamal@gmail.com

Enter Movie name

Tangled

Enter number of tickets

9

Provide valid Ticket Id

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

BookAMovieTicket.java

```
1
2 public class BookAMovieTicket {
3
4     protected String ticketId;
5     protected String customerName;
6     protected long mobileNumber;
7     protected String emailId;
8     protected String movieName;
9
10    public String getTicketId() {
11        return ticketId;
```

```

12     }
13     public void setTicketId(String ticketId) {
14         this.ticketId = ticketId;
15     }
16     public String getCustomerName() {
17         return customerName;
18     }
19     public void setCustomerName(String customerName) {
20         this.customerName = customerName;
21     }
22     public long getMobileNumber() {
23         return mobileNumber;
24     }
25     public void setMobileNumber(long mobileNumber) {
26         this.mobileNumber = mobileNumber;
27     }
28     public String getEmailId() {
29         return emailId;
30     }
31     public void setEmailId(String emailId) {
32         this.emailId = emailId;
33     }
34     public String getMovieName() {
35         return movieName;
36     }
37     public void setMovieName(String movieName) {
38         this.movieName = movieName;
39     }
40
41     public BookAMovieTicket(String ticketId, String customerName, long mobileNumber, String emailId,
String movieName) {
42         this.ticketId = ticketId;
43         this.customerName = customerName;
44         this.mobileNumber = mobileNumber;
45         this.emailId = emailId;
46         this.movieName = movieName;
47     }
48 }
49
50
51
52 }
53

```

GoldTicket.java

```

1
2 public class GoldTicket extends BookAMovieTicket{
3     public GoldTicket(String ticketId,String customerName, long mobileNumber,
4     String emailId,String movieName){
5         super(ticketId, customerName, mobileNumber, emailId, movieName);
6     }
7
8     public boolean validateTicketId(){
9         int count=0;
10        if(ticketId.contains("GOLD"));
11        count++;

```

```

12         char[] cha=ticketId.toCharArray();
13         for(int i=4;i<7;i++){
14             if(cha[i]>='1'&& cha[i]<='9')
15                 count++;
16         }
17         if(count==4)
18             return true;
19         else
20             return false;
21     }
22
23
24     // Include Constructor
25
26     public double calculateTicketCost(int numberOfTickets, String ACFacility){
27         double amount;
28         if(ACFacility.equals("yes")){
29             amount=500*numberOfTickets;
30         }
31         else{
32             amount=350*numberOfTickets;
33         }
34
35         return amount;
36     }
37
38 }

```

PlatinumTicket.java

```

1 public class PlatinumTicket extends BookAMovieTicket{
2     public PlatinumTicket(String ticketId, String customerName, long mobileNumber,
3         String emailId, String movieName){
4         super(ticketId, customerName, mobileNumber, emailId, movieName);
5     }
6
7     public boolean validateTicketId(){
8         int count=0;
9         if(ticketId.contains("PLATINUM"));
10            count++;
11            char[] cha=ticketId.toCharArray();
12            for(int i=8;i<11;i++){
13                if(cha[i]>='1'&& cha[i]<='9')
14                    count++;
15            }
16            if(count==4)
17                return true;
18            else
19                return false;
20        }
21
22        // Include Constructor
23
24        public double calculateTicketCost(int numberOfTickets, String ACFacility){
25            double amount;
26            if(ACFacility.equalsIgnoreCase("yes")){
27                amount=750*numberOfTickets;

```



```

28         }
29         else{
30             amount=600*numberOfTickets;
31         }
32
33         return amount;
34     }
35
36 }
37

```

SilverTicket.java

```

1
2 public class SilverTicket extends BookAMovieTicket{
3     public SilverTicket(String ticketId, String customerName, long mobileNumber,
4     String emailId, String movieName){
5         super(ticketId, customerName, mobileNumber, emailId, movieName);
6     }
7
8     public boolean validateTicketId(){
9         int count=0;
10        if(ticketId.contains("SILVER"));
11        count++;
12        char[] cha=ticketId.toCharArray();
13        for(int i=6;i<9;i++){
14            if(cha[i]>='1'&& cha[i]<='9')
15                count++;
16        }
17        if(count==4)
18            return true;
19        else
20            return false;
21    }
22
23    // Include Constructor
24
25    public double calculateTicketCost(int numberOfTickets, String ACFacility){
26        double amount;
27        if(ACFacility.equals("yes")){
28            amount=250*numberOfTickets;
29        }
30        else{
31            amount=100*numberOfTickets;
32        }
33
34        return amount;
35    }
36
37 }
38

```

UserInterface.java

```

1 import java.util.*;
2
3 public class UserInterface {

```

```

4
5     public static void main(String[] args){
6         Scanner sc=new Scanner(System.in);
7         System.out.println("Enter Ticket Id");
8         String tid=sc.next();
9         System.out.println("Enter Customer name");
10        String cnm=sc.next();
11        System.out.println("Enter Mobile number");
12        long mno=sc.nextLong();
13        System.out.println("Enter Email id");
14        String email=sc.next();
15        System.out.println("Enter Movie name");
16        String mnm=sc.next();
17        System.out.println("Enter number of tickets");
18        int tno=sc.nextInt();
19        System.out.println("Do you want AC or not");
20        String choice =sc.next();
21        if(tid.contains("PLATINUM")){
22            PlatinumTicket PT= new PlatinumTicket(tid,cnm,mno,email,mnm);
23            boolean b1=PT.validateTicketId();
24            if(b1==true){
25                double cost=PT.calculateTicketCost(tno, choice);
26                System.out.println("Ticket cost is "+String.format("%.2f",cost));
27            }
28            else if(b1==false){
29                System.out.println("Provide valid Ticket Id");
30                System.exit(0);
31            }
32        }
33        else if(tid.contains("GOLD")){
34            GoldTicket GT= new GoldTicket(tid,cnm,mno,email,mnm);
35            boolean b2=GT.validateTicketId();
36            if(b2==true){
37                double cost=GT.calculateTicketCost(tno,choice);
38                System.out.println("Ticket cost is "+String.format("%.2f",cost));
39            }
40            else if (b2==false){
41                System.out.println("Provide valid Ticket Id");
42                System.exit(0);
43            }
44        }
45        else if(tid.contains("SILVER")){
46            SilverTicket ST= new SilverTicket(tid,cnm,mno,email,mnm);
47            boolean b3=ST.validateTicketId();
48            if(b3==true){
49                double cost=ST.calculateTicketCost(tno,choice);
50                System.out.println("Ticket cost is "+String.format("%.2f",cost));
51            }
52            else if (b3==false){
53                System.out.println("Provide valid Ticket Id");
54                System.exit(0);
55            }
56        }
57    }
58 }
59
60

```

Grade

Reviewed on Monday, 7 February 2022, 4:18 PM by Automatic grade

Grade 100 / 100

Assessment report

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

=====

Group-2

1. Flight record retrieval

Grade settings: Maximum grade: 100

Based on: [JAVA CC JDBC - MetaData V1 - ORACLE \(w/o Proj Struc\)](#)

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)	Attributes	Methods	Responsibilities
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	FlightManagementSystem		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	integer
source	varchar2(30)
destination	varchar2(30)
noofseats	integer
flightfare	double

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  
2 public class Flight {
```

```

3
4     private int flightId;
5     private String source;
6     private String destination;
7     private int noOfSeats;
8     private double flightFare;
9     public int getFlightId() {
10         return flightId;
11     }
12     public void setFlightId(int flightId) {
13         this.flightId = flightId;
14     }
15     public String getSource() {
16         return source;
17     }
18     public void setSource(String source) {
19         this.source = source;
20     }
21     public String getDestination() {
22         return destination;
23     }
24     public void setDestination(String destination) {
25         this.destination = destination;
26     }
27     public int getNoOfSeats() {
28         return noOfSeats;
29     }
30     public void setNoOfSeats(int noOfSeats) {
31         this.noOfSeats = noOfSeats;
32     }
33     public double getFlightFare() {
34         return flightFare;
35     }
36     public void setFlightFare(double flightFare) {
37         this.flightFare = flightFare;
38     }
39     public Flight(int flightId, String source, String destination,
40                 int noOfSeats, double flightFare) {
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.util.ArrayList;
2 import java.sql.*;
3
4
5 public class FlightManagementSystem {
6
7     public ArrayList<Flight> viewFlightBySourceDestination(String source, String destination){
8         ArrayList<Flight> flightList = new ArrayList<Flight>();
9         try{
10             Connection con = DB.getConnection();
11
12             String query="SELECT * FROM flight WHERE source= " + source + " AND destination= " +
destination + " ";
13
14             Statement st=con.createStatement();

```

```

15
16     ResultSet rst= st.executeQuery(query);
17
18     while(rst.next()){
19         int flightId= rst.getInt(1);
20         String src=rst.getString(2);
21         String dst=rst.getString(3);
22         int noofseats=rst.getInt(4);
23         double flightfare=rst.getDouble(5);
24
25         flightList.add(new Flight(flightId, src, dst, noofseats, flightfare));
26     }
27 }catch(ClassNotFoundException | SQLException e){
28     e.printStackTrace();
29 }
30 return flightList;
31 }
32
33 }

```

Main.java

```

1 import java.util.Scanner;
2 import java.util.ArrayList;
3
4 public class Main{
5     public static void main(String[] args){
6         Scanner sc=new Scanner(System.in);
7         System.out.println("Enter the source");
8         String source=sc.next();
9         System.out.println("Enter the destination");
10        String destination=sc.next();
11
12        FlightManagementSystem fms= new FlightManagementSystem();
13        ArrayList<Flight> flightList=fms.viewFlightBySourceDestination(source,destination);
14        if(flightList.isEmpty()){
15            System.out.println("No flights available for the given source and destination");
16            return;
17        }
18        System.out.println("Flightid Noofseats Flightfare");
19        for(Flight flight : flightList){
20            System.out.println(flight.getFlightId()+" "+flight.getNoOfSeats()+" "+flight.getFlightFare());
21        }
22
23    }
24 }

```

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 =
DriverManager.getConnection(props.getProperty("DB_URL"),props.getProperty("DB_USERNAME"),props.getPr
operty("DB_PASSWORD"));
27     }
28     catch(IOException e){
29         e.printStackTrace();
30     }
31     return con;
32 }
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=oracle.jdbc.driver.OracleDriver
8 DB_URL=jdbc:oracle:thin:@127.0.0.1:1521:XE
9 DB_USERNAME=${sys:db_username}
10 DB_PASSWORD=${sys:db_password}
11

```

Grade

Reviewed on Monday, 7 February 2022, 6:33 PM by Automatic grade

Grade 100 / 100

Assessment report

Assessment Completed Successfully

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

=====

2. Get Text and Display Welcome Message

Grade settings: Maximum grade: 100

Run: Yes **Evaluate:** Yes

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

Amir owns "Bouncing Babies" an exclusive online store for baby toys.

He desires to display a welcome message whenever a customer visits his online store and makes a purchase.

Help him do this by incorporating the customer name using the Lambda expression.

Requirement 1: Display Welcome message

Amir wants to display a welcome message for his customers. The method displayText is used to display the name of the customer who made an online purchase from his store.

Component Specification: DisplayText Interface – This is a Functional Interface.

Type(Interface)	Methods	Responsibilities
DisplayText	public void displayText(String text)	The purpose of this method is to display the welcome message by including the text provided as an argument by using Lambda expression.
DisplayText	public default String getInput()	This method should get a String (name of the customer) as input from the user and return the same. This method should be a default method.

Annotate the interface with the appropriate annotation

Component Specification: Main class

Component Name	Type(Class)	Methods	Responsibilities
Display welcome message	Main	public static DisplayText welcomeMessage()	This method should return a DisplayText object. To do this, implement the lambda expression to print the text received as a parameter in the displayText method as "Welcome <text>".

In the Main class write the main method and perform the given steps :

- Invoke the static method welcomeMessage(). It returns a DisplayText object.
- Capture the DisplayText object in a reference variable.
- Using that reference, invoke the default method getInput.
- It will return a String. Capture that String in a variable.
- Using the reference of DisplayText, invoke the displayText method by passing the String as a parameter.
- The output should be as shown in the sample data mentioned below.

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 name for classes, interfaces and methods as specified in the question.

Adhere to the code template, if provided.

Sample Input 1 :

Watson

Sample Output 1 :

Welcome Watson

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

DisplayText.java

```
1 import java.util.*;
2 @FunctionalInterface
3 public interface DisplayText
4 {
5     public void displayText(String text);
6     public default String getInput()
7     {
8         Scanner read = new Scanner(System.in);
9         String str = read.next();
10        return str;
11        //return null;
12    }
13 }
```

Main.java

```
1 public class Main
2 {
3     public static DisplayText welcomeMessage()
4     {
5
6         DisplayText dis = (str)->{
7
8             System.out.println("Welcome "+str);
9         };
10        return dis;
11    }
12    public static void main(String args[])
13    {
14        DisplayText dis=welcomeMessage();
15        String text = dis.getInput();
16        dis.displayText(text);
17    }
18 }
19 }
```

Grade

Reviewed on Wednesday, 1 December 2021, 10:14 PM by Automatic grade

Grade 100 / 100

Assessment report

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

=====

3. Generate Password

Grade settings: Maximum grade: 100

Run: Yes **Evaluate:** Yes

Automatic grade: Yes

Important Instructions:

- Please read the document thoroughly before you code.
- Import the given skeleton code into your Eclipse.(if provided)
- Do not change the Skeleton code or the package structure, method names, variable names, return types, exception clauses, access specifiers etc.
- You can create any number of private methods inside the given class.
- You can test your code from main() method of the program

The system administrator of an organization wants to set password for all the computers for security purpose. To generate a strong password, he wants to combine the username of each user of the system with the reverse of their respective usernames. Help them by using Lambda expressions that caters to their requirement.

Requirement 1: PasswordInfo

The Administrator wants to generate password for each system by making use of the passwordGeneration method based on the username which is passed as a string.

Component Specification: Password Info Interface – This is a Functional Interface.

Type(Interface)	Methods	Responsibilities
PasswordInfo	public String passwordGeneration(String username)	This method is used to generate the password based on the username and hence returns the generated password

Component Specification: Computer Class

Type(Class)	Methods	Responsibilities
Computer	public static PasswordInfo passwordPropagation()	This method should return a PasswordInfo object. To do this, implement the lambda expression to get the password.
	public static void displayUserDetails(String systemNo,String username>PasswordInfo passwordInfoObj)	This method is used to print the Password Info such as the systemNo, password along with the message, “Your password is generated successfully!!!” based

		on the systemNo, username, passwordInfoObj which is passed as an argument.
--	--	---

In the Computer class write the main method and perform the given steps:

- Get the systemNo and username from the user.
- Invoke the static method passwordPropagation(). It returns a passwordInfo object with the definition of the passwordGeneration method.
- Capture the PasswordInfo object in a reference variable.
- Invoke the displayUserDetails method by passing systemNo, username and passwordInfoObj as parameters.
- Inside the userDetails method, you should invoke the passwordGeneration method using the passwordInfo object and the output should be displayed as shown in the sample input/output.
- The output should be as shown in the sample data mentioned below.

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 use the lambda expression.
- Ensure to follow the object oriented specifications provided in the question.
- Ensure to provide the name for classes, interfaces and methods as specified in the question.
- Adhere to the code template, if provided.

Sample Input 1:

Enter system no

Tek/1234

Enter username

Manoj Kumar

Sample Output 1:

Password Info

System no: Tek/1234

Password: Manoj KumarramuK jonaM

Your password is generated successfully!!!

=====

4. Vatican Museum Manipulation

Grade settings: Maximum grade: 100

Run: Yes **Evaluate:** Yes

Automatic grade: Yes **Maximum execution time:** 60 s **Maximum memory used:** 64

MiB **Maximum execution file size:** 320 KiB

Important Instructions:

- Please read the document thoroughly before you code.
- Import the given skeleton code into your Eclipse.(if provided)
- Do not change the Skeleton code or the package structure, method names, variable names, return types, exception clauses, access specifiers etc.
- You can create any number of private methods inside the given class.
- You can test your code from the main() method of the program.

Vatican Museum is one of the famous museums, they have collections of houses paintings, and sculptures from artists. The Museum management stores their visitor's details in a text file. Now, they need an application to analyze and manipulate the visitor details based on the visitor visit date and the visitor address.

You are provided with a text file – VisitorDetails.txt, which contains all the visitor details like the visitor Id, visitor name, mobile number, date of visiting and address. Your application should satisfy the following requirements.

1. View visitor details within two given dates.
2. View visitor details which are above a particular mentioned visitor address.

You are provided with a code template which includes the following:

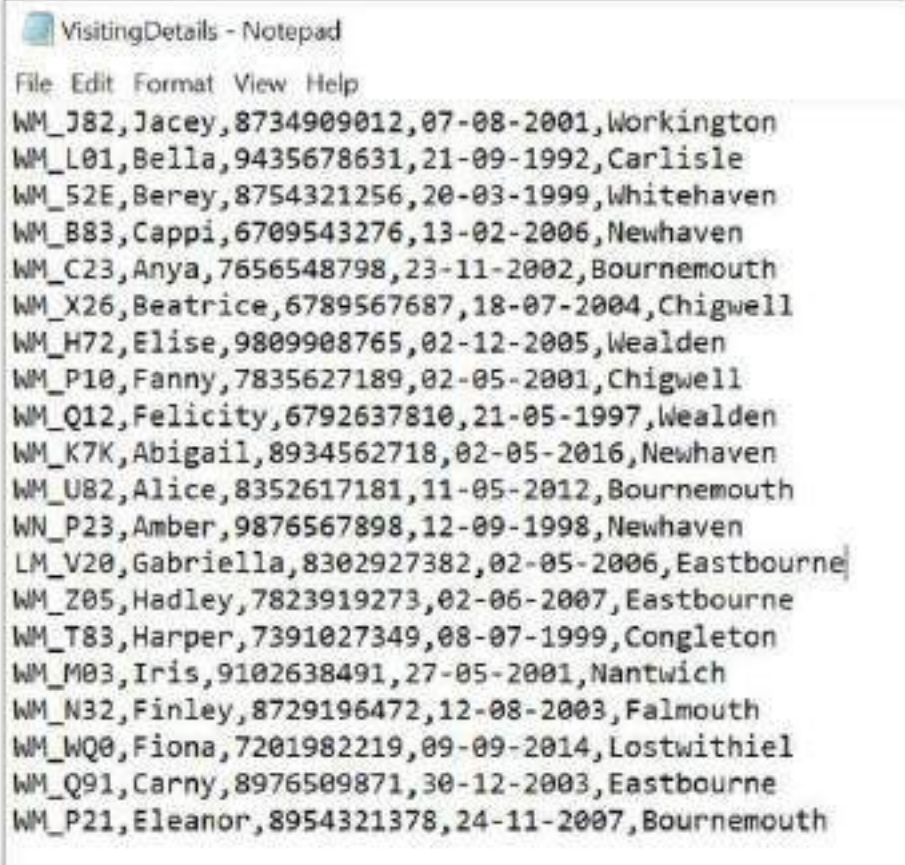
- Visitor class which includes the attributes visitorId, visitorName, mobileNumber, dateOfVisiting and address with all the getters and setters.
- VisitorUtility class which includes the following method declarations.
 - public List<Visitor> generateVisitor(String filePath)
 - public boolean isValidVisitorId(String visitorId)
 - public List<Visitor> viewVisitorDetailsByDateOfVisiting(Stream<Visitor> visitorStream, String fromDate, String toDate)
 - public Stream<Visitor> viewVisitorDetailsByAddress (Stream<Visitor> visitorStream, double address)
- InvalidVisitorIdException class which inherits the Exception class.
- Main class with a main method which creates the required user interface for the application.
- VisitorDetails.txt which contains all the visitor details like visitor id, visitor name, mobile number, date of visiting and address.

Note:

- The Visitor class and the Main class will be provided with all the necessary codes. Please do not edit or delete any line of the code in these two classes.
- Fill your code in the InvalidVisitorIdException class to create a constructor as described in the functional requirements below.
- Fill your code in the respective methods of VisitorUtility class to fulfil all the functional requirements.

- In the VisitorDetails.txt file, each visitor detail has information separated by a comma, and it is given as one customer detail per line.

Sample data in VisitorDetails.txt file



```

File Edit Format View Help
WM_J82,Jacey,8734909012,07-08-2001,Workington
WM_L01,Bella,9435678631,21-09-1992,Carlisle
WM_52E,Berey,8754321256,20-03-1999,Whitehaven
WM_B83,Cappi,6709543276,13-02-2006,Newhaven
WM_C23,Anya,7656548798,23-11-2002,Bournemouth
WM_X26,Beatrice,6789567687,18-07-2004,Chigwell
WM_H72,Elise,9809908765,02-12-2005,Wealden
WM_P10,Fanny,7835627189,02-05-2001,Chigwell
WM_Q12,Felicity,6792637810,21-05-1997,Wealden
WM_K7K,Abigail,8934562718,02-05-2016,Newhaven
WM_U82,Alice,8352617181,11-05-2012,Bournemouth
WM_P23,Amber,9876567898,12-09-1998,Newhaven
LM_V20,Gabriella,8302927382,02-05-2006,Eastbourne
WM_Z05,Hadley,7823919273,02-06-2007,Eastbourne
WM_T83,Harper,7391027349,08-07-1999,Congleton
WM_M03,Iris,9102638491,27-05-2001,Nantwich
WM_N32,Finley,8729196472,12-08-2003,Falmouth
WM_WQ0,Fiona,7201982219,09-09-2014,Lostwithiel
WM_Q91,Carny,8976509871,30-12-2003,Eastbourne
WM_P21,Eleanor,8954321378,24-11-2007,Bournemouth
  
```

Functional Requirements:

Fill your code in the respective class and method declarations based on the required functionalities as given below.

Class	Attributes/ Methods	Rules/ Responsibility
VisitorUtility	<pre> public List < Visitor> generateVisitor(String filePath) </pre>	<p>Read the text file and convert each line in the text file as String and store it in a List. Each String from the List should be converted into a visitor object and each visitor object should be stored in a List. Return the List of visitors.</p> <p>Note:</p> <p>Before converting the separated string into a visitor object, the identified visitorId should be validated using the</p>

		isValidVisitorId method.
VisitorUtility	public boolean isValidVisitorId (String visitorId)	<p>Should check whether the provided visitorId is valid or not.</p> <p>If valid, this method should return true.</p> <p>If invalid, this method should handle an InvalidVisitorIdException with a message "<visitorId> is Invalid Visitor Id".</p> <p>Validation Rules:</p> <ul style="list-style-type: none"> · Length of the visitorId should be exactly 6. · The visitorId should start with "WM_" and the next letter should be an alphabet (A-Z) in upper case and the last two letters should be positive integers(0-9). <p>Example.</p> <p>WM_A23</p>
InvalidVisitorIdException	Create a constructor with a single String argument and pass it to the parent class constructor.	This class Should inherit the Exception class. The constructor should pass the String message which is thrown to it by calling the parent class constructor.

Requirement 1: View visitor details between the dates of visiting

Class	Attributes/ Methods	Rules/ Responsibility
VisitorUtility	public List<Visitor> viewVisitorDetailsByDateOfVisiting(Stream<Visitor> visitorStream, String fromDate, String toDate)	From the provided Stream of Visitor, separate the visitor details which has the date of visiting between fromDate and toDate (both inclusive). Return the

		separated visitor details as a list.
--	--	--------------------------------------

Requirement 2: View visitor details which are above a particular mentioned address

Class	Attributes/ Methods	Rules/ Responsibility
VisitorUtility	public Stream<Visitor> viewVisitorDetailsByAddress(Stream<Visitor> visitorStream, String address)	From the given Stream of Visitor, separate the visitor details based on address, which has a particular mentioned address as provided. Return the separated Stream of visitor.

Note:

1. All inputs/ outputs for processing the functional requirements should be case sensitive.
2. Adhere to the Sample Inputs/ Outputs
3. In the Sample Inputs/ Outputs provided, the highlighted text in bold corresponds to the input given by the user and the rest of the text represents the output.
4. All the Date values used in this application must be in "dd-MM-yyyy" format.
5. Adhere to the code template.
6. Fill all your required codes in the respective blocks. Do not edit or delete the codes provided in the code template.
7. The Sample Inputs/ Outputs given below are generated based on the Sample data given in the VisitorDetails.txt file.
8. Please do not hard code the output.

Sample Input/ Output 1:

WM_52E is Invalid Visitor Id

WM_K7K is Invalid Visitor Id

WN_P23 is Invalid Visitor Id

LM_V20 is Invalid Visitor Id

WM_WQ0 is Invalid Visitor Id

1. ViewVisitorDetailsByDateOfVisiting

2. ViewVisitorDetailsByAddress

Enter your choice

1

Enter the starting date

19-05-2004

Enter the ending date

07-04-2012

WM_B83 Cappi 6709543276 13-02-2006 Newhaven

WM_X26 Beatrice 6789567687 18-07-2004 Chigwell

WM_H72 Elise 9809908765 02-12-2005 Wealden

WM_Z05 Hadley 7823919273 02-06-2007 Eastbourne

WM_P21 Eleanor 8954321378 24-11-2007 Bournemouth

Sample Input/ Output 2:

WM_52E is Invalid Visitor Id

WM_K7K is Invalid Visitor Id

WN_P23 is Invalid Visitor Id

LM_V20 is Invalid Visitor Id

WM_WQ0 is Invalid Visitor Id

1. viewVisitorDetailsByDateOfVisiting

2. viewVisitorDetailsByAddress

Enter your choice

2

Enter the address

Eastbourne

WM_Z05 Hadley 7823919273 02-06-2007 Eastbourne

WM_Q91 Carny 8976509871 30-12-2003 Eastbourne

5. Hospital Management_Streams

Grade settings: Maximum grade: 100

Run: Yes **Evaluate:** Yes

Automatic grade: Yes

Laxmi Hospital is a world-class health care institution providing patient treatment with specialized medical and nursing staff and medical equipment. It typically provides an emergency department to treat urgent health problems ranging from fire and accident victims to sudden illness. The hospital maintains a register to maintain the records of the patients who enter the emergency department. The receptionist at the helpdesk would like to filter the patients based on a criterion. Develop a java application for the same using Streams.

Requirements:

1. Read the patient records from the file.
2. Retrieve the patient details for the specified date interval.
3. Retrieve the patient details which are from a particular area (address).

Component Specification: Patient (POJO Class)

Type (Class)	Attributes	Methods
Patient	String patientId String patientName String contactNumber String dateOfVisit String patientAddress	Getters and Setters are given in the code skeleton.

Component Specification: PatientUtility

Type (Class)	Methods	Responsibilities
--------------	---------	------------------

PatientUtility	public List <Patient> fetchPatient(String filePath)	<p>Read the file using File I/O or Java Streams and return the validated list of patient records. It should filter the valid patient records based on the valid patient Id using the method isValidPatientId ().</p> <p>Note: Make sure that the user-defined exception is handled in this method itself.</p>
PatientUtility	public boolean isValidPatientId (String patientId)	<p>Validation Guidelines for Valid Patient ID:</p> <ul style="list-style-type: none"> • The length of the Patient Id should be exactly 6. • The Patient Id should start with “WM_” and the next letter should be an alphabet (A-Z) in upper case and the last two letters should be positive integers(0-9). Example. WM_A10. <p>Check whether the patient Id is valid or not. If invalid, this method should handle an InvalidPatientIdException with a message “<patientid> is an Invalid Patient Id”.</p>
PatientUtility	public List<Patient> retrievePatientRecords_ByDateOfVisit(Stream<Patient> patientStream, String fromDate, String toDate)	<p>From the provided stream of patient, separate the patient details which has the date of visit between fromDate and toDate (both inclusive) and return the resultant patient records as a list.</p>

PatientUtility	<pre>public Stream<Patient> retrievePatientRecords_ByAddress(Stream<Patient> patientStream, String address)</pre>	From the given stream of patient, filter the patient details based on the user input address, and return the separated Stream of patients.
----------------	---	--

Component Specification: InvalidPatientIdException (User defined Exception)

Type (Class)	Methods	Responsibilities
InvalidPatientIdException	<pre>public InvalidPatientIdException(String message)</pre>	This constructor should set the message to the superclass.

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

You are provided with a text file –PatientRegister.txt, which contains all the patient details like the patient Id, patient name, contact number, date of visit, and patient address. You can add any number of records in the text file to test your code.

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 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:

Invalid Patient Id are:

WM_52E is an Invalid Patient Id

WM_K7K is an Invalid Patient Id

WN_P23 is an Invalid Patient Id

LM_V20 is an Invalid Patient Id

WM_WQ0 is an Invalid Patient Id

Retrieve Patient Details

1. By Date of Visit

2. By Address

Enter your choice:

1

Enter the start date

02-03-2003

Enter the end date

02-12-2005

WM_X26 Beatrice 6789567687 18-07-2004 Texas

WM_H72 Elise 9809908765 02-12-2005 Washington

WM_N32 Finley 8729196472 12-08-2003 Pennsylvania

WM_Q91 Carny 8976509871 30-12-2003 Virginia

Sample Input/Output 2:

Invalid Patient Id are:

WM_52E is an Invalid Patient Id

WM_K7K is an Invalid Patient Id

WN_P23 is an Invalid Patient Id

LM_V20 is an Invalid Patient Id

WM_WQ0 is an Invalid Patient Id

Retrieve Patient Details

1. By Date of Visit

2. By Address

Enter your choice:

2

Enter the address

Carolina

WM_C23 Anya 7656548798 23-11-2002 Carolina

WM_T83 Harper 7391027349 08-07-1999 Carolina

WM_P21 Eleanor 8954321378 24-11-2007 Carolina

Sample Input/Output 3:

Invalid Patient Id are:

WM_52E is an Invalid Patient Id

WM_K7K is an Invalid Patient Id

WN_P23 is an Invalid Patient Id

LM_V20 is an Invalid Patient Id

WM_WQ0 is an Invalid Patient Id

Retrieve Patient Details

1. By Date of Visit

2. By Address

Enter your choice:

1

Enter the start date

03-02-2020

Enter the end date

02-02-2021

No patient records available during this interval

Sample Input/Output 4:

Invalid Patient Id are:

WM_52E is an Invalid Patient Id

WM_K7K is an Invalid Patient Id

WN_P23 is an Invalid Patient Id

LM_V20 is an Invalid Patient Id

WM_WQ0 is an Invalid Patient Id

Retrieve Patient Details

1. By Date of Visit

2. By Address

Enter your choice:

3

Invalid Option

Automatic evaluation[+]

HospitalManagement/PatientRegister.txt

```
1 WM_J82,Jacey,8734909012,07-08-2001,Colorado
2 WM_L01,Bella,9435678631,21-09-1992,Connecticut
3 WM_52E,Berey,8754321256,20-03-1999,Indiana
4 WM_B83,Cappi,6709543276,13-02-2006,Pennsylvania
5 WM_C23,Any,7656548798,23-11-2002,Carolina
6 WM_X26,Beatrice,6789567687,18-07-2004,Texas
7 WM_H72,Elise,9809908765,02-12-2005,Washington
8 WM_P10,Fanny,7835627189,02-05-2001,Virginia
9 WM_Q12,Felicity,6792637810,21-05-1997,Colorado
10 WM_K7K,Abigail,8934562718,02-05-2016,Indiana
11 WM_U82,Alice,8352617181,11-05-2012,Indiana
12 WN_P23,Amber,9876567898,12-09-1998,Pennsylvania
13 LM_V20,Gabriella,8302927382,02-05-2006,Connecticut
14 WM_Z05,Hadley,7823919273,02-06-2007,Connecticut
15 WM_T83,Harper,7391027349,08-07-1999,Carolina
16 WM_M03,Iris,9102638491,27-05-2001,Texas
17 WM_N32,Finley,8729196472,12-08-2003,Pennsylvania
18 WM_WQ0,Fiona,7201982219,09-09-2014,Washington
19 WM_Q91,Carny,8976509871,30-12-2003,Virginia
20 WM_P21,Eleanor,8954321378,24-11-2007,Carolina
```

HospitalManagement/src/InvalidPatientIdException.java

```
1
2 //public class InvalidPatientIdException{
3     //FILL THE CODE HERE
4     public class InvalidPatientIdException extends Exception{
5         public InvalidPatientIdException(String message){
6             super(message);
7         }
8     }
9
10
11
12
```

HospitalManagement/src/Main.java

```
1 public class Main {
2
3     public static void main(String[] args){
4
5         // CODE SKELETON - VALIDATION STARTS
6         // DO NOT CHANGE THIS CODE
7
8         new SkeletonValidator();
9         // CODE SKELETON - VALIDATION ENDS
10
11         // FILL THE CODE HERE
12
13     }
14
```



```
15     }
16
17
```

HospitalManagement/src/Patient.java

```
1 //DO NOT ADD/EDIT THE CODE
2 public class Patient {
3
4     private String patientId;
5     private String patientName;
6     private String contactNumber;
7     private String dateOfVisit;
8     private String patientAddress;
9
10    //Setters and Getters
11
12    public String getPatientId() {
13        return patientId;
14    }
15    public void setPatientId(String patientId) {
16        this.patientId = patientId;
17    }
18    public String getPatientName() {
19        return patientName;
20    }
21    public void setPatientName(String patientName) {
22        this.patientName = patientName;
23    }
24    public String getContactNumber() {
25        return contactNumber;
26    }
27    public void setContactNumber(String contactNumber) {
28        this.contactNumber = contactNumber;
29    }
30    public String getDateOfVisit() {
31        return dateOfVisit;
32    }
33    public void setDateOfVisit(String dateOfVisit) {
34        this.dateOfVisit = dateOfVisit;
35    }
36    public String getPatientAddress() {
37        return patientAddress;
38    }
39    public void setPatientAddress(String patientAddress) {
40        this.patientAddress = patientAddress;
41    }
42
43
44
45
46 }
47
```

HospitalManagement/src/PatientUtility.java

```
1 import java.util.List;
2 import java.util.stream.Stream;
3 import java.util.ArrayList;
4 import java.io.File;
5 import java.io.FileNotFoundException;
6 import java.util.Scanner;
7 import java.util.regex.*;
8 import java.util.stream.Collectors;
9 import java.text.ParseException;
10 import java.text.SimpleDateFormat;
11 import java.util.Date;
12
13
```

```

14 public class PatientUtility {
15
16     public List <Patient> fetchPatient(String filePath) {
17
18
19         //FILL THE CODE HERE
20         List <Patient> patients =new ArrayList<>();
21         try{
22             File register =new File(filePath);
23             Scanner reader=new Scanner(register);
24             while(reader.hasNextLine()){
25                 Patient p = new Patient();
26                 String[] infos=reader.nextLine().split(",");
27                 try{
28                     if(isValidPatientId(infos[0])){
29                         p.setPatientId(infos[0]);
30                         p.setPatientName(infos[1]);
31                         p.setContactNumber(infos[2]);
32                         p.setDateOfVisit(infos[3]);
33                         p.setPatientAddress(infos[4]);
34                         patients.add(p);
35                     }
36                 }
37                 catch(InvalidPatientIdException e1){
38                     System.out.println(e1.getMessage());
39                 }
40             }
41             reader.close();
42         }
43         catch(FileNotFoundException e){}
44         return patients;
45
46         //return null;
47     }
48
49     public boolean isValidPatientId (String patientId)throws InvalidPatientIdException
50     {
51
52         //FILL THE CODE HERE
53         Pattern p =Pattern.compile("WM_[A-Z][0-9]{2}$");
54         Matcher m=p.matcher(patientId);
55         boolean ne =m.matches();
56         if(!ne){
57             throw new InvalidPatientIdException(patientId+"is an Invalid Patient Id.");
58         }
59         //return inValid;
60         return ne;
61     }
62
63     public List<Patient> retrievePatientRecords_ByDateOfVisit(Stream<Patient> patientStream, String
64     fromDate, String toDate)
65     {
66         //FILL THE CODE HERE
67         SimpleDateFormat simpleDateFormat=new SimpleDateFormat("dd-MM-yyyy");
68         return patientStream
69             .filter((p)->{
70                 try{
71                     Date start=simpleDateFormat.parse(fromDate);
72                     Date end= simpleDateFormat.parse(toDate);
73                     Date current =simpleDateFormat.parse(p.getDateOfVisit());
74                     return start.compareTo(current)*current.compareTo(end)>=0;
75                 }
76                 catch(ParseException e){}
77             })
78         return false;
79     }

```

```

80         }).collect(Collectors.toList());
81         //         return null;
82     }
83
84
85
86     public Stream<Patient> retrievePatientRecords_ByAddress(Stream<Patient> patientStream, String
address)
87     {
88
89         //FILL THE CODE HERE
90         return patientStream.filter(p->address.equals(p.getPatientAddress()));
91         //return null;
92
93
94
95     }
96
97 }
98

```

HospitalManagement/src/SkeletonValidator.java

```

1  import java.lang.reflect.Method;
2  import java.util.List;
3  import java.util.logging.Level;
4  import java.util.logging.Logger;
5  import java.util.stream.Stream;
6
7  /**
8   * @author TJ
9   *
10  * This class is used to verify if the Code Skeleton is intact and not modified by participants thereby ensuring
smooth auto evaluation
11  *
12  */
13  public class SkeletonValidator {
14
15      public SkeletonValidator() {
16
17
18          validateClassName("Patient");
19          validateClassName("PatientUtility");
20          validateClassName("InvalidPatientIdException");
21          validateMethodSignature(
22
23              "fetchPatient:java.util.List,isValidPatientId:boolean,retrievePatientRecords_ByDateOfVisit:java.util.List
,retrievePatientRecords_ByAddress:java.util.stream.Stream",
24              "PatientUtility");
25      }
26
27      private static final Logger LOG = Logger.getLogger("SkeletonValidator");
28
29      protected final boolean validateClassName(String className) {
30
31          boolean iscorrect = false;
32          try {
33              Class.forName(className);
34              iscorrect = true;
35              LOG.info("Class Name " + className + " is correct");
36
37          } catch (ClassNotFoundException e) {
38              LOG.log(Level.SEVERE, "You have changed either the " + "class
name/package. Use the correct package "
39                  + "and class name as provided in the skeleton");
40
41          } catch (Exception e) {

```

```

42             LOG.log(Level.SEVERE,
43                 "There is an error in validating the " + "Class Name.
Please manually verify that the "
44                 + "Class name is same as
skeleton before uploading");
45         }
46         return incorrect;
47     }
48 }
49
50     protected final void validateMethodSignature(String methodWithExcpn, String className) {
51         Class cls = null;
52         try {
53
54             String[] actualMethods = methodWithExcpn.split(",");
55             boolean errorFlag = false;
56             String[] methodSignature;
57             String methodName = null;
58             String returnType = null;
59
60             for (String singleMethod : actualMethods) {
61                 boolean foundMethod = false;
62                 methodSignature = singleMethod.split(":");
63
64                 methodName = methodSignature[0];
65                 returnType = methodSignature[1];
66                 cls = Class.forName(className);
67                 Method[] methods = cls.getMethods();
68                 for (Method findMethod : methods) {
69                     if (methodName.equals(findMethod.getName())) {
70                         foundMethod = true;
71                         if
72                         (!findMethod.getReturnType().getName().equals(returnType))) {
73                             errorFlag = true;
74                             LOG.log(Level.SEVERE, " You
have changed the " + "return type in " + methodName
75                             + "
method. Please stick to the " + "skeleton provided");
76                         } else {
77                             LOG.info("Method signature of "
+ methodName + " is valid");
78                         }
79                     }
80                 }
81             }
82             if (!foundMethod) {
83                 errorFlag = true;
84                 LOG.log(Level.SEVERE, " Unable to find the given
public method " + methodName
85                 + ". Do not change the " + "given
public method name. " + "Verify it with the skeleton");
86             }
87         }
88     }
89     if (!errorFlag) {
90         LOG.info("Method signature is valid");
91     }
92 }
93     } catch (Exception e) {
94         LOG.log(Level.SEVERE,
95             " There is an error in validating the " + "method
structure. Please manually verify that the "
96             + "Method signature is same as
the skeleton before uploading");
97     }
98 }

```

99
100 }

Grade

Reviewed on Monday, 7 February 2022, 6:04 PM by Automatic grade

Grade 100 / 100

Assessment report

Assessment Completed Successfully

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

6. Technology Fest

Grade settings: Maximum grade: 100

Run: Yes **Evaluate:** Yes

Automatic grade: Yes

Institute of Technology is organizing an All-India Technology Fest for various engineering colleges across the country. The management would like to automate the registration so that it is easier and more systematic while conducting the fest. Create a java application for the same using Threads.

Component Specification: Participant (POJO Class)

Type (Class)	Attributes	Methods
Participant	String name String yearofstudy String department String collegeName String eventName double registrationFee	Getters, Setters, and a five-argument constructor in the given order - name, yearofstudy, department, collegeName, eventName are included in the code Skeleton.

Requirements:

- To calculate the registration fee of the participant based on the event name.
- To calculate the number of participants registered for a particular event.

Sl No	Event Name	Registration Fee
1	Robocar	1000
2	PaperTalk	500
3	Quiz	300
4	Games	100

***Note that Event name is case in- sensitive**

Component Specification: EventManagement (Thread Class)

Type (Class)	Attributes	Methods	Responsibilities
EventManagement	List<Participant> TechList String searchEvent int counter		Include getters and setter methods for all the attributes.
EventManagement		public void calculateRegistrationFee(List<Participant> list)	Calculate the registration fee of the participant based on the event name. If the event name doesn't exist, throw an InvalidEventException with an error message "Event Name is invalid".
EventManagement		public void run()	Calculate the number of participants registered for a particular event. Increment the counter attribute based on the search.

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

Component Specification: InvalidEventException

Type (Class)	Methods	Responsibilities
InvalidEventException	public InvalidEventException (String message)	To set the message string to the superclass.

Create a class called Main with the main method and perform the tasks are given below:

- Get the inputs as provided in the sample input.
- Call the calculateRegistrationFee () method to calculate the registration fee of the participant based on the event name.
- Print the list of Participant objects with the registration fee.
- Get the event type to search to find the number of the participants registered for that particular event.
- Handle the user-defined exception in the main method.
- Display the output as shown in the sample input/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 represent the output.
- Ensure to follow the 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 entries

3

Enter the Participant Name/Yearofstudy/Department/CollegeName/EventName

rinu/4/EEE/mnm/robocar

fina/3/EEE/psg/papertalk

rachel/4/civil/kcg/quiz

Print participant details

ParticipantName=rinu, Yearofstudy=4, Department=EEE, CollegeName=mnm,
EventName=robocar, RegistrationFee=1000.0

ParticipantName=fina, Yearofstudy=3, Department=EEE, CollegeName=psg,
EventName=papertalk, RegistrationFee=500.0

ParticipantName=rachel, Yearofstudy=4, Department=civil, CollegeName=kcg,
EventName=quiz, RegistrationFee=300.0

Enter the event to search

robocar

Number of participants for ROBOCAR event is 1

Sample Input/Output 2:

Enter the number of entries

3

Enter the Participant Name/Yearofstudy/Department/CollegeName/EventName

rinu/4/EEE/mnm/robocar

fina/3/EEE/psg/papertalk

rachel/4/civil/kcg/quiz

Print participant details

ParticipantName=rinu, Yearofstudy=4, Department=EEE, CollegeName=mnmm,
EventName=robocar, RegistrationFee=1000.0

ParticipantName=fina, Yearofstudy=3, Department=EEE, CollegeName=psg,
EventName=papertalk, RegistrationFee=500.0

ParticipantName=rachel, Yearofstudy=4, Department=civil, CollegeName=kcg,
EventName=quiz, RegistrationFee=300.0

Enter the event to search

games

No participant found

Sample Input/Output 3:

Enter the number of entries

2

Enter the Participant Name/Yearofstudy/Department/CollegeName/EventName

vishal/4/mech/vjc/flyingrobo

vivek/3/mech/hdl/games

Event Name is invalid

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

TechnologyFest/src/EventManager.java

```
1 import java.util.List;
2
3 public class EventManagement implements Runnable {
4     private List<Participant> TechList;
5     private String searchEvent;
6     private int counter=0;
7     public List<Participant>getTechList()
8     {
9         return TechList;
10    }
11
12    public void setTechList(List<Participant>techList)
13    {
14        TechList=techList;
15    }
16    public String getSearchEvent()
17    {
18        return searchEvent;
```



```

19     }
20     public void setSearchEvent(String searchEvent)
21     {
22         this.searchEvent=searchEvent;
23     }
24     public int getCounter()
25     {
26         return counter;
27     }
28     public void setCounter(int counter)
29     {
30         this.counter=counter;
31     }
32     //FILL THE CODE HERE
33
34     public void calculateRegistrationFee(List<Participant> list) throws InvalidEventException
35     {
36         for(Participant p:list)
37         {
38             if(p.getEventName().equalsIgnoreCase("robocar"))
39             {
40                 p.setRegistrationFee(1000);
41             }
42             else if(p.getEventName().equalsIgnoreCase("papertalk")){
43                 p.setRegistrationFee(500);
44             }
45             }
46         }
47         else if(p.getEventName().equalsIgnoreCase("quiz")){
48             p.setRegistrationFee(300);
49         }
50         else if(p.getEventName().equalsIgnoreCase("games")){
51             p.setRegistrationFee(100);
52         }
53         else{
54             throw new InvalidEventException("Event Name is Invalid");
55         }
56     }
57     }
58     //FILL THE CODE HERE
59     setTechList(list);
60 }
61
62 public void run()
63 {
64     String str="robocarpapertalkquizgames";
65     if(str.contains(this.getSearchEvent())){
66         for(Participant P:this.getTechList()){
67             if(this.getSearchEvent().equals(P.getEventName())){
68                 counter++;
69             }
70         }
71     }
72     setCounter(counter);
73
74     //FILL THE CODE HERE
75
76 }
77 }
78

```

TechnologyFest/src/InvalidEventException.java

```

1 public class InvalidEventException extends Exception{
2     //FILL THE CODE HERE
3 public InvalidEventException(String str){
4     super(str);
5

```

```

6 }
7
8 }
9

```

TechnologyFest/src/Main.java

```

1
2 import java.util.Scanner;
3 import java.util.*;
4 public class Main {
5     public static void main(String [] args)
6     {
7         // CODE SKELETON - VALIDATION STARTS
8         // DO NOT CHANGE THIS CODE
9
10        new SkeletonValidator();
11
12        // CODE SKELETON - VALIDATION ENDS
13
14        Scanner sc=new Scanner(System.in);
15        System.out.println("Enter the number of entries");
16        int n=sc.nextInt();
17        System.out.println("Enter the Participant
Name/Yearofstudy/Department/CollegeName/EventName");
18        List<Participant> list=new ArrayList<Participant>();
19        String strlist[]=new String[n];
20        for(int i=0;i<n;i++)
21        {
22            strlist[i]=sc.next();
23            String a[]=strlist[i].split("/");
24            Participant pt=new Participant(a[0],a[1],a[2],a[3],a[4]);
25            list.add(pt);
26        }
27        EventManagement em=new EventManagement();
28        try {
29            em.calculateRegistrationFee(list);
30        }
31        catch(InvalidEventException e)
32        {
33            e.printStackTrace();
34        }
35        System.out.println("Print participant details");
36        for(Participant p:list)
37        {
38            System.out.println(p);
39        }
40        System.out.println("Enter the event to search");
41        String srch=sc.nextLine();
42        em.setSearchEvent(srch);
43        em.run();
44        int count=em.getCounter();
45        if(count<=0){
46            System.out.println("No participant found");
47        }
48        else{
49            System.out.println("Number of participants for"+srch+"event is "+count);
50        }
51    }
52 }
53
54
55
56
57

```

TechnologyFest/src/Participant.java

```

1 public class Participant {

```

```

2     private String name;
3     private String yearofstudy;
4     private String department;
5     private String collegeName;
6     private String eventName;
7     private double registrationFee;
8
9     //5 argument Constructor
10    public Participant(String name, String yearofstudy, String department, String collegeName, String
eventName) {
11        super();
12        this.name = name;
13        this.yearofstudy = yearofstudy;
14        this.department = department;
15        this.collegeName = collegeName;
16        this.eventName = eventName;
17    }
18
19    public String getName() {
20        return name;
21    }
22    public void setName(String name) {
23        this.name = name;
24    }
25    public String getYearofstudy() {
26        return yearofstudy;
27    }
28    public void setYearofstudy(String yearofstudy) {
29        this.yearofstudy = yearofstudy;
30    }
31    public String getDepartment() {
32        return department;
33    }
34    public void setDepartment(String department) {
35        this.department = department;
36    }
37    public String getCollegeName() {
38        return collegeName;
39    }
40    public void setCollegeName(String collegeName) {
41        this.collegeName = collegeName;
42    }
43    public String getEventName() {
44        return eventName;
45    }
46    public void setEventName(String eventName) {
47        this.eventName = eventName;
48    }
49    public double getRegistrationFee() {
50        return registrationFee;
51    }
52    public void setRegistrationFee(double registrationFee) {
53        this.registrationFee = registrationFee;
54    }
55
56    @Override
57    public String toString() {
58        return "Participant [name=" + name + ", yearofstudy=" + yearofstudy + ", department=" +
department
59        + ", collegeName=" + collegeName + ", eventName=" +
eventName + ", registrationFee=" + registrationFee
60        + "];"
61    }
62
63
64
65

```

```

66 }
67
TechnologyFest/src/SkeletonValidator.java
1
2 import java.lang.reflect.Method;
3 import java.util.List;
4 import java.util.logging.Level;
5 import java.util.logging.Logger;
6 import java.util.stream.Stream;
7
8 /**
9  * @author TJ
10  *
11  * This class is used to verify if the Code Skeleton is intact and not modified by participants thereby ensuring
smooth auto evaluation
12  *
13  */
14 public class SkeletonValidator {
15
16     public SkeletonValidator() {
17
18         //classes
19         validateClassName("Main");
20         validateClassName("EventManagement");
21         validateClassName("Participant");
22         validateClassName("InvalidEventException");
23         //functional methods
24         validateMethodSignature(
25             "calculateRegistrationFee:void", "EventManagement");
26         validateMethodSignature(
27             "run:void", "EventManagement");
28
29         //setters and getters of HallHandler
30         validateMethodSignature(
31             "getTechList:List", "EventManagement");
32         validateMethodSignature(
33             "setTechList:void", "EventManagement");
34
35         validateMethodSignature(
36             "getCounter:int", "EventManagement");
37         validateMethodSignature(
38             "setCounter:void", "EventManagement");
39
40         validateMethodSignature(
41             "getSearchEvent:String", "EventManagement");
42         validateMethodSignature(
43             "setSearchEvent:void", "EventManagement");
44
45         //setters and getters of Hall
46         validateMethodSignature(
47             "getName:String", "Participant");
48         validateMethodSignature(
49             "setName:void", "Participant");
50
51         validateMethodSignature(
52             "getYearofstudy:String", "Participant");
53         validateMethodSignature(
54             "setYearofstudy:void", "Participant");
55
56         validateMethodSignature(
57             "getDepartment:String", "Participant");
58         validateMethodSignature(
59             "setDepartment:void", "Participant");
60
61         validateMethodSignature(
62             "getCollegeName:String", "Participant");

```

```

63         validateMethodSignature(
64             "setCollegeName:void", "Participant");
65
66         validateMethodSignature(
67             "getEventName:String", "Participant");
68         validateMethodSignature(
69             "setEventName:void", "Participant");
70
71         validateMethodSignature(
72             "getRegistrationFee:double", "Participant");
73         validateMethodSignature(
74             "setRegistrationFee:void", "Participant");
75
76     }
77
78     private static final Logger LOG = Logger.getLogger("SkeletonValidator");
79
80     protected final boolean validateClassName(String className) {
81
82         boolean iscorrect = false;
83         try {
84             Class.forName(className);
85             iscorrect = true;
86             LOG.info("Class Name " + className + " is correct");
87
88         } catch (ClassNotFoundException e) {
89             LOG.log(Level.SEVERE, "You have changed either the " + "class
name/package. Use the correct package "
90                 + "and class name as provided in the skeleton");
91
92         } catch (Exception e) {
93             LOG.log(Level.SEVERE,
94                 "There is an error in validating the " + "Class Name.
Please manually verify that the "
95                 + "Class name is same as
skeleton before uploading");
96         }
97         return iscorrect;
98
99     }
100
101     protected final void validateMethodSignature(String methodWithExcpn, String className) {
102         Class cls = null;
103         try {
104
105             String[] actualMethods = methodWithExcpn.split(",");
106             boolean errorFlag = false;
107             String[] methodSignature;
108             String methodName = null;
109             String returnType = null;
110
111             for (String singleMethod : actualMethods) {
112                 boolean foundMethod = false;
113                 methodSignature = singleMethod.split(":");
114
115                 methodName = methodSignature[0];
116                 returnType = methodSignature[1];
117                 cls = Class.forName(className);
118                 Method[] methods = cls.getMethods();
119                 for (Method findMethod : methods) {
120                     if (methodName.equals(findMethod.getName())) {
121                         foundMethod = true;
122                         if
(!findMethod.getReturnType().getName().contains(returnType)) {
123                             errorFlag = true;
124                             LOG.log(Level.SEVERE, " You
have changed the " + "return type in " + methodName

```

```

125                                     + ""
method. Please stick to the " + "skeleton provided");
126
127                                     } else {
128                                     LOG.info("Method signature of "
+ methodName + " is valid");
129                                     }
130
131                                     }
132                                     }
133                                     if (!foundMethod) {
134                                     errorFlag = true;
135                                     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");
137                                     }
138
139                                     }
140                                     if (!errorFlag) {
141                                     LOG.info("Method signature is valid");
142                                     }
143
144                                     } catch (Exception e) {
145                                     LOG.log(Level.SEVERE,
146                                     " There is an error in validating the " + "method
structure. Please manually verify that the "
+ "Method signature is same as
the skeleton before uploading");
148                                     }
149     }
150
151 }

```

Grade

Reviewed on Monday, 7 February 2022, 6:34 PM by Automatic grade

Grade 74 / 100

Assessment report

```

Fail 1 -- test4CheckTheOutput::
$Expected output:"[Print participant details
ParticipantName=Weni
Yearofstudy=3
Department=civil
CollegeName=vjc
EventName=robocar
RegistrationFee=1000.0
ParticipantName=gina
Yearofstudy=2
Department=mech
CollegeName=vjc
EventName=quiz
RegistrationFee=300.0
ParticipantName=jos
Yearofstudy=4
Department=ece
CollegeName=vjec
EventName=games
RegistrationFee=100.0
ParticipantName=fida
Yearofstudy=1
Department=eee

```

```

CollegeName=vjec
EventName=papertalk
RegistrationFee=500.0
Enter the event to search
Number of participants for PAPERTALK event is 1]" Actual output:"[Enter the number of
entries
Enter the Participant Name/Yearofstudy/Department/CollegeName/EventName
Print participant details
Participant [name=Weni
yearofstudy=3
department=civil
collegeName=vjc
eventName=robocar
registrationFee=1000.0]
Participant [name=gina
yearofstudy=2
department=mech
collegeName=vjc
eventName=quiz
registrationFee=300.0]
Participant [name=jos
yearofstudy=4
department=ece
collegeName=vjec
eventName=games
registrationFee=100.0]
Participant [name=fida
yearofstudy=1
department=eee
collegeName=vjec
eventName=papertalk
registrationFee=500.0]
Enter the event to search
No participant found]"$
Check your code with the input :Weni/3/civil/vjc/robocar
gina/2/mech/vjc/quiz
jos/4/ece/vjec/games
fida/1/eee/vjec/papertalk

```

```

Fail 2 -- test6CheckTheOutputfor_NCount::
$Expected output:"[Print participant details
ParticipantName=philip
Yearofstudy=4
Department=eee
CollegeName=mvc
EventName=robocar
RegistrationFee=1000.0
ParticipantName=susan
Yearofstudy=4
Department=eee
CollegeName=mvc
EventName=robocar
RegistrationFee=1000.0
ParticipantName=vivek
Yearofstudy=3
Department=civil
CollegeName=mvc
EventName=quiz
RegistrationFee=300.0
ParticipantName=vishal
Yearofstudy=3
Department=civil
CollegeName=mvc

```

```

EventName=papertalk
RegistrationFee=500.0
Enter the event to search
Number of participants for ROBOCAR event is 2]" Actual output:"[Enter the number of
entries
Enter the Participant Name/Yearofstudy/Department/CollegeName/EventName
Print participant details
Participant [name=philip
yearofstudy=4
department=eee
collegeName=mvc
eventName=robocar
registrationFee=1000.0]
Participant [name=susan
yearofstudy=4
department=eee
collegeName=mvc
eventName=robocar
registrationFee=1000.0]
Participant [name=vivek
yearofstudy=3
department=civil
collegeName=mvc
eventName=quiz
registrationFee=300.0]
Participant [name=vishal
yearofstudy=3
department=civil
collegeName=mvc
eventName=papertalk
registrationFee=500.0]
Enter the event to search
No participant found]"$
Check your code with the input :philip/4/eee/mvc/robocar
susan/4/eee/mvc/robocar
vivek/3/civil/mvc/quiz
vishal/3/civil/mvc/papertalk
robocar

```

Obtained Pass Percentage. Still few testcases failed . Kindly revisit the Solution

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

=====

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 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;
}
}

```

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 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;
```

```
}
```

```
}
```

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.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:

```
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);
}
}

```

}
}

```

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));
    }
}
```

```

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("pilgrimage")){
                        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");
}
}
}
}

```

```

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;
    }
}

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));
    }
}

```

Software License Details

index.html

```
<!--Do not make any change in this code template -->

<html>

<head>

<script src="script.js" type="text/javascript"> </script>

</head>

<body>

  <h2>Software License Details</h2>
    <table>
      <tr>
        <td> Software Name</td>
        <td><input type="text" id="softwareName" placeholder="Enter the software
name" required></td>
      </tr>
      <tr>
        <td> Serial Key </td>
        <td> <input type="text" id="serialKey" placeholder="Enter 12 digit alphanumeric
serial key" required></td>
      </tr>

      <tr>
        <td> </td>
        <td> <button id="validate" onclick=validate()>Validate</button> </td>
      </tr>
    </table>
    <div id="result"></div>
  </body>
</html>
```

script.js

```
// Fill the code wherever necessary
```

```
function validate()
{
    var softwareName=document.getElementById("softwareName").value;
    var serialKey = document.getElementById("serialKey").value;
    //var serialKey= //Fill your code here to get the value of element by using id "serialKey" and
store it in a variable "serialKey"
    //HINT: use the above "softwareName" as a sample to get "serialKey"

    if(softwareName && serialKey)
    {
        if(validateSerialKey(serialKey))
```

```

        document.getElementById("result").innerHTML = "The serial key "+serialKey+" is validated
successfully for the software "+softwareName;
    else
        document.getElementById("result").innerHTML = "Please, provide a valid serial key with 12
alphanumeric characters";
    }
    else
        document.getElementById("result").innerHTML = "Software name (or) serial key missing";
}

function validateSerialKey(serialKey)
{
    var pattern=/^[0-9a-zA-Z]{12}$/;
    var isSerialKey = serialKey.match(pattern);
    return Boolean(isSerialKey);
    // Fill your code here
    // find if the serialKey is valid by checking if it matches the given pattern
    // return true or false
}

```

Find Highest Enrollment of Policies

index.html

```

<!--Do not make any change in this code template -->

<html>

<head>

<script src="script.js" type="text/javascript"> </script>

</head>

<body>

    <h2>Find Highest Enrollment of Policies</h2>
    <table>
        <tr>
            <td>Policy Number</td>
            <td><input type="text" id="policyNumber" placeholder="Enter the 7 digit policy
number" required></td>
        </tr>
        <tr>
            <td>Enrolled Amount </td>
            <td><input type="number" id="amount" placeholder="Enter the amount"
required></td>
        </tr>

    <tr>

```

```

                <td> </td>
            <td> <button id="submit" onclick=validate()>Submit</button> </td>
        </tr>
    </table>
    <div id="result"></div>
</body>
</html>

```

script.js

// Fill the code wherever necessary

```

function validate()
{
    var policyNumber=document.getElementById("policyNumber").value;
    amount=document.getElementById("amount").value;
    //var amount= //Fill your code here to get the value of element by using id "amount" and
store it in a variable "amount"
    //HINT: use the above "policyNumber" as a sample to get "amount"

    if(policyNumber && amount)
    {
        if(validatePolicyNumber(policyNumber))
            document.getElementById("result").innerHTML = "The policy number "+policyNumber+" is
enrolled successfully with Rs "+amount;
        else
            document.getElementById("result").innerHTML = "Please, provide a valid 7 digit policy
number";
    }
    else
        document.getElementById("result").innerHTML = "Policy Number (or) Amount is missing";
}

function validatePolicyNumber(policyNumber)
{
    var pattern=/^[0-9]{7}$/;
    if(!(policyNumber.match(pattern)))
        return false
    else
        return true;

    // Fill your code here
    // find if the policyNumber is valid by checking if it matches the given pattern
    // return true or false

}

```

Email Validation

index.html

```
<!--Do not make any change in this code template -->

<html>

<head>

<script src="script.js" type="text/javascript"> </script>

</head>

<body>

    <h2>Registration form</h2>
    <table>
        <tr>
            <td> Trainee Name</td>
            <td><input type="text" id="traineeName" placeholder="Enter the trainee name"
required></td>
        </tr>
        <tr>
            <td> Email ID </td>
            <td> <input type="text" id="emailId" placeholder="Enter the email id"
required></td>
        </tr>

        <tr>
            <td> </td>
            <td> <button id="register" onclick=validate()>Register</button> </td>
        </tr>
    </table>
    <div id="result"></div>
</body>
</html>
```

script.js

```
// Fill the code wherever necessary

function validate()
{
    var traineeName=document.getElementById("traineeName").value;
    //var emailId= //Fill your code here to get the value of element by using id "emailId" and
store it in a variable "emailId"
    //HINT: use the above "traineeName" as a sample to get "emailId"
    var emailId = document.getElementById("emailId").value;
    if(traineeName && emailId)
    {
        if(validateEmailId(emailId))
            document.getElementById("result").innerHTML = "The email id : "+emailId+" is validated
successfully for the trainee "+traineeName;
        else
```

```

        document.getElementById("result").innerHTML = "Please, provide a valid email id";
    }
    else
        document.getElementById("result").innerHTML = "Trainee name (or) email id missing";
}

function validateEmailId(emailId)
{

    // Fill your code here to check whether the 'email' has '@' symbol and '.' symbol
    // HINT : emailId.includes("@") will return true, if the emailId has '@' symbol.
    // find whether email has both '@' and '.'
    // Return true or false
    if(emailId.includes('@') && emailId.includes('.')){
        return true;
    }
    return false;
}

```

Number Of Days

index.html

```

<!--Do not make any change in this code template -->

<html>

<head>

<script src="script.js" type="text/javascript"> </script>

</head>

<body>

    <h2>Recharge Pack Validity</h2>
    <table>
        <tr>
            <td> Recharge Pack Name</td>
            <td><input type="text" id="rechargePackName" placeholder="Enter the recharge
pack name" required></td>
        </tr>
        <tr>
            <td> Validity (in days) </td>
            <td> <input type="number" id="validity" min="1" placeholder="Enter the validity
in days" required></td>

```

```

        </tr>

    <tr>
        <td> </td>
        <td> <button id="validate" onclick=validate()>Submit</button> </td>
    </tr>
</table>
<div id="result"></div>
</body>
</html>
script.js
// Fill the code wherever necessary

function validate()
{
    var rechargePackName=document.getElementById("rechargePackName").value;
    var validity=document.getElementById("validity").value;//Fill your code here to get the value
of element by using id "validity" and store it in a variable "validity"
    //HINT: use the above "rechargePackName" as a sample to get "validity"

    if(rechargePackName && validity)
    {
        if(validateRechargePackName(rechargePackName))
            document.getElementById("result").innerHTML = "The recharge pack name
"+rechargePackName+" with a validity of "+validity+" days is validated successfully";
        else
            document.getElementById("result").innerHTML = "Please, provide a valid recharge pack
name";
    }
    else
        document.getElementById("result").innerHTML = "Recharge pack name (or) validity in
days missing";
}

function validateRechargePackName(rechargePackName)
{
    var pattern=/^[A-Z]{2}[0-9]{3}$/;
    // Fill your code here
    // find if the rechargePackName is valid by checking if it matches the given pattern
    // return true or false
    if(rechargePackName.match(pattern))
    {
        return true;
    }
    else
    {
        return false;
    }
}

```

```
}  
  
}
```

Frequency Calculation

index.html

```
<!--Do not make any change in this code template -->  
  
<html>  
  
<head>  
  
<script src="script.js" type="text/javascript"> </script>  
  
</head>  
  
<body>  
  <h2>Frequency Calculator</h2>  
  <table>  
    <tr>  
      <td> Frequency Band</td>  
      <td><input type="text" id="band" placeholder="Enter the frequency band"  
required></td>  
    </tr>  
    <tr>  
      <td> Wavelength in mm </td>  
      <td> <input type="number" id="wavelength" placeholder="0.1-1000" min="0.1"  
max="1000" step="0.1" required></td>  
    </tr>  
  
    <tr>  
      <td> </td>  
      <td> <button id="submit" onclick=validate()>Submit</button> </td>  
    </tr>  
    <tr>  
      <td colspan="2">  
        * Acceptable frequency bands are H, M, L, U, S, C, X, K.  
      </td>  
    </tr>  
  </table>  
  <div id="result"></div>  
  
</body>  
</html>
```

script.js


```
// Fill the code wherever necessary

function validate()
{
    var band=document.getElementById("band").value;
    var wavelength=document.getElementById("wavelength").value //Fill your code here to get
the value of element by using id "wavelength" and store it in a variable "wavelength"
    //HINT: use the above "band" as a sample to get "wavelength"

    if(band && wavelength)
    {
        if(validateFrequencyBand(band))
            document.getElementById("result").innerHTML = "The frequency band "+band+" with
wavelength "+wavelength+" is validated successfully";
        else
            document.getElementById("result").innerHTML = "Please, provide a valid frequency band";
    }
    else
        document.getElementById("result").innerHTML = "Frequency band (or) wavelength
missing";
}

function validateFrequencyBand(band)
{
    var pattern=/^[H|M|L|U|S|C|X|K]$/;
    // Fill your code here
    // find if the band is valid by checking if it matches the given pattern
    // return true or false
    if(band.match(pattern))
    {
        return true;
    }
    else

    {
        return false;
    }
}
}
```

AC Maintenance Service-V1

AcMaintenanceService.html

```

<!DOCTYPE html>

<html>

<head>

<script src="script.js" lang="text/javascript"></script>
<title>AC Maintenance Service</title>
<style type="text/css">
  body {
    /* Fill attributes and values */
    background-color: #0CA2B9;
    width: 80%;
    margin-left: 10%;
  }
  h1 {
    /* Fill attributes and values */
    color: #FFFFFF;
    font-family: Calibri;
    font-style: italic;
    background-color: #900043;
    text-align: center;
  }
  #result {
    /* Fill attributes and values */
    font-weight: bold;
    font-family: Arial;
    font-size: 18px;
    color: #782E07;
  }

  #submit, #reset {
    /* Fill attributes and values */
    font-weight: bold;
    font-family: Candara;
    background-color: #556B2F;
    width: 10em;
    height: 35px;
    border-radius: 10px;
  }

  input {
    width: 13.6em;
  }
  #appointment {
    font-family: sans-serif;
    width: 80%;
    border-collapse: collapse;
    text-align: left;
  }

```

```

#acType, textarea{
    width:13.6em;
}
select {
    width:14em;
}
td{
    padding:3px;
}
#male, #female, #yearlyMaintenance {
    width:10pt;
}
.checkboxes label {
    display: inline-block;
    padding-right: 10px;
    white-space: nowrap;
}
.checkboxes input {
    vertical-align: middle;
}
.checkboxes label span {
    vertical-align: middle;
}
</style>
</head>

```

```
<body>
```

```
<h1>AC Maintenance Service</h1>
```

```
<form onsubmit="return bookAppointment()" >
```

```

<table id="appointment">
  <tr>
    <td> <label for = 'customerName'>Customer Name</label></td>
    <td><input type='text' id = 'customerName' placeholder="Enter your name" required> </td>
  </tr>

  <tr>
    <td> <label for = 'mobileNumber'>Mobile Number</label> </td>
    <td> <input type = 'tel' id = 'mobileNumber' name = 'Mobile Number' placeholder="Enter your
mobile number" pattern="^[7-9][0-9]{9}$" maxLength="10" minLength = '10' required> </td>
  </tr>

  <tr>
    <td> <label for = 'address'>Address</label></td>

```

```

        <td> <textarea id= 'address' name = 'address' placeholder="Enter your address" rows = '5' cols
        ='25' required></textarea> </td>
    </tr>

    <tr>
        <td> <label for = 'acType'>AC Type</label> </td>
        <td>

            <select id="acType">
                <option id="Split" value ="Split">Split</option>
                <option id="Window" value ="Window">Window</option>
                <option id = "Centralized" value = "Centralized">Centralized</option>
                <option id='Portable' value = 'Portable'>Portable</option>
            </select>

        </td>
    </tr>

    <tr>
        <td> <label for ='serviceType'>Service Type</label> </td>
        <td>
            <input type="checkbox" name="serviceType" id="Cleaning" value="Cleaning" ><label for =
            'Cleaning'> Cleaning</label>
            <input type="checkbox" name="serviceType" id="Repair" value="Repair" ><label for =
            'Repair'> Repair</label>
            <input type="checkbox" name="serviceType" id="Gas Refill" value="Gas Refill" ><label for =
            'Gas Refill'> Gas Refill</label>
            <input type="checkbox" name="serviceType" id="Relocation" value="Relocation" ><label for =
            'Relocation'> Relocation</label>
            <input type="checkbox" name="serviceType" id="Filter" value="Filter" ><label for = 'Filter'>
            Filter</label>

        </td>
    </tr>

    <tr>
        <td> <label for = 'dateForAppointment'>Date for Appointment</label> </td>
        <td> <input type = 'date' id = 'dateForAppointment' required> </td>
    </tr>

    <tr>
        <td> <label for ='yearlyMaintenance'>Yearly Maintenance</label> </td>
        <td> <input type = 'checkbox' id = 'yearlyMaintenance' name = 'yearlyMaintenance'> <label
        for = 'yearlyMaintenance'>Select if required</label></td>
    </tr>

    <tr>
        <td> <!-- empty cell --></td>

```

```

        <td>
            <input type = 'submit' value = 'Submit' id = 'submit'>
            <input type ='reset' value ='Clear' id = 'reset' >
        </td>
    </tr>

    <tr>
        <td colspan="2">
            <div id="result"></div>
        </td>
    </tr>

</table>
</form>

</body>
</html>
script.js
function getTotalService() {
    var totalServices = document.getElementsByName("serviceType");
    var count = 0;
    for(var i=0; i<totalServices.length; i++) {
        if(totalServices[i].checked) {
            count++
        }
    }
    return count;
}

function getServiceCost() {
    var totalServices = document.getElementsByName("serviceType");
    var totalCost = 0;
    for(var i=0; i<totalServices.length; i++) {
        if(totalServices[i].checked) {
            switch(totalServices[i].value) {
                case "Cleaning":
                    totalCost += 500;
                    break;
                case "Repair":
                    totalCost += 2500;
                    break;
                case "Gas Refill":
                    totalCost += 750;
                    break;
                case "Relocation":

```

```

        totalCost += 1500;
        break;
    case "Filter":
        totalCost += 250;
        break;
    default:
        break;
    }
}
}
return totalCost;
}

```

```

function calculateDiscount(serviceCost) {
    serviceCost = serviceCost*0.85;
    return serviceCost;
}

```

```

function getYearlyMaintenanceCost() {
    var yearlyMaintenance = document.getElementsByName("yearlyMaintenance");
    if(yearlyMaintenance[0].checked)
        return 1500;
    else
        return 0;
}

```

```

function bookAppointment() {
    var totalNumberOfServices = getTotalService();
    var serviceCost = 0;
    if(totalNumberOfServices > 2) {
        serviceCost = calculateDiscount(getServiceCost());
    } else {
        serviceCost = getServiceCost();
    }

    var yearlyMaintenanceCost = getYearlyMaintenanceCost();
    var totalCost = serviceCost + yearlyMaintenanceCost;

    var acType = document.getElementById("acType").value;

    if(yearlyMaintenanceCost) {
        document.getElementById("result").innerHTML = "Your booking for " + acType +

```

```

        " AC service is successful!<br>The estimated service cost with maintenance is Rs." +
Math.round(totalCost);
    } else {
        document.getElementById("result").innerHTML = "Your booking for " + acType +
        " AC service is successful!<br>The estimated service cost is Rs." + Math.round(totalCost);
    }
}

```

We-Host Server Resellers - Purchase Entry-V1

WEHOST.html

```

<!DOCTYPE html>

<html>

<head>

<script src="script.js" lang="text/javascript"></script>

<title>We-Host Server Resellers - Purchase Entry</title>
<style>

        input[type="text"] {
            width: 97%;}

        input[type="number"] {
            width: 97%;}

        input[type="tel"] {
            width: 97%;}

    body{
        background-image:url('WEHOST.jpg');
        background-size: 100%;
        font-weight: bold;
    }

        div{
            font-size: 20px;
            text-align: center;
            color:#FFFFFFF;
            margin-left: auto;
            margin-right: auto;
        }

        h3{
            width: 50%;
            color: #FFFFFFF;

```

```

        background-color: #000080;
        margin-left: 25%;
        margin-right: auto;
        text-align: center;
        font-family: Verdana;
        padding: 5px;
        border-radius: 6px;
    }

    table, td, tr{
        border : solid 1px;
        width: 50%;
        margin-left: auto;
        margin-right: auto;
        border-spacing: 1px;
        border-radius: 6px;
        color: #000080;
        background-color: #FFFFFF;
        padding: 1px;
    }

    ::-webkit-input-placeholder {
        color: #808080; }

    #submit{
        width: 50%;
        color: #FFFFFF;
        background-color: #000080;
        margin-left: 25%;
        margin-right: auto;
        padding: 5px;
        font-family: Verdana;
        font-weight: bold;
        border-radius: 6px;
    }

</style>
</head>
<body>

<h3>We-Host Server Resellers - Purchase Entry</h3>
<!--<form onsubmit="return calculatePurchaseCost()" >-->

<table>
    <tr>

```



```

        <td>Purchase Date</td>
        <td><input type="text" id="pdate" onfocus="today()" required/></td>
    </tr>
    <tr>
        <td>Customer Name</td>
        <td><input type="text" id="cname" placeholder="Enter the customer name" pattern="[a-zA-Z\s]+" required></td>
    </tr>
    <tr>
        <td>Address</td>
        <td><textarea placeholder="Enter the address" rows="4" cols="50" id="address"
required></textarea></td>
    </tr>
    <tr>
        <td>Phone Number</td>
        <td><input type="tel" id="phno" placeholder="Phone number" pattern="[7|8|9]+[0-9]{9}"
required></td>
    </tr>
    <tr>
        <td>Server Type</td>
        <td><select id="stype" required>
            <option value="Select Server Type..">Select Server Type.</option>
            <option id="Dedicated Server" value="Dedicated Server">Dedicated Server</option>
            <option id="VPS" value="VPS">VPS</option>
            <option id="Storage Server" value="Storage Server">Storage
Server</option>
            <option id="Database Server" value="Database Server">Database Server</option>
        </select>
        </td>
    </tr>
    <tr>
        <td>CPU(Core)</td>
        <td><select id="core" required>
            <option value="Select no of cores..">Select no of cores.</option>
            <option id="2 cores" value="2 cores">2 cores</option>
            <option id="4 cores" value="4 cores">4 cores</option>
            <option id="6 cores" value="6 cores">6 cores</option>
            <option id="8 cores" value="8 cores">8 cores</option>
        </select>
        </td>
    </tr>
    <tr>
        <td>Configuration</td>
        <td><select id="configuration" required>
            <option value="Select configuration..">Select configuration.</option>
            <option id="4 GB RAM , 300 GB SSD-boosted Disk Storage" value="4 GB RAM , 300 GB SSD-
boosted Disk Storage">4 GB RAM , 300 GB SSD-boosted Disk Storage</option>
            <option id="8 GB RAM , 700 GB SSD-boosted Disk Storage" value="8 GB RAM , 700 GB SSD-
boosted Disk Storage">8 GB RAM , 700 GB SSD-boosted Disk Storage</option>
            <option id="12 GB RAM , 1 TB SSD-boosted Disk Storage" value="12 GB RAM , 1
TB SSD-boosted Disk Storage">12 GB RAM , 1TB SSD-boosted Disk Storage</option>

```

```

        </select>
    </td>
</tr>

<tr>
    <td>Payment Type</td>
    <td><select id="ptype" required>
        <option id="Card" value="Card">Debit card / Credit card</option>
        <option id="Cash" value="Cash">Cash</option>
    </select>
    </td>
</tr>
</table>

<br/><br/>
<input type="submit" value="CONFIRM PURCHASE" id="submit" onclick="calculatePurchaseCost()">
<br/><br/>

<div id="result"> </div>

<br/><br/>

<!--</form>-->
</body>
</html>

```

```

script.js
function getCoreCost(core)
{
    if(core.startsWith("2"))
    {
        return 20000;
    }
    if(core.startsWith("4"))
    {
        return 25000;
    }
    if(core.startsWith("6"))
    {

```

```

        return 30000;
    }
    if(core.startsWith("8"))
    {
        return 40000;
    }
}

```

```
function getConfigurationCost(config)
```

```

{
    if(config.startsWith("4"))
    {
        return 5000;
    }
    if(config.startsWith("8"))
    {
        return 10000;
    }
    if(config.startsWith("1"))
    {
        return 15000;
    }
}

```

```

}

```

```
function calculateTax(totalcost,ptype)
```

```

{
    let tax,ex=0;
    totalcost=parseInt(totalcost);
    tax=totalcost*12/100;
    if(ptype.startsWith("Card"))
    {
        ex=(totalcost+tax)*2/100;
    }
    return Math.round(totalcost+tax+ex);
}

```

```
function calculatePurchaseCost()
```

```

{
    var core=document.getElementById('core').value;
    var conf=document.getElementById('configuration').value;
    var corecost=getCoreCost(core);
    var confcost=getConfigurationCost(conf);
    var totalcost=corecost+confcost;
}

```

```

var ptype=document.getElementById('ptype').value;
var tax=calculateTax(totalcost,ptype);
var server=document.getElementById('stype').value;
document.getElementById('result').innerHTML="Purchase of a "+server+" with "+conf+" has been
logged!<br>An amount of Rs."+tax+", inclusive of tax has been received by "+ptype;
}

```

Boat Ride Bill Automation-V1

index.html

```

<!DOCTYPE html>

<html>

<head>

<title>Boat Ride Bill Automation</title>

<style>

    input[type="number"] {
    width:98%;
    }
    input[type="text"] {
    width:98%;
    }
    input[type="date"] {
    width: 98%;
    }
    input[type="email"] {
    width:98%;
    }
    input[type="tel"] {
    width: 98%;
    }
    select {
    width: 98%;
    }
    body{
        margin-left: auto;
        margin-right: auto;
        width: 60%;
        background-size:60%;
    }
    form {
        margin-left: auto;

```

```

        margin-right: auto;
        text-align: center;
        width: 50%;
    }
    h1 {
        background-color: #00cc66;
        color: #FFFFFF;
        font-family: Courier New;
        font-style: italic;
        text-align: center;
    }
    td, th {
        border: 1px solid #ddd;
        padding: 8px;
    }
    #main{
        background-color: #9999ff;
        padding-top: 12px;
        padding-bottom: 12px;
        text-align: center;
        color: #FFFFFF;
        font-weight: bold;
        padding-left: 10px;
        padding-right: 10px;
    }
    #result{
        font-size:20px;
        font-weight:bold;
    }
</style>
</head>
<body>
<div id="main">
<h1>Boat Ride Bill Automation</h1>
<form onsubmit="return bookRide()">
<table>
    <tr>
        <td>Customer Name</td>
        <td><input type="text" id="cname" name="cname" placeholder="Customer
Name" /></td>
    </tr>
    <tr>
        <td>Phone Number</td>
        <td><input type="tel" id="phno" name="phno" placeholder="Phone
Number" /></td>
    </tr>
    <tr>

```

```

        <td>Email</td>
        <td><input type="email" id="email" name="email" placeholder="Email" /></td>
    </tr>
    <tr>
        <td>Number of Persons</td>
        <td><input type="text" id="noOfPersons" name="noOfPersons"
placeholder="Number of Persons" required /></td>
    </tr>
    <tr>
        <td>Boat Type</td>
        <td><select id="btype" name="btype">
            <option id="2seater" value="2 Seater Boat">2 Seater Pedal Boat</option>
            <option id="4seater" value="4 Seater Boat">4 Seater Pedal Boat</option>
            <option id="8seater" value="8 Seater Boat">8 Seater Motor Boat</option>
            <option id="15seater" value="15 Seater Boat">15 Seater Motor Boat</option>
        </select></td>
    </tr>
    <tr>
        <td>Travel Duration in Hours</td>
        <td><input type="number" id="duration" name="duration" /></td>
    </tr>
</table>

```

```

<br>
<p><input type="submit" id="submit" name="submit" value="Book Ride"/></p>
<div id="result"></div>

```

```

</form>
</div>
<script src="script.js"></script>
</body>
</html>

```

script.js

```

function bookRide(){
    var btype=document.getElementById("btype").value;
    var noOfPersons=document.getElementById("noOfPersons").value;
    var duration=document.getElementById("duration").value;
    var boatCount=getBoatCount(btype,noOfPersons);
    var boatPrice=getBoatPrice(btype,boatCount);
    var cal=calculateBill(boatPrice,duration);
    document.getElementById("result").innerHTML="You need to pay Rs."+cal;
}

function calculateBill(boatPrice,duration){
    return boatPrice*duration;
}

```

```

}

function getBoatPrice(btype,boatCount){
    if(btype=="2 Seater Boat"){
        return (boatCount*240);
    }
    if(btype=="4 Seater Boat"){
        return (boatCount*260);
    }
    if(btype=="8 Seater Boat"){
        return (boatCount*560);
    }
    if(btype=="15 Seater Boat"){
        return (boatCount*990);
    }
}

function getBoatCount(btype,noOfPersons){
    if(btype=="2 Seater Boat"){
        if (noOfPersons%2===0){
            return(parseInt(noOfPersons/2));
        }
        else{
            return(parseInt(noOfPersons/2)+1);
        }
    }
    if(btype=="4 Seater Boat"){
        if (noOfPersons%4===0){
            return(parseInt(noOfPersons/4));
        }
        else{
            return(parseInt(noOfPersons/4)+1);
        }
    }
    if(btype=="8 Seater Boat"){
        if (noOfPersons%8===0){
            return(parseInt(noOfPersons/8));
        }
        else{
            return(parseInt(noOfPersons/8)+1);
        }
    }
    if(btype=="15 Seater Motor Boat"){
        if (noOfPersons%15===0){
            return(parseInt(noOfPersons/15));
        }
        else{
            return(parseInt(noOfPersons/15)+1);
        }
    }
}

```

Singapore Tourism-V1

index.html

```
<!DOCTYPE html>
```

```
<html>
```

```
  <head>
```

```
    <title>Singapore Tourism</title>
```

```
    <style>
```

```
      input[type="number"],input[type="text"],input[type="date"],input[type="email"],input[type="tel"],select {
```

```
        width:95%;
```

```
      }
```

```
      body{
```

```
        background-color: #993366;
```

```
        font-weight: bold;
```

```
      }
```

```
      div{
```

```
        margin-left: auto;
```

```
        margin-right: auto;
```

```
        text-align: center;
```

```
        color: #FFFFFF;
```

```
        font-size: 20px;
```

```
      }
```

```
      h3{
```

```
        font-family: Verdana;
```

```
        text-align: center;
```

```
        border-radius: 6px;
```

```
        margin-left: auto;
```

```
        margin-right: auto;
```

```
        background-color: #00ccff;
```

```
        color: #FFFFFF;
```

```
        width: 50%;
```

```
        padding: 5px;
```

```
      }
```

```
      table, td, tr{
```

```
        margin-left: auto;
```

```
        margin-right: auto;
```

```
        text-align: left;
```

```
        border: solid 2px black;
```

```
        border-spacing: 1px;
```

```
        border-radius: 6px;
```

```
        width: 50%;
```

```
        padding: 1px;
```

```
        color: #009900;
```

```
        background-color: #f2f2f2 ;
```

```
      }
```

```
      ::-webkit-input-placeholder {
```



```

        color: #696969;
        font-weight: bold;
    }

    #submit{
        color: #FFFFFF;
        font-weight: bold;
        font-family: Verdana;
        background-color: #00ccff;
        border-radius: 6px;
        padding: 5px;
        width: 50%;
        margin-right: auto;
        margin-left: auto;
    }
    #result{
        color: #000000;
        font-size: 20px;
    }
</style>

</head>
<body>
<div>
<h3>Singapore Tourism</h3>
<form onsubmit="return calculateCost()">
    <table border="1">
        <tr>
            <td> Name </td>
            <td> <input type="text" id="name" required></td>
        </tr>
        <tr>
            <td> Phone no </td>
            <td> <input type="tel" id="phno" required></td>
        </tr>
        <tr>
            <td> Email ID </td>
            <td> <input type="email" id="email" required></td>
        </tr>
        <tr>
            <td> Number of Persons </td>
            <td> <input type="number" id="noOfPersons" required></td>
        </tr>
        <tr>
            <td> Prefer Stay</td>
            <td><input type="radio" id="yes" name="preferStay" value="Yes" required
onchange="disableNoOfDaysStay()">Yes
                <input type="radio" id="no" name="preferStay" value="No" required
onchange="disableNoOfDaysStay()">No</td>
        </tr>
    </table>

```

```

</tr>
<tr>
  <td> Number of Days Stay </td>
  <td> <input type="number" id="noOfDaysStay" required></td>
</tr>
<tr>
  <td>Places you would like to visit</td>
  <td>
    <input type="checkbox" name="placesOfChoice" id="Pilgrimage"
value="Pilgrimage">Places Of Pilgrimage<br>
    <input type="checkbox" name="placesOfChoice" id="Heritage"
value="Heritage">Places Of Heritage<br>
    <input type="checkbox" name="placesOfChoice" id="Hills" value="Hills">Hills<br>
    <input type="checkbox" name="placesOfChoice" id="Falls" value="Falls">Falls<br>
    <input type="checkbox" name="placesOfChoice" id="Beach"
value="Beach">Beach<br>
    <input type="checkbox" name="placesOfChoice" id="Adventures"
value="Adventures">Places Of Adventures
  </td>
</tr>
</table>

```

```

<p><input type="submit" id="submit" value="Calculate Cost"></p>

```

```

<div id="result"></div>

```

```

</form>

```

```

</div>

```

```

<script src="script.js" type="text/javascript"> </script>

```

```

</body>

```

```

</html>

```

```

script.js

```

```

function getCount()
{
  var count=0;
  if(document.getElementById("Pilgrimage").checked===true)
  {
    count+=1;
  }
  if(document.getElementById("Heritage").checked===true)
  {
    count+=1;
  }
  if(document.getElementById("Hills").checked===true)

```

```

{
    count+=1;
}
if(document.getElementById("Falls").checked===true)
{
    count+=1;
}
if(document.getElementById("Beach").checked===true)
{
    count+=1;
}
if(document.getElementById("Adventures").checked===true)
{
    count+=1;
}
return count;
}

```

function getTotalCost(noOfpersons)

```

{
    var initcost=0;
    if(document.getElementById("Pilgrimage").checked===true)
    {
        initcost+=350;
    }
    if(document.getElementById("Heritage").checked===true)
    {
        initcost+=430;
    }
    if(document.getElementById("Hills").checked===true)
    {
        initcost+=780;
    }
    if(document.getElementById("Falls").checked===true)
    {
        initcost+=1200;
    }
    if(document.getElementById("Beach").checked===true)
    {
        initcost+=270;
    }
    if(document.getElementById("Adventures").checked===true)
    {
        initcost+=4500;
    }
    return initcost*noOfpersons;
}

```

function calculateDiscount(cost)

```

{

```

```

    if(getCount()>=2)
    {
        return (cost*(85/100));
    }
    return 0;
}

function getStayCost(noOfPersons)
{
    if(document.getElementById("yes").checked===true)
    {
        var noOfDays=document.getElementById("noOfDaysStay").value;
        return noOfPersons* noOfDays *150;
    }
    return 0;
}

function disableNoOfDaysStay()
{
    if(document.getElementById("no").checked===true)
    {
        document.getElementById("noOfDaysStay").setAttribute("disabled",true);
    }
    /*if(document.getElementById("yes").checked===true)
    {
        document.getElementById("noOfDaysStay").setAttribute("disabled",false);
    }*/
}

function calculateCost()
{
    var noOfPersons=document.getElementById("noOfPersons").value;
    var totalcost=getTotalCost(noOfPersons);
    var discount=calculateDiscount(totalcost);
    var staycost=getStayCost(noOfPersons);
    var packagecost=discount+staycost;
    var res=packagecost+936;
    document.getElementById("result").innerHTML="Your preferred package cost "+res+"$";
    return false;
}

```

Monthly Instalment Estimator-V1

index.html

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>Monthly Instalment Estimator</title>
```

```
<style>
```

```
    input[type="number"] {  
        width:98%;  
    }
```

```
    input[type="text"] {  
        width:98%;  
    }
```

```
    input[type="date"] {  
        width: 98%;  
    }
```

```
    input[type="email"] {  
        width:98%;  
    }
```

```
    input[type="tel"] {  
        width: 98%;  
    }
```

```
    select {  
        width: 98%;  
    }
```

```
    body{  
        background-color:#FFAACC;  
    }
```

```
    div{  
        margin-left: auto;  
        margin-right: auto;  
        text-align: center;  
        color: #FFFFFF;  
        font-size: 20px;  
    }
```

```
h3{  
    font-family: Verdana;  
    text-align: center;  
    border-radius: 6px;
```

```

        margin-left: auto;
        margin-right: auto;
        background-color: #770080;
        color: #FFFFFFF;
        width: 50%;
        padding: 5px;
    }

    table, td, tr{
        margin-left: auto;
        margin-right: auto;
        border: solid 2px black;
        border-spacing: 1px;
        border-radius: 6px;
        width: 50%;
        padding: 1px;
        color: #000099;
        background-color: #F2F2F2 ;
    }

    ::-webkit-input-placeholder {
        color: #696969;
        font-weight: bold;
    }

    #submit{
        margin-right: auto;
        margin-left: auto;
        color: #FFFFFFF;
        font-weight: bold;
        font-family: Verdana;
        background-color: #770080;
        text-align:center;
        border-radius: 6px;
        padding: 5px;
        width: 50%;
    }

    #result{
        color: #770080;
        font-size: 20px;
        font-weight: bold;
    }

</style>
</head>
<body>
<div>
<h3>Monthly Instalment Estimator</h3>

```

```

<form onsubmit="return availLoan()">

<table>

    <tr>
        <td>Applicant Name</td>
        <td><input type="text" id="aname" name="aname" placeholder="Applicant Name"
required /></td>
    </tr>
    <tr>
        <td>Phone Number</td>
        <td><input type="tel" id="phno" name="phno" placeholder="Phone Number"
required /></td>
    </tr>
    <tr>
        <td>Email</td>
        <td><input type="email" id="email" name="email" placeholder="Email" required
/></td>
    </tr>
    <tr>
        <td>Aadhar Number</td>
        <td><input type="text" id="aadhar" name="aadhar" placeholder="Aadhar Number"
required /></td>
    </tr>
    <tr>
        <td>Pan Number</td>
        <td><input type="text" id="pan" name="pan" placeholder="Pan Number" required
/></td>
    </tr>
    <tr>
        <td>Monthly Income</td>
        <td><input type="text" id="income" name="income" placeholder="Monthly
Income" required /></td>
    </tr>
    <tr>
        <td>Loan Type</td>
        <td><select name="loanType" id="loanType">
            <option id="home" value="Home Loan">Home Loan</option>
            <option id="personal" value="Personal Loan">Personal Loan</option>
            <option id="vehicle" value="Vehicle Loan">Vehicle Loan</option>
        </select></td>
    </tr>
    <tr>
        <td>Expected Loan Amount</td>
        <td><input type="number" id="expectedAmt" name="expectedAmt" required
/></td>
    </tr>
    <tr>
        <td>Tenure In Months</td>
        <td><input type="number" id="tenure" name="tenure" required /></td>
    </tr>
</table>

```

```

<br>
<p><input type="submit" id="submit" name="submit" value="Avail Loan"/></p>
<div id="result"></div>
</form>
</div>
<script src="script.js" type="text/javascript"> </script>
</body>
</html>
script.js
function calculateEMI (income, expectedAmt, tenure, interestRatePerAnnum)
{
    var EMI, R, N;
    R=(interestRatePerAnnum/100)/12;
    N=tenure;
    EMI= (expectedAmt*R*(Math.pow((1+R),N))/(Math.pow((1+R),N)-1)).toFixed(2);
    return Math.round(EMI);
}

function getInterestRate(loanType)
{
    var intre;
    if(loanType=="Home Loan")
    {
        intre=7;
    }
    else if(loanType=="Personal Loan")
    {
        intre=7.8;
    }
    else if(loanType=="Vehicle Loan")
    {
        intre=15;
    }
    return intre;
}

function checkEligibility(income,emi)
{
    var tmp;
    tmp=income*60/100;

```



```

if(emi<=tmp)
{
    return true;
}
else
{
    return false;
}
}

```

```

function availLoan()

```

```

{
    var lt,ltt;
    ltt=document.getElementById("loanType");//
    lt=ltt.options[ltt.selectedIndex].value;//

    var irpa;
    irpa=parseFloat(getInterestRate(lt));

    var expectedLoanAmount, income, tenure, emival;
    income=parseFloat(document.getElementById("income").value);
    expectedLoanAmount=parseFloat(document.getElementById("expectedAmt").value);
    tenure=parseFloat(document.getElementById("tenure").value);
    emival=parseInt(calculateEMI(income, expectedLoanAmount, tenure, irpa));
    var elig=checkEligibility(income, emival);

    if(elig===true)
    {
        document.getElementById("result").innerText="You are eligible to get a loan amount as
"+expectedLoanAmount+"and emi per month is "+emival;
    }
    else
    {
        document.getElementById("result").innerText="You are not eligible";
    }
    return false;
}

```

Automatic evaluation[+]

Driver.cs

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6 using System.Data.SqlClient;
7 using System.Collections;
8 using System.Data;
9 using System.Configuration;
10
11 namespace TicketManagement //DO NOT change the namespace name
12 {
13     public class Program //DO NOT change the class name
14     {
15
16         static void Main(string[] args) //DO NOT change the 'Main' method signature
17         {
18             //Implement the code here
19             char choice = 'y';
20             Console.WriteLine("Enter Ticket Details: ");
21             while( choice == 'y')
22             {
23                 Console.WriteLine("Enter Passenger Id:");
24                 string id = Console.ReadLine();
25                 Console.WriteLine("Enter Passenger Name:");
26                 string name = Console.ReadLine();
27                 Console.WriteLine("Enter Travel Date:");
28                 string date = Console.ReadLine();
29                 Console.WriteLine("Enter Distance Travelled:");
30                 int dist = Convert.ToInt32(Console.ReadLine());
31                 DistanceValidator dv = new DistanceValidator();
32                 while ( dv.ValidateTravelDistance(dist) == "true")
33                 {
34                     Console.WriteLine("Given distance is invalid");
35                     Console.WriteLine("Enter Distance Travelled: ");
36                     dist = Convert.ToInt32(Console.ReadLine());
37                 }
38                 TicketDetail td = new TicketDetail(id, name, date, dist);
39                 TicketBooking tb = new TicketBooking();
40                 tb.CalculateCost(td);
41                 tb.AddTicket(td);
42                 Console.WriteLine(td.PassengerId);
43                 Console.WriteLine(td.PassengerName);
44                 Console.WriteLine(td.TravelDate);
45                 Console.WriteLine(td.DistanceTravel);
46                 Console.WriteLine($"Ticket Cost : {td.TicketCost}");
47                 Console.WriteLine("Book Another Ticket (y/n): ");
48                 choice =Convert.ToChar(Console.ReadLine());
49             }
50         }
51     }
52     public class DistanceValidator
53     { //DO NOT change the class name
54
55         public String ValidateTravelDistance(int distance) //DO NOT change the method signature
56         {
57             //Implement code here
58             if(distance < 0)
59             {
60                 return "Given distance is invalid";
61             }
62             else
63             {
```

```

64         return "";
65     }
66 }
67 }
68 }
69

```

App.config

```

1 <!-- THIS IS FOR REFERENCE ONLY. YOU ARE NOT REQUIRED TO MAKE ANY CHANGES HERE -->
2
3 <?xml version="1.0" encoding="utf-8" ?>
4 <configuration>
5     <startup>
6         <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.6.1" />
7     </startup>
8     <connectionStrings>
9         <add name="SqlCon"
connectionString="server=localhost;database=TicketBookingDB;uid=XXXXXX;password=XXXXXX;" />
10    </connectionStrings>
11 </configuration>

```

DBHandler.cs

```

1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6 using System.Data.SqlClient;
7 using System.Configuration;
8
9 namespace TicketManagement          //DO NOT change the namespace name
10
11 {
12     public class DBHandler          //DO NOT change the class name
13     {
14         //Implement the methods as per the description
15         public DBHandler() { }
16
17         public SqlConnection GetConnection()
18         {
19             return new SqlConnection(ConfigurationManager.ConnectionStrings["SqlCon"].ConnectionString);
20         }
21     }
22 }
23
24 }
25

```

TicketBooking.cs

```

1 using System;
2 using System.Collections;
3 using System.Collections.Generic;
4 using System.Data;
5 using System.Data.SqlClient;
6 using System.Linq;
7 using System.Text;
8 using System.Threading.Tasks;
9
10 namespace TicketManagement          //DO NOT change the namespace name
11 {
12     public class TicketBooking      //DO NOT change the class name
13     {
14         //Implement the property as per the description
15         public SqlConnection Sqlcon { get; set; }
16
17         public TicketBooking() { }
18         DBHandler d = new DBHandler();
19         public void AddTicket(TicketDetail detail)

```

```

20     {
21
22         string query = "INSERT INTO TicketBooking VALUES (@id, @name, @date, @dist, @cost)";
23         using (SqlConnection con = d.GetConnection())
24         using (SqlCommand cmd = new SqlCommand(query, con))
25         {
26             cmd.Parameters.Add("@id", SqlDbType.VarChar).Value = detail.PassengerId;
27             cmd.Parameters.Add("@name", SqlDbType.VarChar).Value = detail.PassengerName;
28             cmd.Parameters.Add("@date", SqlDbType.VarChar).Value = detail.TravelDate;
29             cmd.Parameters.Add("@dist", SqlDbType.Int).Value = detail.DistanceTravel;
30             cmd.Parameters.Add("@cost", SqlDbType.Float).Value = detail.TicketCost;
31             con.Open();
32
33             try
34             {
35                 cmd.ExecuteNonQuery();
36             }
37             catch (Exception e)
38             {
39                 Console.WriteLine(e.Message);
40             }
41             finally
42             {
43                 con.Close();
44             }
45         }
46     }
47
48     //Implement the methods as per the description
49     public void CalculateCost(TicketDetail detail)
50     {
51         if(detail.DistanceTravel <= 100)
52         {
53             detail.TicketCost = detail.DistanceTravel * 1;
54         }
55         else if(detail.DistanceTravel >100 && detail.DistanceTravel <= 300)
56         {
57             detail.TicketCost = detail.DistanceTravel * 1.5;
58         }
59         else if (detail.DistanceTravel > 300 && detail.DistanceTravel <= 500)
60         {
61             detail.TicketCost = detail.DistanceTravel * 2.5;
62         }
63         else if (detail.DistanceTravel > 500)
64         {
65             detail.TicketCost = detail.DistanceTravel * 4.5;
66         }
67     }
68
69 }
70 }
71

```

TicketDetail.cs

```

1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace TicketManagement           //DO NOT change the namespace name
8  {
9      public class TicketDetail         //DO NOT change the class name
10     {
11         //Implement the fields and properties as per description
12
13         private string passengerId;

```

```

14     private string passengerName;
15     private string travelDate;
16     private int distanceTravel;
17     private double ticketCost;
18
19     public string PassengerId
20     {
21         get { return passengerId; }
22         set { this.passengerId = value; }
23     }
24     public string PassengerName
25     {
26         get { return passengerName; }
27         set { this.passengerName = value; }
28     }
29     public string TravelDate
30     {
31         get { return travelDate; }
32         set { this.travelDate = value; }
33     }
34     public int DistanceTravel
35     {
36         get { return distanceTravel; }
37         set { this.distanceTravel = value; }
38     }
39     public double TicketCost
40     {
41         get { return ticketCost; }
42         set { this.ticketCost = value; }
43     }
44
45
46     public TicketDetail() { }
47     public TicketDetail(string passengerId, string passengerName, string travelDate, int distanceTravel)
48     {
49         this.passengerId = passengerId;
50         this.passengerName = passengerName;
51         this.travelDate = travelDate;
52         this.distanceTravel = distanceTravel;
53     }
54 }
55 }
56
57 }
58

```

Grade

Reviewed on Friday, 7 January 2022, 7:32 PM by Automatic grade

Grade 100 / 100

Assessment report

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

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 validateCusomerId(){

        // 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 validateCusomerId(){

    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 validateCusomerId(){

    // 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;

    }

}

UIInterface.java

import java.util.Scanner;

public class UIInterface {

    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.validateCusomerId()){
    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.validateCusomerId()){
    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.validateCusomerId()){
    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());
                }
            }
        }
    }
}

```

}

```
package com.utility;
```

```
public class Player {
```

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

198

//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=83.33333333333333

so this method should return 83.33333333333333

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 case

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;
    }
    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]);
    }
}
}

```

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");
        }
    }
}
```

ChequePaymentProcess

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"));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return paymentList;
    }
}

```

```

        payment.setAmount(resultSet.getInt("amount"));
        paymentList.add(payment);
    }

    } 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
ent");

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

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

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

        validateMethodSignature("getConnection:java.sql.Connection","com.cts.paymentProcess.uti
l.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 incorrect;
    }
}

```

```

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"),p
rops.getProperty("DB_PASSWORD"));
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
    catch(IOException e){
        e.printStackTrace();
    }

    return con;

}

}

```

App.java

```
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");
    }

    }while(true);

}

}

```

Passenger 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("");
    }
}
}

```

ExamScheduler

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 rowCount = 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)

                rowCount=rowCount+1;

        }

    } catch(SQLException e){

    }

    return rowCount;

}

```

```

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(te
mp[4]),Period.parse(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"),p
    rops.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 FileUtil {

    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;

    }

}

```

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);

    }

}
}

```

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.LocalDateTime;
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 funtionalClass = "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(funtionalClass, 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(funtionalClass, "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. "
the skeleton");
        + "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. "
with the skeleton");
        + "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");
    }
}
}
}

```

Find MemberShip Category 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());
        }

    }

}

```

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++;  
        }  
    }  
  
}
```

```
}
```

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;

    }

}

```



```
}  
  
}
```

UserInfo.java

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

```
public class UserInfo {
```

```
    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");
}

}

```

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])/([0-2]|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 sdf=new SimpleDateFormat("dd/MM/yyyy");
    Date d1=sdf.parse(date1);
    Date d2=sdf.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 sdf=new SimpleDateFormat("dd/MM/yyyy");
    try{
        Date mydate=sdf.parse(date);
        c.setTime(mydate);
        c.add(Calendar.DATE, days);
    }catch(ParseException e){
        e.printStackTrace();
    }
    String toDate=sdf.format(c.getTime());
    return toDate;
}
}

```

TravelRequestSystem

TravelRequestDao.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("DB_URL"),props.getProperty("DB_USERNAME"),p
rops.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
}
}

```

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");
        }
    }

}

}

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";
        }
    }
    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,getT
ravelDetails: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 incorrect;
    }

```

```

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;
    }
}

```

AirVoice - Registration

Customer.java

```
public class Customer {  
    private String customerName;  
  
    private long contactNumber;  
  
    private String emailId;  
  
    private int age;  
  
    public String getCustomerName() {  
        return customerName;  
    }  
  
    public void setCustomerName(String customerName) {  
        this.customerName = customerName;  
    }  
  
    public long getContactNumber() {  
        return contactNumber;  
    }  
  
    public void setContactNumber(long contactNumber) {  
        this.contactNumber = contactNumber;  
    }  
  
    public String getEmailId() {  
        return emailId;  
    }  
}
```

```

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

        public int getAge() {
            return age;
        }

        public void setAge(int age) {
            this.age = age;
        }

    }

Main.java
import java.util.Scanner;

public class Main {

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

        Scanner sc=new Scanner(System.in);
        Customer c=new Customer();
        System.out.println("Enter the Name:");
        String name=(sc.nextLine());
        System.out.println("Enter the ContactNumber:");
        long no=sc.nextLong();
        sc.nextLine();
        System.out.println("Enter the EmailId:");
        String mail=sc.nextLine();
    }
}

```

```

System.out.println("Enter the Age:");
int age=sc.nextInt();
c.setCustomerName(name);
c.setContactNumber(no);
c.setEmailId(mail);
c.setAge(age);
System.out.println("Name:"+c.getCustomerName());
System.out.println("ContactNumber:"+c.getContactNumber());
System.out.println("EmailId:"+c.getEmailId());
System.out.println("Age:"+c.getAge());

    }

}

```

Alliteration

```

import java.util.*;

public class Main {

    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]=='a' || (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");
}

}

}

```

Alternate Numbers Difference

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



```

public class Main {

    public static void main (String[] args) {

        Scanner sc=new Scanner(System.in);

        int n1 = 0;

        int n2 = 0;

        int a[] = new int[9];

        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[]=new int[10];

        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;
        }
    }
}
}

```

BonBon Publishing Company

Advertisement.java

abstract public class Advertisement

```

{
    protected int advertisementId;
    protected String priority;

```

```
protected int noOfDays;
protected String clientName;

abstract public float calculateAdvertisementCharge(float baseCost);

public int getAdvertisementId() {
    return advertisementId;
}

public void setAdvertisementId(int advertisementId) {
    this.advertisementId = advertisementId;
}

public String getPriority() {
    return priority;
}

public void setPriority(String priority) {
    this.priority = priority;
}

public int getNoOfDays() {
    return noOfDays;
}

public void setNoOfDays(int noOfDays) {
    this.noOfDays = noOfDays;
}
```

```

    public String getClientName() {
        return clientName;
    }

    public void setClientName(String clientName) {
        this.clientName = clientName;
    }

    public Advertisement(int advertisementId, String priority, int noOfDays, String clientName) {
        super();
        this.advertisementId = advertisementId;
        this.priority = priority;
        this.noOfDays = noOfDays;
        this.clientName = clientName;
    }
}

```

ImageAdvertisement

```

public class ImageAdvertisement extends Advertisement
{
    private int inches;

    public ImageAdvertisement(int advertisementId, String priority, int noOfDays, String
clientName, int inches) {
        super(advertisementId, priority, noOfDays, clientName);
        this.inches = inches;
    }

    public int getInches() {
        return inches;
    }
}

```

```

    public void setInches(int inches) {
        this.inches = inches;
    }

```

// Override the abstract method

@Override

```

public float calculateAdvertisementCharge(float baseCost){
    float baseAdvertisementCost=baseCost*inches*noOfDays;
    float boosterCost=0f;
    float serviceCost=0f;
    if(priority.equals("high")){
        boosterCost+=baseAdvertisementCost*0.1f;
        serviceCost+=1000;
    }
    else if(priority.equals("medium")){
        boosterCost+=baseAdvertisementCost*0.07f;
        serviceCost+=700;
    }
    else if(priority.equals("low")){
        serviceCost+=200;
    }
    return baseAdvertisementCost+boosterCost+serviceCost;
}
}

```

Main.java

```

import java.util.Scanner;

public class Main {

```

```

public static void main(String[] args) {

    //Type your code here

    Scanner input=new Scanner(System.in);

    System.out.println("Enter the advertisement id");

    int id=input.nextInt();

    System.out.println("Enter the priority (high / medium / low)");

    String priority=input.next();

    System.out.println("Enter the no of days advertisement is published");

    int noOfDays=input.nextInt();

    input.nextLine();

    System.out.println("Enter the client name");

    String clientName=input.nextLine();

    System.out.println("Enter the type of Advertisement (video/image/text)");

    String type=input.next();

    if(type.equals("video")){

        System.out.println("Enter the duration in minutes");

        int duration=input.nextInt();

        VideoAdvertisement ad1=new
VideoAdvertisement(id,priority,noOfDays,clientName,duration);

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

        float baseCost=(float)input.nextDouble();

        System.out.printf("The Advertisement cost is
%.1f",ad1.calculateAdvertisementCharge(baseCost));

    }

    else if(type.equals("image")){

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

        int inches=input.nextInt();

        ImageAdvertisement ad1=new
ImageAdvertisement(id,priority,noOfDays,clientName,inches);

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

        float baseCost=(float)input.nextDouble();

```

```

        System.out.printf("The Advertisement cost is
%.1f",ad1.calculateAdvertisementCharge(baseCost));
    }
    else if(type.equals("text")){
        System.out.println("Enter the number of characters");
        int characters=input.nextInt();

        TextAdvertisement ad1=new
TextAdvertisement(id,priority,noOfDays,clientName,characters);

        System.out.println("Enter the base cost");
        float baseCost=(float)input.nextDouble();

        System.out.printf("The Advertisement cost is
%.1f",ad1.calculateAdvertisementCharge(baseCost));
    }

}

```

TextAdvertisement

```

public class TextAdvertisement extends Advertisement
{
    private int noOfCharacters;

    public int getNoOfCharacters() {
        return noOfCharacters;
    }

    public void setNoOfCharacters(int noOfCharacters) {
        this.noOfCharacters = noOfCharacters;
    }
}

```

```

        public TextAdvertisement(int advertisementId, String priority, int noOfDays, String
clientName,

                                int noOfCharacters) {

        super(advertisementId, priority, noOfDays, clientName);

        this.noOfCharacters = noOfCharacters;

    }

```

// Override the abstract method

@Override

```

public float calculateAdvertisementCharge(float baseCost){

    float baseAdvertisementCost=baseCost*noOfCharacters*noOfDays;

    float boosterCost=0f;

    float serviceCost=0f;

    if(priority.equals("high")){

        boosterCost+=baseAdvertisementCost*0.1f;

        serviceCost+=1000;

    }

    else if(priority.equals("medium")){

        boosterCost+=baseAdvertisementCost*0.07f;

        serviceCost+=700;

    }

    else if(priority.equals("low")){

        serviceCost+=200;

    }

    return baseAdvertisementCost+boosterCost+serviceCost;

}

}

```

VideoAdvertisement

```

public class VideoAdvertisement extends Advertisement

{

```



```

private int duration;

    public VideoAdvertisement(int advertisementId, String priority, int no_of_days, String clientName,
int duration) {

        super(advertisementId, priority, no_of_days,clientName);

        this.duration = duration;

    }

    public int getDuration() {

        return duration;

    }

    public void setDuration(int duration) {

        this.duration = duration;

    }


// Override the abstract method
@Override
public float calculateAdvertisementCharge(float baseCost){

    float baseAdvertisementCost=baseCost*duration*noOfDays;

    float boosterCost=0f;

    float serviceCost=0f;

    if(priority.equals("high")){

        boosterCost+=baseAdvertisementCost*0.1f;

        serviceCost+=1000;

    }

    else if(priority.equals("medium")){

        boosterCost+=baseAdvertisementCost*0.07f;

        serviceCost+=700;

    }

    else if(priority.equals("low")){

```

```

        serviceCost+=200;
    }
    return baseAdvertisementCost+boosterCost+serviceCost;
}
}

```

Call Details

Call.java

```

public class Call {
    private int callId;
    private long calledNumber;
    private float duration;

    public void Call(){

    }

    public void parseData(String data){

        this.callId=Integer.parseInt(data.substring(0,3));
        this.calledNumber=Long.parseLong(data.substring(4,14));
        this.duration=Float.parseFloat(data.substring(15));

    }

    public int getCallId(){
        return this.callId;
    }
    public long getCalledNumber(){
        return this.calledNumber;
    }
    public float getDuration(){

```

```
        return this.duration;
    }
}
```

```
}
```

Main.java

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

```
public class Main {
```

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

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

```
        Call obj=new Call();
```

```
        System.out.println("Enter the call details:");
```

```
        String data = sc.nextLine();
```

```
        obj.parseData(data);
```

```
        System.out.println("Call id:"+obj.getCallId());
```

```
        System.out.println("Called number:"+obj.getCalledNumber());
```

```
        System.out.println("Duration:"+obj.getDuration());
```

```
    }
```

```
}
```

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
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;
```

```
/**
```

```
 * @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.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 methodWithExcpn, String className) {

    Class cls = null;

    try {

        String[] actualmethods = methodWithExcpn.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

```

```

        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
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
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,countOf
BulkOrderLineItems: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 incorrect;

    }

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

            String[] actualmethods = methodWithExcpn.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");
}
}
}
}
}
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;
    }
}

```

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 gradeobtained is A, its equivalent grade points is 9 has to 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

$$\text{GPA} = (\text{gradePoint1} + \text{gradePoint2} + \dots + \text{gradePointN}) / (\text{size of List})$$

For Example:

if the list contains the following marks [9,10,8,5]

$$\text{GPA} = (9 + 10 + 8 + 5) / (4) = 8.0$$

*/

```

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;
    }
}

```

InsurancePremiumGenerator_v2

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.getHous
eholdValuation();

    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:bool
            ean,calculatePremiumByPropertyAge:double,calculatePremiumForBurglaryCoverage:double,calculat
            ePremiumForPoliticalUnrestCoverage: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;

        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 methodWithExcpn, String className) {

    Class cls = null;

    try {

        String[] actualmethods = methodWithExcpn.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");
    }
}

```

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

}

}

```

Oil Stores

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));
}
}

```

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;
    }
}

```

Power Progress

```
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)+" ");
        }

    }
}
```

Singapore Tourism

```
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
}

}

```

Code RED

CasualEmployee:

```
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 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;
}
}

```

PermanentEmployee:

```

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;
}
}

```

TraineeEmployee:

```

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;
} }

```

UserInterface:

```

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");
} }

```


BookAMovieTicket:

```
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;

}

}

```

GoldTicket:

```

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;

}

}

```

PlatinumTicket:

```
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;

}

}

```

SilverTicket:

```

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;

}

}

```

UserInterface:

```

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);

}

}

}

}
```


TICKET RESERVATION

INVALID CARRIER-

```
public class InvalidCarrierException extends Exception{
    //FILL THE CODE HERE
    public InvalidCarrierException (String message){
        super(message);
    }
}
```

PASSENGER.JAVA-

//DO NOT EDIT OR ADD ANY CODE

```
public class Passenger {

    private String passengerName;
    private long phoneNumber;
    private String emailId;
    private String carrierName;
    private String dateOfJourney;
    private String source;
    private String destination;

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

    public Passenger(String passengerName, long phoneNumber, String emailId,
String carrierName, String dateOfJourney,
        String source, String destination) {
        super();
        this.passengerName = passengerName;
        this.phoneNumber = phoneNumber;
        this.emailId = emailId;
        this.carrierName = carrierName;
        this.dateOfJourney = dateOfJourney;
        this.source = source;
        this.destination = destination;
    }

    public String getPassengerName() {
        return passengerName;
    }
    public void setPassengerName(String passengerName) {
        this.passengerName = passengerName;
    }

    public long getPhoneNumber() {
        return phoneNumber;
    }
    public void setPhoneNumber(long phoneNumber) {
        this.phoneNumber = phoneNumber;
    }
    public String getEmailId() {
        return emailId;
    }
}
```

```

    }
    public void setEmailId(String emailId) {
        this.emailId = emailId;
    }
    public String getCarrierName() {
        return carrierName;
    }
    public void setCarrierName(String carrierName) {
        this.carrierName = carrierName;
    }
    public String getDateOfJourney() {
        return dateOfJourney;
    }
    public void setDateOfJourney(String dateOfJourney) {
        this.dateOfJourney = dateOfJourney;
    }
    public String getSource() {
        return source;
    }
    public void setSource(String source) {
        this.source = source;
    }
    public String getDestination() {
        return destination;
    }
    public void setDestination(String destination) {
        this.destination = destination;
    }
}

```

PASSENGER CATERGORY-

```

import java.util.List;
@FunctionalInterface
public interface PassengerCategorization {
    abstract public List<Passenger> retrievePassenger_BySource(List<Passenger>
passengerRecord,String source);
}

```

PASSENGER UTILITY-

```

import java.util.List;
import java.io.*;
import java.util.*;

public class PassengerUtility {

    public List<Passenger> fetchPassenger(String filePath) throws Exception{

        //FILL THE CODE HERE
        List<Passenger> list = new ArrayList<Passenger>();
        String line = "";
        String splitBy = ",";
    }
}

```

```

        BufferedReader br = new BufferedReader(new FileReader(filePath));
        while((line=br.readLine())!=null){
            String[] p = line.split(splitBy);
            Passenger passenger = new
Passenger(p[0],Long.parseLong(p[1]),p[2],p[3],p[4],p[5],p[6]);
            if(isValidCarrierName(passenger.getCarrierName())){
                list.add(passenger);
            }
        }

        return list;
    }

    public boolean isValidCarrierName (String carrierName)
    {
        //FILL THE CODE HERE
        String temp = carrierName;
        if((temp.toLowerCase()).equals("bella")){
            return true;
        }else{
            try{
                throw new InvalidCarrierException(carrierName+" is an Invalid carrier
name.");
            }
            catch(InvalidCarrierException e){
                System.out.println(e.getMessage());
            }
        }
        return false;
    }
}

```

SKELETON VALIDATION-

```

import java.lang.reflect.Method;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.stream.Stream;

/**
 * @author TJ
 *
 * This class is used to verify if the Code Skeleton is intact and not modified by
participants thereby ensuring smooth auto evaluation
 *
 */
public class SkeletonValidator {

    public SkeletonValidator() {

        validateClassName("PassengerCategorization");
        validateClassName("Passenger");
        validateClassName("InvalidCarrierException");
    }
}

```

```

        validateClassName("PassengerUtility");

        validateMethodSignature(
            "retrievePassenger_BySource:java.util.List",
            "PassengerCategorization");
        validateMethodSignature(
            "fetchPassenger:java.util.List",
            "PassengerUtility");
        validateMethodSignature(
            "isValidCarrierName:boolean",
            "PassengerUtility");
        validateMethodSignature(
            "searchPassengerRecord:PassengerCategorization",
            "UserInterface");
    }

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

    protected final boolean validateClassName(String className) {
        boolean incorrect = false;
        try {
            Class.forName(className);
            incorrect = true;
            LOG.info("Class Name " + className + " is correct");

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

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

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

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

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

                methodName = methodSignature[0];
                returnType = methodSignature[1];
            }
        }
    }

```

```

        cls = Class.forName(className);
        Method[] methods = cls.getMethods();
        for (Method findMethod : methods) {
            if (methodName.equals(findMethod.getName())) {
                foundMethod = true;
                if (!
(findMethod.getReturnType().getName().equals(returnType))) {
                    errorFlag = true;
                    LOG.log(Level.SEVERE, " You have changed
the " + "return type in '" + methodName
                                + "' method. Please stick to
the " + "skeleton provided");
                } else {
                    LOG.info("Method signature of " +
methodName + " is valid");
                }
            }
        }
        if (!foundMethod) {
            errorFlag = true;
            LOG.log(Level.SEVERE, " Unable to find the given
public method " + methodName
                                + ". Do not change the " + "given public
method name. " + "Verify it with the skeleton");
        }
    }
    if (!errorFlag) {
        LOG.info("Method signature is valid");
    }
} catch (Exception e) {
    LOG.log(Level.SEVERE,
            " There is an error in validating the " + "method
structure. Please manually verify that the "
            + "Method signature is same as the
skeleton before uploading");
}
}
}

```

USER INTERFACE-

```

import java.util.*;
import java.io.*;

public class UserInterface{
    public static PassengerCategorization searchPassengerRecord(){
        //FILL THE CODE HERE
        return (list,source)->{
            List<Passenger> result = new ArrayList<Passenger>();
            for(Passenger pass : list){
                if((pass.getSource().toLowerCase()).equals(source.toLowerCase())){

```

```

        result.add(pass);
    }
}
return result;
};
}

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

    PassengerCategorization pc = searchPassengerRecord();
    //FILL THE CODE HERE
    System.out.println("Invalid Carrier Records are:");
    PassengerUtility pu = new PassengerUtility();
    List<Passenger> list = null;
    try{
        list = pu.fetchPassenger(new String("PassengerRecord.txt"));
    }
    catch(FileNotFoundException e){
        e.printStackTrace();
    }
    catch(IOException e){
        e.printStackTrace();
    }
    catch(Exception e){
        e.printStackTrace();
    }
    System.out.println("Enter the source to search");
    Scanner sc = new Scanner(System.in);
    String inp = sc.next();

    List<Passenger> result = pc.retrievePassenger_BySource(list,inp);
    if(result.size()==0){
        System.out.println("No Passenger Record");
    }
    else{
        for(Passenger passenger: result){
            System.out.println(passenger.getPassengerName()+"
"+passenger.getPhoneNumber()+" "+passenger.getDateOfJourney()+" "+
passenger.getDestination());
        }
    }
}
}

```

ZEE LAPTOP AGENCY

INVALID LAPTOP-

package com.cts.zeelaptopagency.exception;

```

public class InvalidLaptopIdException extends Exception{
    public InvalidLaptopIdException() {

    }

    public InvalidLaptopIdException(String string) {
        super(string);
    }

}

```

MAIN.JAVA-

```

package com.cts.zeelaptopagency.main;
import com.cts.zeelaptopagency.service.LaptopService;
import java.util.*;
import com.cts.zeelaptopagency.skeletonvalidator.SkeletonValidator;
import java.io.*;
import com.cts.zeelaptopagency.vo.Laptop;
import com.cts.zeelaptopagency.exception.*;

public class Main {
    public static void main(String args[]) {

        // CODE SKELETON - VALIDATION STARTS
        // DO NOT CHANGE THIS CODE
        new SkeletonValidator();
        // CODE SKELETON - VALIDATION ENDS

        //Add your code here to retrieve file object from Service
class
        //Add Code here to print valid LaptopDetails returned by
Service Method

        LaptopService l=new LaptopService();
        File f=l.accessFile();
        List<Laptop> lap=l.readData(f);
        System.out.println("The Valid Laptop Details are:-");
        for(Laptop la:lap)
        {
            try{
                if(l.validate(la.getLaptopId())==true){
                    System.out.println(la.toString());
                }
            }
            catch(InvalidLaptopIdException e)
            {
                e.printStackTrace();
            }
        }

    }
}

```

LAPTOP SERVICE.JAVA-

```

package com.cts.zeelaptopagency.service;

```

```

import com.cts.zeelaptopagency.vo.Laptop;
import java.io.File;
import java.io.*;
import java.util.List;
import java.util.*;

import com.cts.zeelaptopagency.exception.InvalidLaptopIdException;
import com.cts.zeelaptopagency.vo.Laptop;

public class LaptopService {

    /**
     * Method to access file
     *
     * @return File
     */
    public File accessFile()
    {

        //Type Code to open text file here
        //File f=new File("LaptopDetails.txt");

        return new File("LaptopDetails.txt"); //TODO change this return value
    }

    /**
     * Method to validate LaptopId and, for invalid laptopId throw
     InvalidLaptopIdException with laptopId as argument
     *
     * @param laptopid
     * @return status
     */
    public boolean validate(String laptopId)throws InvalidLaptopIdException {

        if(laptopId.toUpperCase().startsWith("ZEE"))
        {

        }else{
            throw new InvalidLaptopIdException(laptopId);
        }
        return true;
        //TODO change this return value
    }

    /**
     * Method to read file ,Do necessary operations , writes validated data to
     List and prints invalid laptopID in its catch block
     *
     * @param file
     * @return List
     */
    public List<Laptop> readData(File file)
    { String s1="";

```



```

int c;
FileInputStream file1;
List<Laptop> lap=new LinkedList<>(); ;
try{
    file1=new FileInputStream(file);

    while((c=file1.read())!=-1)
    {
        s1+=(char)c;
    }
}catch(FileNotFoundException e)
{
    e.printStackTrace();
}catch(IOException e)
{
    e.printStackTrace();
}
String[] arr=s1.split("\n");
String[] laptopids=new String[4];
Laptop l;
for(String s:arr)
{

    l=new Laptop();
    laptopids=s.split(",");
    l.setLaptopId(laptopids[0]);
    l.setCustomerName(laptopids[1]);
    l.setBasicCost(Double.parseDouble((laptopids[2])));
    l.setNoOfDays(Integer.parseInt(laptopids[3]));
    this.calculateFinalAmount(l);
    l.setTotalAmount(l.getBasicCost()*l.getNoOfDays());
    lap.add(l);

}

    return lap; //TODO change this return value
}

/**
 * Method to find and set totalAmount based on basicCost and noOfdays
 *
 *
 */
public void calculateFinalAmount(Laptop l)
{
    //Type code here to calculate totalAmount based on no of days and basic
cost    double d=l.getBasicCost()*l.getNoOfDays();
    l.setTotalAmount(d);

```

```

    }
}

```

SKELETON VALIDATOR-

```

package com.cts.zeelaptopagency.skeletonvalidator;

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

public class SkeletonValidator {

    public SkeletonValidator() {

        validateClassName("com.cts.zeelaptopagency.service.LaptopService");
        validateClassName("com.cts.zeelaptopagency.vo.Laptop");

        validateMethodSignature(
            "accessFile:java.io.File,validate:boolean,readData:java.util.List",
            "com.cts.zeelaptopagency.service.LaptopService");

    }

    private static final Logger LOG =
        Logger.getLogger("SkeletonValidator");
    protected final boolean validateClassName(String className) {

        boolean isincorrect = false;
        try {
            Class.forName(className);
            isincorrect = 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 isincorrect;

    }

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

```

```

Class cls = null;
try {

    String[] actualMethods = methodWithExcpn.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 (!
changed the " + "return type in '" + methodName
stick to the " + "skeleton provided");
                LOG.log(Level.SEVERE, " You have
+ "'" method. Please

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

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

```

LAPTOP.JAVA-

```
package com.cts.zeelaptopagency.vo;
/**
 * Value Object - Laptop
 *
 */
public class Laptop {
    private String laptopId;
    private String customerName;
    private double basicCost;
    private int noOfDays;
    private double totalAmount;

    public Laptop()
    {

    }
    public String toString()
    {
        return "Laptop [laptopId="+this.getLaptopId()+",
customerName="+this.getCustomerName()+", basicCost="+this.getBasicCost()+",
noOfDays="+this.getNoOfDays()+", totalAmount="+this.getTotalAmount()+"]";
    }
    public String getLaptopId() {
        return laptopId;
    }
    public void setLaptopId(String laptopId) {
        this.laptopId = laptopId;
    }
    public String getCustomerName() {
        return customerName;
    }
    public void setCustomerName(String customerName) {
        this.customerName = customerName;
    }
    public double getBasicCost() {
        return basicCost;
    }
    public void setBasicCost(double basicCost) {
        this.basicCost = basicCost;
    }
    public int getNoOfDays() {
        return noOfDays;
    }
    public void setNoOfDays(int noOfDays) {
        this.noOfDays = noOfDays;
    }
    public double getTotalAmount() {
        return totalAmount;
    }
    public void setTotalAmount(double totalAmount) {
```

```

        this.totalAmount = totalAmount;
    }

```

```

}

```

LAPTOP DETAILS-

Laptop Details:

```

ZEE01, Jack, 2000.50, 4
ZEE02, Dev, 4000.00, 3
EEZ03, John, 4500.00, 5
ZAE04, Milan, 3500.00, 4
ZEE05, Surya, 2500.50, 7
ZEE06, Milan, 5000.00, 6

```

DOLLAR CITY THEME PARK

USER INTERFACE-

```

package com.ui;

import java.util.Scanner;

import com.utility.ThemeParkB0;

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

        // Fill the UI code
        boolean flag = true;
        int choice = 0;
        ThemeParkB0 park = new ThemeParkB0();
        while(flag){

            System.out.println("1.Add booking details");
            System.out.println("2.Average customer booked");
            System.out.println("3.Exit");
            System.out.println("Enter your choice");
            choice = sc.nextInt();

            switch(choice){
                case 1:
                    System.out.println("Enter the day");
                    String day = sc.next();
                    System.out.println("Enter the customer count");
                    int cc = sc.nextInt();
                    park.addBookingDetails(cc);

                    break;
                case 2:
                    double res = park.findAverageCustomerBooked();
                    if(res==0){

```

```

        System.out.println("No records found");
        //break;
    }
    else{
        System.out.println(res);
        //break;
    }
    break;
case 3:
    System.out.println("Thank you for using the application");
    flag = false;
    break;
}
}
}
}
}

```

THEMEPARKBO.JAVA-

```

package com.utility;

import com.ui.UserInterface;
import java.util.*;
import java.util.List;

public class ThemeParkBO {

    private List<Integer> bookingList = new ArrayList<>();

    public List<Integer> getBookingList() {
        return bookingList;
    }

    public void setBookingList(List<Integer> bookingList) {
        this.bookingList = bookingList;
    }

    // This Method should add the customerCount passed as argument into the
    // bookingList

    public void addBookingDetails(int customerCount) {

        // Fill the Code here
        bookingList.add(customerCount);

    }

    /*
    * This method should return the average customer booked based on the
    * customerCount values available in the bookingList.
    */

    public double findAverageCustomerBooked() {
        double avg;

        // Fill the Code here
        double count = 0;
    }
}

```

```

        double counter = 0;
        for(int i=0;i<bookingList.size();++i){
            count+=bookingList.get(i);
            counter++;
        }

        if(counter==0) return 0;
        avg = count/counter;
        return avg;
    }
}

```

PASSENGER

PASSENGER UTILITY-

```

import java.util.List;
import java.io.*;
import java.util.*;

```

```

public class PassengerUtility {

    public List<Passenger> fetchPassenger(String filePath) throws Exception{

        //FILL THE CODE HERE
        List<Passenger> list = new ArrayList<Passenger>();
        String line = "";
        String splitBy = ",";

        BufferedReader br = new BufferedReader(new FileReader(filePath));
        while((line=br.readLine())!=null){
            String[] p = line.split(splitBy);
            Passenger passenger = new
Passenger(p[0],Long.parseLong(p[1]),p[2],p[3],p[4],p[5],p[6]);
            if(isValidCarrierName(passenger.getCarrierName())){
                list.add(passenger);
            }
        }

        return list;
    }

    public boolean isValidCarrierName (String carrierName)
    {
        //FILL THE CODE HERE
        String temp = carrierName;
        if((temp.toLowerCase()).equals("bella")){
            return true;
        }else{
            try{
                throw new InvalidCarrierException(carrierName+" is an Invalid carrier
name.");
            }
            catch(InvalidCarrierException e){
                System.out.println(e.getMessage());
            }
        }
    }
}

```

```

        }
        return false;
    }
}

```

PASSENGER CATEGORIZATION-

```

//DO NOT ADD OR EDIT ANY CODE HERE
import java.util.List;
@FunctionalInterface
public interface PassengerCategorization {
    abstract public List<Passenger> retrievePassenger_BySource(List<Passenger>
passengerRecord,String source);
}

```

PASSENGER SKELETION-

```

import java.lang.reflect.Method;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.util.stream.Stream;

/**
 * @author TJ
 *
 * This class is used to verify if the Code Skeleton is intact and not modified by
participants thereby ensuring smooth auto evaluation
 */
public class SkeletonValidator {

    public SkeletonValidator() {

        validateClassName("PassengerCategorization");
        validateClassName("Passenger");
        validateClassName("InvalidCarrierException");
        validateClassName("PassengerUtility");

        validateMethodSignature(
            "retrievePassenger_BySource:java.util.List",
            "PassengerCategorization");
        validateMethodSignature(
            "fetchPassenger:java.util.List",
            "PassengerUtility");
        validateMethodSignature(
            "isValidCarrierName:boolean",
            "PassengerUtility");
        validateMethodSignature(
            "searchPassengerRecord:PassengerCategorization",
            "UserInterface");
    }

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

```



```

protected final boolean validateClassName(String className) {
    boolean incorrect = false;
    try {
        Class.forName(className);
        incorrect = true;
        LOG.info("Class Name " + className + " is correct");
    } catch (ClassNotFoundException e) {
        LOG.log(Level.SEVERE, "You have changed either the " + "class
name/package. Use the correct package "
+ "and class name as provided in the skeleton");
    } catch (Exception e) {
        LOG.log(Level.SEVERE,
"Please manually verify that the "
+ "Class name is same as skeleton before
uploading");
    }
    return incorrect;
}

protected final void validateMethodSignature(String methodWithExcpn, String
className) {
    Class cls = null;
    try {
        String[] actualmethods = methodWithExcpn.split(",");
        boolean errorFlag = false;
        String[] methodSignature;
        String methodName = null;
        String returnType = null;

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

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

```

```

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

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

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

```

PASSENGER USER INTERFACE-

```

import java.util.*;
import java.io.*;

public class UserInterface{
    public static PassengerCategorization searchPassengerRecord(){

        //FILL THE CODE HERE
        return (list,source)->{
            List<Passenger> result = new ArrayList<Passenger>();
            for(Passenger pass : list){

if((pass.getSource().toLowerCase()).equals(source.toLowerCase())){
                result.add(pass);
            }
        }
        return result;
    };
}

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

        PassengerCategorization pc = searchPassengerRecord();
        //FILL THE CODE HERE
        System.out.println("Invalid Carrier Records are:");
        PassengerUtility pu = new PassengerUtility();
    }
}

```

```

List<Passenger> list = null;
try{
list = pu.fetchPassenger(new String("PassengerRecord.txt"));
}
catch(FileNotFoundException e){
    e.printStackTrace();
}
catch(IOException e){
    e.printStackTrace();
}
catch(Exception e){
    e.printStackTrace();
}
System.out.println("Enter the source to search");
Scanner sc = new Scanner(System.in);
String inp = sc.next();

List<Passenger> result = pc.retrievePassenger_BySource(list,inp);
if(result.size()==0){
    System.out.println("No Passenger Record");
}
else{
    for(Passenger passenger: result){
        System.out.println(passenger.getPassengerName()+"
"+passenger.getPhoneNumber()+" "+passenger.getDateOfJourney()+" "+
passenger.getDestination());
    }
}
}
}

```

INVALID CARRIER EXEMPTION -

```

public class InvalidCarrierException extends Exception{
    //FILL THE CODE HERE
    public InvalidCarrierException (String message){
        super(message);
    }
}

```

EMPLOYEE SALARY

EMPLOYEE-

```

public class Employee {
3
4 // Fill the code
5 private String employeeName;
6 private int employeeId;
7 private int incrementPercentage;
8 private double salary;
9
10 public void setEmployeeId(int employeeId){
11 this.employeeId=employeeId;
12 }

```

```

13 public int getEmployeeId(){
14 return employeeId;
15 }
16 public void setEmployeeName(String employeeName){
17 this.employeeName=employeeName;
18 }
19 public String getEmployeeName(){
20 return employeeName;
21 }
22 public void setSalary(double salary){
23 this.salary=salary;
24 }
25 public double getSalary(){
26 return salary;
27 }
28 public void setIncrementPercentage(int incrementPercentage){
29 this.incrementPercentage=incrementPercentage;
30 }
31 public int getIncrementPercentage(){
32 return incrementPercentage;
33 }
34 public Employee(int employeeId,String employeeName,double salary){
35 this.employeeId=employeeId;
36 this.employeeName=employeeName;
37 this.salary=salary;
38 }
39 public void findIncrementPercentage(int yearsOfExperience){
40 //Calculate the incremented salary of the employee
41 if(yearsOfExperience>=1&&yearsOfExperience<=5){
42 incrementPercentage=15;
43 }
44 else if(yearsOfExperience>=6&&yearsOfExperience<=10){
45 incrementPercentage=30;
46 }
47 else if(yearsOfExperience>=11&&yearsOfExperience<=15){
48 incrementPercentage=45;
49 }
50 }
51 public double calculateIncrementSalary(){
52 double incrementedSalary=salary+((salary*(double)incrementPercentage)/100);
53 return incrementedSalary;
54 }

```

MAIN .JAVA-

```

1 import java.util.*;
2 public class Main {
3
4 public static void main(String[] args)
5 {
6 Scanner read=new Scanner(System.in);
7
8 //Fill the code
9 try
10 {
11 System.out.println("Enter the Employee Id");
12 int id=Integer.parseInt(read.nextLine());
13 System.out.println("Enter the Employee Name");

```

```

14 String name=read.nextLine();
15 System.out.println("Enter the salary");
16 double salary=Double.parseDouble(read.nextLine());
17 System.out.println("Enter the Number of Years in Experience");
18 int exp_year=Integer.parseInt(read.nextLine());
19 Employee e=new Employee(id,name,salary);
20 e.findIncrementPercentage(exp_year);
21
22 double incrementedSalary=e.calculateIncrementSalary();
23 System.out.printf("Incremented Salary %.2f", incrementedSalary);
24 }
25 catch(Exception e)
26 {
27 System.out.println(e);
28 }
29 }
30
31 }

```

HOME APPLIANCES

USER INTERFACE-

```

import java.util.*;
public class HomeAppliances {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter Product Id");
        String id = sc.nextLine();
        System.out.println("Enter Product Name");
        String name = sc.nextLine();
        switch (name)
        {
            case "AirConditioner":
            {
                System.out.println("Enter Batch Id");
                String batch = sc.next();
                System.out.println("Enter Dispatch
date");
                String date = sc.next();
                System.out.println("Enter Warranty
Years");
                int years = sc.nextInt();
                System.out.println("Enter type of Air
Conditioner");
                String type = sc.nextLine();
                System.out.println("Enter quantity");
                double capac = sc.nextDouble();
                AirConditioner ob1 = new
AirConditioner(id, name, batch, date, years, type,
capac);
                double price =
ob1.calculateProductPrice();
                System.out.printf("Price of the Product
is %.2f ", price);
            }
        }
    }
}

```

```

        case "LEDTV":
        {
            System.out.println("Enter Batch Id");
            String batch = sc.nextLine();
            System.out.println("Enter Dispatch
date");
            String date = sc.nextLine();
            System.out.println("Enter Warranty
Years");
            int years = sc.nextInt();
            System.out.println(name);
            System.out.println("Enter size in
inches");
            int size = sc.nextInt();
            System.out.println("Enter quality");
            String quality = sc.nextLine();
            LEDTV ob2 = new LEDTV(id, name, batch,
date, years, size, quality);
            ob2.calculateProductPrice();
            System.out.printf("Price of the Product
is %.2f ", price);
        }
        case "MicrowaveOven":
        {
            System.out.println("Enter Batch Id");
            String batch = sc.nextLine();
            System.out.println("Enter Dispatch
date");
            String date = sc.nextLine();
            System.out.println("Enter Warranty
Years");
            int years = sc.nextInt();
            System.out.println("Enter quantity");
            int quantity = sc.nextInt();
            System.out.println("Enter quality");
            String quality = sc.nextLine();
            MicrowaveOven ob3 = new
MicrowaveOven(id, name, batch, date, years,
quantity, quality);
            ob3.calculateProductPrice();
            System.out.printf("Price of the Product
is %.2f ", price);
        }
        default: {
            System.out.println("Provide a valid
Product name");
        }
    }
}

```

MICROWAVE.JAVA

```

public class MicrowaveOven extends ElectronicProducts {
    private int quantity;

```

```

        private String quality;
        public int getQuantity() {
            return quantity;
        }
        public void setQuantity(int quantity) {
            this.quantity = quantity;
        }
        public String getQuality() {
            return quality;
        }
        public void setQuality(String quality) {
            this.quality = quality;
        }
        // Include Constructor
        public MicrowaveOven(String productId, String productName, String
batchId, String
dispatchDate, int warrantyYears,
                                int quantity, String quality) {
            super(productId, productName, batchId, dispatchDate,
warrantyYears);
            this.quantity = quantity;
            this.quality = quality;
        }
        public double calculateProductPrice() {
            // Fill Code
            double price = 0;
            if (quality == "Low") {
                price = quantity * 1250;
            } else if (quality == "Medium") {
                price = quantity * 1750;
            } else if (quality == "High") {
                price = quantity * 2000;
            }
            return price;
        }
    }
}

```

ELECTRONIC PRODUCT.JAVA-

```

public class ElectronicProducts {
    protected String productId;
    protected String productname;
    protected String batchId;
    protected String dispatchDate;
    protected int warrantyYears;
    public String getProductId() {
        return productId;
    }
    public void setProductId(String productId) {
        this.productId = productId;
    }
    public String getProductname() {
        return productname;
    }
    public void setProductname(String productname) {
        this.productname = productname;
    }
    public String getBatchId() {

```

```

        return batchId;
    }
    public void setBatchId(String batchId) {
        this.batchId = batchId;
    }
    public String getDispatchDate() {
        return dispatchDate;
    }
    public void setDispatchDate(String dispatchDate) {
        this.dispatchDate = dispatchDate;
    }
    public int getWarrantyYears() {
        return warrantyYears;
    }
    public void setWarrantyYears(int warrantyYears) {
        this.warrantyYears = warrantyYears;
    }
    public ElectronicProducts(String productId, String productname,
String batchId, String
dispatchDate,
                                int warrantyYears) {
        this.productId = productId;
        this.productname = productname;
        this.batchId = batchId;
        this.dispatchDate = dispatchDate;
        this.warrantyYears = warrantyYears;
    }
}

```

AIR CONDITIONER .JAVA-

```

public class AirConditioner extends ElectronicProducts {
    private String airConditionerType;
    private double capacity;
    public String getAirConditionerType() {
        return airConditionerType;
    }
    public void setAirConditionerType(String airConditionerType) {
        this.airConditionerType = airConditionerType;
    }
    public double getCapacity() {
        return capacity;
    }
    public void setCapacity(double capacity) {
        this.capacity = capacity;
    }
    // Include Constructor
    public AirConditioner(String productId, String productName, String
batchId, String
dispatchDate, int warrantyYears,
                                String airConditionerType, double
capacity) {
        super(productId, productName, batchId, dispatchDate,
warrantyYears);
        this.airConditionerType = airConditionerType;
        this.capacity = capacity;
    }
    public double calculateProductPrice() {

```



```

// Fill Code
double cost = 0;
if (airConditionerType == "Residential") {
    if (capacity == 2.5) {
        cost = 32000;
    } else if (capacity == 4) {
        cost = 40000;
    } else if (capacity == 5.5) {
        cost = 47000;
    }
} else if (airConditionerType == "Commercial") {
    if (capacity == 2.5) {
        cost = 40000;
    } else if (capacity == 4) {
        cost = 55000;
    } else if (capacity == 5.5) {
        cost = 67000;
    }
} else if (airConditionerType == "Industrial") {
    if (capacity == 2.5) {
        cost = 47000;
    } else if (capacity == 4) {
        cost = 60000;
    } else if (capacity == 5.5) {
        cost = 70000;
    }
}
return cost;
}
}

```

LED TV.JAVA

```

public class LEDTV extends ElectronicProducts {
    private int size;
    private String quality;
    public int getSize() {
        return size;
    }
    public void setSize(int size) {
        this.size = size;
    }
    public String getQuality() {
        return quality;
    }
    public void setQuality(String quality) {
        this.quality = quality;
    }
    // Include Constructor
    public LEDTV(String productId, String productName, String batchId,
String dispatchDate, int
warrantyYears, int size,
String quality) {
        super(productId, productName, batchId, dispatchDate,
warrantyYears);
        this.size = size;
        this.quality = quality;
    }
}

```

```

        public double calculateProductPrice() {
            // Fill Code
            double price = 0;
            if (quality == "Low") {
                price = size * 850;
            } else if (quality == "Medium") {
                price = size * 1250;
            } else if (quality == "High") {
                price = size * 1550;
            }
            return price;
        }
    }
}

```

CINEMA

BOOK A MOVIE TICKET-

```

public class BookAMovieTicket {
    protected String ticketId;
    protected String customerName;
    protected long mobileNumber;
    protected String emailId;
    protected String movieName;
    public void setticketId( String ticketId){
        this.ticketId=ticketId;
    }
    public void setcustomerName( String customerName){
        this.customerName=customerName;
    }
    public void setmobileNumber( long mobileNumber){
        this.mobileNumber=mobileNumber;
    }
    public void setemailId( String emailId){
        this.emailId=emailId;
    }
    public void setmovieName( String movieName){
        this.movieName=movieName;
    }
    public String getticketId(){
        return ticketId;
    }
    public String getcustomerName(){
        return customerName;
    }
    public String getemailId(){
        return emailId;
    }
    public String getmovieName(){
        return movieName;
    }
    public long getmobileNumber(){
        return mobileNumber;
    }
    public BookAMovieTicket(String ticketId,String customerName,long
mobileNumber,String emailId,String movieName){

```

```

this.ticketId=ticketId;
this.customerName=customerName;
this.mobileNumber=mobileNumber;
this.emailId=emailId;
this.movieName=movieName;
}

}

```

PLATINUM TICKET-

```

public class PlatinumTicket extends BookAMovieTicket {
public PlatinumTicket(String ticketId, String customerName, long mobileNumber,
String emailId, String movieName) {
super(ticketId, customerName, mobileNumber, emailId, movieName);
}
public boolean validateTicketId(){
int count=0;
if(ticketId.contains("PLATINUM"));
count++;
char[] cha=ticketId.toCharArray();
for(int i=8;i<11;i++){
if(cha[i]>='1'&& cha[i]<='9')
count++;
}
if(count==4)
return true;
else
return false;
}
public double caculateTicketCost(int numberOfTickets,String ACFacility){
double amount;
if(ACFacility.equals("yes")){
amount=750*numberOfTickets;
}
else{
amount=600*numberOfTickets;
}
return amount;
}
}

```

GOLD TICKET-

```

public class GoldTicket extends BookAMovieTicket {
public GoldTicket(String ticketId, String customerName, long mobileNumber,
String emailId, String movieName) {
super(ticketId, customerName, mobileNumber, emailId, movieName);
}
public boolean validateTicketId(){
int count=0;
if(ticketId.contains("GOLD"));
count++;
char[] cha=ticketId.toCharArray();
for(int i=4;i<7;i++){
if(cha[i]>='1'&& cha[i]<='9')
count++;
}
}

```

```

if(count==4)
return true;
else
return false;
}
public double caculateTicketCost(int numberOfTickets,String ACFacility){
double amount;
if(ACFacility.equals("yes")){
amount=500*numberOfTickets;
}
else{
amount=350*numberOfTickets;
}
return amount;
}
}

```

SILVER TICKET-

```

public class SilverTicket extends BookAMovieTicket{
public SilverTicket(String ticketId, String customerName, long mobileNumber,
String emailId, String movieName) {
super(ticketId, customerName, mobileNumber, emailId, movieName);
}
public boolean validateTicketId(){
int count=0;
if(ticketId.contains("SILVER"));
count++;
char[] cha=ticketId.toCharArray();
for(int i=6;i<9;i++){
if(cha[i]>='1'&& cha[i]<='9')
count++;
}
if(count==4)
return true;
else
return false;
}
public double caculateTicketCost(int numberOfTickets,String ACFacility){
double amount;
if(ACFacility.equals("yes")){
amount=250*numberOfTickets;
}
else{
amount=100*numberOfTickets;
}
return amount;
}
}

```

USER INTERFACE -

```

import java.util.*;
public class UserInterface {
public static void main(String[] args) {
Scanner sc=new Scanner(System.in);
System.out.println("Enter Ticket Id");
}
}

```

```

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.caculateTicketCost(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.caculateTicketCost(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.caculateTicketCost(tno, choice);
System.out.println("Ticket cost is "+cost);
}
else if(b3==false){
System.out.println("Provide valid Ticket Id");
System.exit(0);
}
}
}
}
}
}

```

Reverse A word

helloworld:

```
import java .util. Scanner;

public class HelloWorld {

    public static void main (String[]arugs){

        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.equals(b)&& b.equals(c)) {

                StringBuilder input1 = new StringBuilder ();

                input1.append(words[words.length-1]);

                input1 =input1.reverse();

                input1.append(words[0]);

                System.out.println(input1);

            }

            else {

                StringBuilder input1 = new StringBuilder();

                input1.append(words[0]);

                input1.reverse();

                input1.append(words[words.length-1]);
```

```
        System.out.println(input1); }  
    }  
}
```

AirConditioner:

```

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;
} }

```

ElectronicProducts:

```

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;
} }

```

LEDTV:

```

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;

} }

```

MicrowaveOven:

```

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;
    } }

```

UserInterface:

```

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);  
}  
}  
}
```

Slogan

Main:

```
import java.util.Scanner;

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

        String slogan=sc.nextLine();
        char[] ch=slogan.toCharArray();
        for(int i=0;i<slogan.length();i++){
            if(ch[i]>='a' && ch[i]<='z' || ch[i]>='A' && ch[i]<='Z'){
            }
            else if(slogan.charAt(i)==' '){
            }
            else{
                System.out.println("Invalid slogan");
                return;
            }
        }

        int count[] = new int[256];
        String str = slogan.replaceAll("\\s", "");
        int len = str.length();
        int sum=0;
        int mul=0;
        for (int i = 0; i < len; i++) {
            char c = str.charAt(i);
            count[c]++;
        }
        for (int i = 0; i < len; i++) {
            char chh = str.charAt(i);
            if (count[chh] == 1) {
```

```
        sum++;  
    }    else {  
        mul++; }  
}    if(sum==mul){  
    System.out.println("All the guidelines are satisfied for "+slogan);  
}    else{  
    System.out.println(slogan+" does not satisfy the guideline");  
}  
}  
}
```


Group -1

1. AirVoice - Registration

Grade settings: Maximum grade: 100

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 }
45
```

Main.java

```
1 import java.util.Scanner;
2
3 public class Main {
4
```

```

5      public static void main(String[] args) {
6          // TODO Auto-generated method stub
7      Scanner sc=new Scanner(System.in);
8      Customer c=new Customer();
9      System.out.println("Enter the Name:");
10     String name=(sc.nextLine());
11     System.out.println("Enter the ContactNumber:");
12     long no=sc.nextLong();
13     sc.nextLine();
14     System.out.println("Enter the EmailId:");
15     String mail=sc.nextLine();
16
17     System.out.println("Enter the Age:");
18     int age=sc.nextInt();
19     c.setCustomerName(name);
20     c.setContactNumber(no);
21     c.setEmailId(mail);
22     c.setAge(age);
23     System.out.println("Name:"+c.getCustomerName());
24     System.out.println("ContactNumber:"+c.getContactNumber());
25     System.out.println("EmailId:"+c.getEmailId());
26     System.out.println("Age:"+c.getAge());
27
28
29
30     }
31
32 }

```

Grade

Reviewed on Monday, 7 February 2022, 4:45 PM by Automatic grade

Grade 100 / 100

Assessment report

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

=====

2. Payment - Inheritance

Grade settings: Maximum grade: 100

Run: Yes **Evaluate:** Yes

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

Payment Status

Roy is a wholesale cloth dealer who sells cloth material to the local tailors on monthly installments. At the end of each month, he collects the installment amount from all his customers.

Some of his customers pay by Cheque, some pay by Cash and some by Credit Card. He wants to automate this payment process.

Help him to do this by writing a java program.

Requirement 1: Make Payment

The application needs to verify the payment process and display the status report of payment by getting the inputs like due amount, payment mode and data specific to the payment mode from the user and calculate the balance amount.

Component Specification: Payment Class (Parent Class)

Component Name	Type(Class)	Attributes	Methods	Responsibilities
Make payment for EMI amount	Payment	int dueAmount	Include a public getter and setter method	
Make payment for EMI amount	Payment		public boolean payAmount()	The boolean payAmount() method should return true if there is no due to be paid, else return false.

Note:

- The attributes of Payment class should be private.
- The payment can be of three types: Cheque, Cash, Credit Card.

Component Specification: Cheque class (Needs to be a child of Payment class)

Component Name	Type(Class)	Attributes	Methods	Responsibilities
	Cheque	String chequeNo int chequeAmount Date dateOfIssue	Include a public getter and setter method for all the attributes.	

Make payment for EMI amount	Cheque		public boolean payAmount()	This is an overridden method of the parent class. It should return true if the cheque is valid and the amount is valid. Else return false.
-----------------------------	--------	--	----------------------------	--

Note:

- The cheque is valid for 6 months from the date of issue.
- Assume the current date is 01-01-2020 in dd-MM-yyyy format.
- The chequeAmount is valid if it is greater than or equal to the dueAmount.

Component Specification: Cash class (Needs to be a child of Payment class)

Component Name	Type(Class)	Attributes	Methods	Responsibilities
Make payment for EMI amount	Cash	int cashAmount	Include a public getter and setter method for the attribute.	
Make payment for EMI amount	Cash		public boolean payAmount()	This is an overridden method of the parent class. It should return true if the cashAmount is greater than or equal to the dueAmount. Else return false.

Component Specification: Credit class (Needs to be a child of Payment class)

Component Name	Type (Class)	Attributes	Methods	Responsibilities
Make payment for EMI amount	Credit	int creditCardNo String cardType int creditCardAmount	Include a public getter and setter method for all the attributes.	

Make payment for EMI amount	Credit		public boolean payAmount()	This is an overridden method of the parent class. It should deduct the dueAmount and service tax from the creditCardAmount and return true if the credit card payment was done successfully. Else return false.
-----------------------------	--------	--	----------------------------	---

Note:

- The payment can be done if the credit card amount is greater than or equal to the sum of due amount and service tax. Else payment cannot be made.
- The cardType can be “silver” or “gold” or “platinum”. Set the creditCardAmount based on the cardType.
- Also service tax is calculated on dueAmount based on cardType.

Credit Card Type	Credit Card Amount	Service Tax
silver	10000	2% of the due amount
gold	50000	5% of the due amount
platinum	100000	10% of the due amount

- The boolean payAmount() method should deduct the due amount and the service tax amount from a credit card. If the creditCardAmount is less than the dueAmount+serviceTax, then the payment cannot be made.
- The balance in credit card amount after a successful payment should be updated in the creditCardAmount by deducting the sum of dueAmount and serviceTax from creditCardAmount itself.

Component Specification: Bill class

Component Name	Type(Class)	Attributes	Methods	Responsibilities
Payment Status Report	Bill		public String processPayment (Payment obj)	This method should return a message based on the status of the payment made.

Note:

- If the payment is successful, processPayment method should return a message “Payment done successfully via cash” or “Payment done successfully via cheque” or “Payment done successfully via creditcard. Remaining amount in your <<cardType>> card is <<balance in CreditCardAmount>>”
- If the payment is a failure, then return a message “Payment not done and your due amount is <<dueAmount>>”

Create a **public class Main** with the main method to test the application.

Note:

- Assume the current date as 01-01-2020 in dd-MM-yyyy format.
- 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.
- Adhere to the sample input and output.

Sample Input 1:

Enter the due amount:

3000

Enter the mode of payment(cheque/cash/credit):

cash

Enter the cash amount:

2000

Sample Output 1:

Payment not done and your due amount is 3000

Sample Input 2:

Enter the due amount:

3000

Enter the mode of payment(cheque/cash/credit):

cash

Enter the cash amount:

3000

Sample Output 2:

Payment done successfully via cash

Sample Input 3:

Enter the due amount:

3000

Enter the mode of payment(ch cheque/cash/credit):

cheque

Enter the cheque number:

123

Enter the cheque amount:

3000

Enter the date of issue:

21-08-2019

Sample Output 3:

Payment done successfully via cheque

Sample Input 4:

Enter the due amount:

3000

Enter the mode of payment(ch cheque/cash/credit):

credit

Enter the credit card number:

234

Enter the card type(silver,gold,platinum):

silver

Sample Output 4:

Payment done successfully via credit card. Remaining amount in your silver card is 6940

Automatic evaluation[+]

Main.java

```
1 import java.text.ParseException;
2 import java.text.SimpleDateFormat;
3 import java.util.Date;
4 import java.util.Scanner;
5 public class Main {
6
7     public static void main(String[] args) {
8
9         Scanner sc=new Scanner(System.in);
10        System.out.println("Enter the due amount:");
11        int dueAmount=sc.nextInt();
12
13        System.out.println("Enter the mode of payment(cheque/cash/credit:");
14        String mode=sc.next();
15        Bill b = new Bill();
16        if(mode.equals("cheque"))
17        {
18            System.out.println("enter the cheque number:");
19            String chequeNumber=sc.next();
20            System.out.println("enter the cheque amount:");
21            int chequeAmount=sc.nextInt();
22            System.out.println("enter the date of issue:");
23            String date=sc.next();
24            SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MM-yyyy");
25            Date dateOfIssue=null;
26            try
27            {
28                dateOfIssue = dateFormat.parse(date);
29            }
30            catch (ParseException e)
31            {
32
33            }
34            Cheque cheque= new Cheque();
35            cheque.setChequeNo(chequeNumber);
36            cheque.setChequeAmount(chequeAmount);
37            cheque.setDateOfIssue(dateOfIssue);
38            cheque.setDueAmount(dueAmount);
39            System.out.println(b.processPayment(cheque));
40        }
41        else if(mode.equals("cash"))
42        {
43            System.out.println("enter the cash amount:");
44            int CashAmount=sc.nextInt();
45            Cash cash=new Cash();
46            cash.setCashAmount(CashAmount);
47            cash.setDueAmount(dueAmount);
48            System.out.println(b.processPayment(cash));
49        }
50        else if(mode.equals("credit"))
51        {
52            System.out.println("enter the credit card number:");
53            int creditCardNumber=sc.nextInt();
54            System.out.println("enter the card type:");
55            String cardType=sc.next();
56
```

```

57         Credit credit=new Credit();
58         credit.setCreditCardNo(creditCardNumber);
59         credit.setCardType(cardType);
60         credit.setDueAmount(dueAmount);
61         System.out.println(b.processPayment(credit));
62     }
63 }
64 }

```

Payment.java

```

1 public class Payment {
2     private int dueAmount;
3
4     public boolean payAmount()
5     {
6         if(dueAmount == 0)
7             return true;
8         else
9             return false;
10    }
11
12    public int getDueAmount() {
13        return dueAmount;
14    }
15
16    public void setDueAmount(int dueAmount) {
17        this.dueAmount = dueAmount;
18    }
19 }

```

Cheque.java

```

1 import java.text.ParseException;
2 import java.text.SimpleDateFormat;
3 import java.util.Date;
4 public class Cheque extends Payment {
5     String chequeNo;
6     int chequeAmount;
7     Date dateOfIssue;
8     public String getChequeNo() {
9         return chequeNo;
10    }
11    public void setChequeNo(String chequeNo) {
12        this.chequeNo = chequeNo;
13    }
14    public int getChequeAmount() {
15        return chequeAmount;
16    }
17    public void setChequeAmount(int chequeAmount) {
18        this.chequeAmount = chequeAmount;
19    }
20    public Date getDateOfIssue() {
21        return dateOfIssue;
22    }
23    public void setDateOfIssue(Date dateOfIssue) {
24        this.dateOfIssue = dateOfIssue;
25    }
26
27    @Override
28    public boolean payAmount()
29    {
30        SimpleDateFormat format = new SimpleDateFormat("dd-MM-yyyy");
31        Date today = new Date();
32        try
33        {
34            today = format.parse("01-01-2020");

```

```

35         }
36         catch (ParseException e)
37         {
38             return false;
39         }
40         long diff = today.getTime()-dateOfIssue.getTime();
41         int day = (int) Math.abs(diff/(1000*60*60*24));
42         int month = day/30;
43         if(month <=6)
44         {
45
46             if(chequeAmount>=getDueAmount())
47             {
48                 return true;
49             }
50             else
51                 return false;
52
53         }
54         else
55             return false;
56     }
57
58 }
59 }

```

Cash.java

```

1 public class Cash extends Payment {
2     int cashAmount;
3
4     public int getCashAmount() {
5         return cashAmount;
6     }
7
8     public void setCashAmount(int cashAmount) {
9         this.cashAmount = cashAmount;
10    }
11
12    @Override
13    public boolean payAmount()
14    {
15        if(cashAmount>=getDueAmount())
16            return true;
17        else
18            return false;
19    }
20
21
22 }

```

Credit.java

```

1 public class Credit extends Payment {
2     int creditCardNo;
3     String cardType;
4     int creditCardAmount;
5     public int getCreditCardNo() {
6         return creditCardNo;
7     }
8     public void setCreditCardNo(int creditCardNo) {
9         this.creditCardNo = creditCardNo;
10    }
11    public String getCardType() {
12        return cardType;
13    }
14    public void setCardType(String cardType) {

```

```

15         this.cardType = cardType;
16     }
17     public int getCreditCardAmount() {
18         return creditCardAmount;
19     }
20     public void setCreditCardAmount(int creditCardAmount) {
21         this.creditCardAmount = creditCardAmount;
22     }
23
24
25     @Override
26     public boolean payAmount()
27     {
28         int netAmount = 0;
29         if(cardType.equals("silver"))
30         {
31             netAmount = (int) (getDueAmount()*1.02);
32             creditCardAmount = 10000;
33         }
34         else if(cardType.equals("gold"))
35         {
36             netAmount = (int) (getDueAmount()*1.05);
37             creditCardAmount = 50000;
38         }
39         else if(cardType.equals("platinum"))
40         {
41             netAmount = (int) (int) (getDueAmount()*1.1);
42             creditCardAmount = 100000;
43         }
44
45         if(creditCardAmount>=netAmount)
46         {
47             creditCardAmount = creditCardAmount - netAmount;
48             return true;
49         }
50         else
51             return false;
52     }
53
54
55 }

```

Bill.java

```

1 public class Bill {
2     public String processPayment(Payment obj)
3     {
4         String res="";
5         if(obj instanceof Cheque)
6         {
7             if(obj.payAmount())
8                 res = "Payment done successfully via cheque";
9             else
10                 res = "Payment not done and your due amount is
"+obj.getDueAmount();
11         }
12         else if(obj instanceof Cash)
13         {
14             if(obj.payAmount())
15                 res = "Payment done successfully via cash";
16             else
17                 res = "Payment not done and your due amount is
"+obj.getDueAmount();
18         }
19         else if(obj instanceof Credit)
20         {
21             Credit c = (Credit) obj;

```

```

22                                     if(obj.payAmount())
23                                     res = "Payment done successfully via credit card. Remaining
amount in your "+c.getCardType()+" card is "+c.getCreditCardAmount();
24                                     else
25                                     res = "Payment not done and your due amount is
"+obj.getDueAmount();
26                                     }
27                                     return res;
28                                     }
29 }

```

Grade

Reviewed on Wednesday, 1 December 2021, 10:08 PM by Automatic grade

Grade 100 / 100

Assessment report

TEST CASE PASSED

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

3.Power Progress

Grade settings: Maximum grade: 100

Run: Yes **Evaluate:** Yes

Automatic grade: Yes

Andrews taught exponential multiplication to his daughter and gave her two inputs.

Assume, the first input as M and the second input as N. He asked her to find the sequential power of M until N times. For Instance, consider M as 3 and N as 5. Therefore, 5 times the power is incremented gradually from 1 to 5 such that, $3^1=3$, $3^2=9$, $3^3=27$, $3^4=81$, $3^5=243$. The input numbers should be greater than zero Else print "<Input> is an invalid". The first Input must be less than the second Input, Else print "<first input> is not less than <second input>".

Write a Java program to implement this process programmatically and display the output in sequential order. (3^3 means $3*3*3$).

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.

Adhere to the code template, if provided.

Kindly do not use System.exit() in the code.

Sample Input 1:

3

5

Sample Output 1:

3 9 27 81 243

Explanation: Assume the first input as 3 and second input as 5. The output is to be displayed are based on the sequential power incrementation. i.e., $3(3)$ $9(3*3)$ $27(3*3*3)$ $81(3*3*3*3)$ $243(3*3*3*3*3)$

Sample Input 2:

-3

Sample Output 2:

-3 is an invalid

Sample Input 3:

3

0

Sample Output 3:

0 is an invalid

Sample Input 4:

4

2

Sample Output 4:

4 is not less than 2

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

Main.java

```
1 import java.util.*;
2 public class Main
3 {
4     public static void main(String[] args)
5     {
6         Scanner sc=new Scanner(System.in);
7         //Fill the code
8         int m=sc.nextInt();
9         if(m<=0){
10             System.out.println(""+m+" is an invalid");
11             return;
12         }
13         int n=sc.nextInt();
14         if(n<=0){
15             System.out.println(""+n+" is an invalid");
16             return;
17         }
18         if(m>=n){
19             System.out.println(""+m+" is not less than "+n);
20             return;
21         }
22         for(int i=1;i<=n;i++){
```

```

23      System.out.print((int)Math.pow(m,i)+"");
24      }
25  }
26 }

```

Grade

Reviewed on Monday, 7 February 2022, 4:46 PM by Automatic grade

Grade 100 / 100

Assessment report

TEST CASE PASSED

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

4. ZeeZee bank

Grade settings: Maximum grade: 100

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)	<p>This method takes the amount to be deposited as an argument</p> <p>This method should perform the deposit, by adding the deposited amount to the balanceAmount</p>

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)	<p>This method should take the amount to be withdrawn as an argument.</p> <p>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.</p>

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.text.DecimalFormat;
2 import java.util.Scanner;
3 import java.util.Scanner;
4
5
6 public class Main{
7     static Account ac=new Account(0, 0);
8     public static void main (String[] args) {
9         Scanner sc=new Scanner(System.in);
10        System.out.println("Enter the account number:");
11        ac.setAccountNumber(sc.nextLong());
12        System.out.println("Enter the available amount in the account:");
13        ac.setBalanceAmount(sc.nextDouble());
14        System.out.println("Enter the amount to be deposited:");
15        ac.deposit(sc.nextDouble());
16        System.out.printf("Available balance is:%.2f",ac.getBalanceAmount());
17        System.out.println();
18        System.out.println("Enter the amount to be withdrawn:");
19        ac.withdraw(sc.nextDouble());
20        System.out.printf("Available balance is:%.2f",ac.getBalanceAmount());
21        //Fill the code
22    }
23 }
24
25
26
```

Account.java

```
1
2 public class Account {
3     long accountNumber;
4     double balanceAmount;
5
6
7     public Account(long accno, double bal){
```

```

8    super();
9    this.accountNumber=accno;
10   this.balanceAmount=bal;
11 }
12 public long getAccountNumber(){
13     return accountNumber;
14 }
15 public void setAccountNumber(long accno){
16     this.accountNumber=accno;
17 }
18 public double getBalanceAmount(){
19     return balanceAmount;
20 }
21 public void setBalanceAmount(double bal) {
22     this.balanceAmount=bal;
23 }
24 public void deposit(double depositAmt){
25     float total=(float)(balanceAmount+depositAmt);
26     balanceAmount=total;
27 }
28 public boolean withdraw(double withdrawAmt){
29     float total;
30     if(withdrawAmt>balanceAmount){
31         System.out.println("Insufficient balance");
32
33         return false;
34     }else{
35         total=(float)(balanceAmount-withdrawAmt);
36         setBalanceAmount(total);
37         return true;
38     }
39 }
40 }

```

Grade

Reviewed on Monday, 7 February 2022, 4:47 PM by Automatic grade

Grade 100 / 100

Assessment report

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

5. Reverse a word

Grade settings: Maximum grade: 100

Run: Yes **Evaluate:** Yes

Automatic grade: Yes

Reverse a word

Rita and Brigitha want to play a game. That game is to check the first letter of each word in a given sentence (Case Insensitive). If it is equal, then reverse the last word and concatenate the first word. Else reverse the first word and concatenate the last word. Create a Java application and help them to play the game

Note:

- Sentence must contain at least 3 words else print "Invalid Sentence" and terminate the program
- Each word must contain alphabet only else print "Invalid Word" and terminate the program
- Check the first letter of each word in a given sentence (Case Insensitive). If it is equal, then reverse the last word and concatenate the first word and print. Else reverse the first word and concatenate the last word and print.
- Print the output without any space.

Please do not use `System.exit(0)` to terminate the program

Sample Input 1:

Sea sells seashells

Sample Output 1:

sllehsaesSea

Sample Input 2:

Sam is away from Australia for a couple of days

Sample Output 2:

maSdays

Sample Input 3:

Welcome home

Sample Output 3:

Invalid Sentence

Sample Input 4:

Friendly fire fighting fr@gs.

Sample Output 4:

Invalid Word

Automatic evaluation[+]

Main.java

```
1 import java.util.Scanner;
2 import java.lang.String.*;
3 import java.util.*;
4 public class Main{
5     public static void main(String[] args){
6         String[] words;
7         Scanner read =new Scanner(System.in);
8         String sentence=read.nextLine();
9         words=sentence.split(" ");
10        if(words.length<3)
11            System.out.println("Invalid Sentence");
12        else{
13            String a=words[0].substring(0,1);
14            String b=words[1].substring(0,1);
15            String c=words[2].substring(0,1);
16            if(a.equalsIgnoreCase(b)&&b.equalsIgnoreCase(c))
17            {
18                StringBuilder k= new StringBuilder();
19                k.append(words[words.length-1]);
20                k=k.reverse();
21                k.append(words[0]);
22                System.out.println(k);
23            }
24            else{
25                StringBuilder k = new StringBuilder();
26                k.append(words[0]);
27                k=k.reverse();
28                k.append(words[words.length-1]);
29                System.out.println(k);
30            }
31        }
32    }
33 }
```

Grade

Reviewed on Monday, 7 February 2022, 5:12 PM by Automatic grade

Grade 90 / 100

Assessment report

Fail 1 -- test5_CheckForTheSentenceContainsOtherThanAlphabets::
\$Expected output:"[Invalid Word]" Actual output:"[tahwme]"\$
[\[+\]Grading and Feedback](#)

6. Dominion cinemas

Grade settings: Maximum grade: 100

Run: Yes **Evaluate:** Yes

Automatic grade: Yes

Dominion cinema is a famous theatre in the city. It has different types of seat tiers – Platinum, Gold and Silver. So far the management was manually calculating the ticket cost for all their customers which proved very hectic and time consuming. Going forward they want to calculate ticket cost using their main computer. Assist them in calculating and retrieving the amount to be paid by the Customer.

Requirements 1: Calculation of Ticket Cost

The application needs to calculate the ticket cost to be paid by the Customer according to the seat tier.

Component Specification: BookAMovieTicket Class (Parent Class)

Component Name	Type(Class)	Attributes	Methods	Responsibilities
Calculation of Ticket cost	BookAMovieTicket	String ticketId String customerName long mobileNumber String emailId String movieName	Public getter and setter method for all the attributes and 5 argument constructor in the given order - ticketId, customerName, mobileNumber, emailId, movieName are provided as a part of the code skeleton.	

Note:

- The attributes of the BookAMovieTicket class should be protected.

Component Specification: GoldTicket class (Needs to be a child of BookAMovieTicket class)

Component Name	Type(Class)	Attributes	Methods	Responsibilities
Calculation of Ticket cost	GoldTicket		Include a public 5 argument constructor in the given order - ticketId, customerName, mobileNumber, emailId, movieName.	
Validate Ticket Id	GoldTicket		public boolean validateTicketId ()	This method should validate the Ticket Id, Ticket Id should contain a string “GOLD” followed by 3 digits. If the ticket id is valid this method should return true else it should return false.
Calculation of Ticket cost	GoldTicket		public double calculateTicketCost (int numberOfTickets, String ACFacility)	This method should calculate the ticket cost according to the seat tier and return the same.

Component Specification: PlatinumTicket class (Needs to be a child of the BookAMovieTicket class)

Component Name	Type(Class)	Attributes	Methods	Responsibilities
Calculation of Ticket cost	PlatinumTicket		Include a public 5 argument constructor in the given order - ticketId, customerName, mobileNumber, emailId, movieName.	
Validate Ticket Id	PlatinumTicket		public boolean validateTicketId()	This method should validate the Ticket Id, Ticket Id should contain a string “PLATINUM” followed by 3 digits. If the ticket id is

				valid this method should return true else it should return false.
Calculation of Ticket cost	PlatinumTicket		calculateTicketCost(int numberOfTickets, String ACFacility)	This method should calculate the ticket cost according to the seat tier and return the same.

Component Specification: SilverTicket class (Needs to be a child of the BookAMovieTicket class)

Component Name	Type(Class)	Attributes	Methods	Responsibilities
Calculation of Ticket cost	SilverTicket		Include a public 5 argument constructor in the given order - ticketId, customerName, mobileNumber, emailId, movieName.	
Validate Ticket Id	SilverTicket		public boolean validateTicketId()	This method should validate the Ticket Id, Ticket Id should contain a string "SILVER" followed by 3 digits. If the ticket id is valid this method should return true else it should return false.
Calculation of Ticket cost	SilverTicket		calculateTicketCost(int numberOfTickets, String ACFacility)	This method should calculate the ticket cost according to the seat tier and return the same.

Note:

- The classes GoldTicket, PlatinumTicket and SilverTicket should be concrete classes.

Ticket cost according to the seat tier without AC facilities.

Seat Tier	Silver	Gold	Platinum
-----------	--------	------	----------

Without AC Facility	100	350	600
With AC Facility	250	500	750

Amount is calculated based on the seat tier,

Amount = ticketCost * numberOfTickets

Use a **public class UserInterface** with the main method to test the application. In the main method call the validateTicketId() method, if the method returns true display the amount else display "**Provide valid Ticket Id**".

Note:

- **Display the amount to be paid to 2 decimal places.**
- **Use the System.out.printf method.**
- 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 Ticket Id

SILVER490

Enter Customer name

Venkat

Enter Mobile number

9012894578

Enter Email Id

venkat@gmail.com

Enter Movie name

Avengers

Enter number of tickets

8

Do you want AC or not

yes // Case insensitive

Ticket cost is 2000.00

Sample Input 2:

Enter Ticket Id

ACN450

Enter Customer name

Kamal

Enter Mobile number

9078561093

Enter Email Id

kamal@gmail.com

Enter Movie name

Tangled

Enter number of tickets

9

Provide valid Ticket Id

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

BookAMovieTicket.java

```
1
2 public class BookAMovieTicket {
3
4     protected String ticketId;
5     protected String customerName;
6     protected long mobileNumber;
7     protected String emailId;
8     protected String movieName;
9
10    public String getTicketId() {
11        return ticketId;
```

```

12     }
13     public void setTicketId(String ticketId) {
14         this.ticketId = ticketId;
15     }
16     public String getCustomerName() {
17         return customerName;
18     }
19     public void setCustomerName(String customerName) {
20         this.customerName = customerName;
21     }
22     public long getMobileNumber() {
23         return mobileNumber;
24     }
25     public void setMobileNumber(long mobileNumber) {
26         this.mobileNumber = mobileNumber;
27     }
28     public String getEmailId() {
29         return emailId;
30     }
31     public void setEmailId(String emailId) {
32         this.emailId = emailId;
33     }
34     public String getMovieName() {
35         return movieName;
36     }
37     public void setMovieName(String movieName) {
38         this.movieName = movieName;
39     }
40
41     public BookAMovieTicket(String ticketId, String customerName, long mobileNumber, String emailId,
String movieName) {
42         this.ticketId = ticketId;
43         this.customerName = customerName;
44         this.mobileNumber = mobileNumber;
45         this.emailId = emailId;
46         this.movieName = movieName;
47     }
48 }
49
50
51
52 }
53

```

GoldTicket.java

```

1
2 public class GoldTicket extends BookAMovieTicket{
3     public GoldTicket(String ticketId,String customerName, long mobileNumber,
4     String emailId, String movieName){
5         super(ticketId, customerName, mobileNumber, emailId, movieName);
6     }
7
8     public boolean validateTicketId(){
9         int count=0;
10        if(ticketId.contains("GOLD"));
11        count++;

```

```

12         char[] cha=ticketId.toCharArray();
13         for(int i=4;i<7;i++){
14             if(cha[i]>='1'&& cha[i]<='9')
15                 count++;
16         }
17         if(count==4)
18             return true;
19         else
20             return false;
21     }
22
23
24     // Include Constructor
25
26     public double calculateTicketCost(int numberOfTickets, String ACFacility){
27         double amount;
28         if(ACFacility.equals("yes")){
29             amount=500*numberOfTickets;
30         }
31         else{
32             amount=350*numberOfTickets;
33         }
34
35         return amount;
36     }
37
38 }

```

PlatinumTicket.java

```

1 public class PlatinumTicket extends BookAMovieTicket{
2     public PlatinumTicket(String ticketId, String customerName, long mobileNumber,
3     String emailId, String movieName){
4         super(ticketId, customerName, mobileNumber, emailId, movieName);
5     }
6
7     public boolean validateTicketId(){
8         int count=0;
9         if(ticketId.contains("PLATINUM"));
10            count++;
11            char[] cha=ticketId.toCharArray();
12            for(int i=8;i<11;i++){
13                if(cha[i]>='1'&& cha[i]<='9')
14                    count++;
15            }
16            if(count==4)
17                return true;
18            else
19                return false;
20        }
21
22        // Include Constructor
23
24        public double calculateTicketCost(int numberOfTickets, String ACFacility){
25            double amount;
26            if(ACFacility.equalsIgnoreCase("yes")){
27                amount=750*numberOfTickets;

```

```

28         }
29         else{
30             amount=600*numberOfTickets;
31         }
32
33         return amount;
34     }
35
36 }
37

```

SilverTicket.java

```

1
2 public class SilverTicket extends BookAMovieTicket{
3     public SilverTicket(String ticketId, String customerName, long mobileNumber,
4     String emailId, String movieName){
5         super(ticketId, customerName, mobileNumber, emailId, movieName);
6     }
7
8     public boolean validateTicketId(){
9         int count=0;
10        if(ticketId.contains("SILVER"));
11        count++;
12        char[] cha=ticketId.toCharArray();
13        for(int i=6;i<9;i++){
14            if(cha[i]>='1'&& cha[i]<='9')
15                count++;
16        }
17        if(count==4)
18            return true;
19        else
20            return false;
21    }
22
23    // Include Constructor
24
25    public double calculateTicketCost(int numberOfTickets, String ACFacility){
26        double amount;
27        if(ACFacility.equals("yes")){
28            amount=250*numberOfTickets;
29        }
30        else{
31            amount=100*numberOfTickets;
32        }
33
34        return amount;
35    }
36
37 }
38

```

UserInterface.java

```

1 import java.util.*;
2
3 public class UserInterface {

```

```

4
5     public static void main(String[] args){
6         Scanner sc=new Scanner(System.in);
7         System.out.println("Enter Ticket Id");
8         String tid=sc.next();
9         System.out.println("Enter Customer name");
10        String cnm=sc.next();
11        System.out.println("Enter Mobile number");
12        long mno=sc.nextLong();
13        System.out.println("Enter Email id");
14        String email=sc.next();
15        System.out.println("Enter Movie name");
16        String mnm=sc.next();
17        System.out.println("Enter number of tickets");
18        int tno=sc.nextInt();
19        System.out.println("Do you want AC or not");
20        String choice =sc.next();
21        if(tid.contains("PLATINUM")){
22            PlatinumTicket PT= new PlatinumTicket(tid,cnm,mno,email,mnm);
23            boolean b1=PT.validateTicketId();
24            if(b1==true){
25                double cost=PT.calculateTicketCost(tno, choice);
26                System.out.println("Ticket cost is "+String.format("%.2f",cost));
27            }
28            else if(b1==false){
29                System.out.println("Provide valid Ticket Id");
30                System.exit(0);
31            }
32        }
33        else if(tid.contains("GOLD")){
34            GoldTicket GT= new GoldTicket(tid,cnm,mno,email,mnm);
35            boolean b2=GT.validateTicketId();
36            if(b2==true){
37                double cost=GT.calculateTicketCost(tno,choice);
38                System.out.println("Ticket cost is "+String.format("%.2f",cost));
39            }
40            else if (b2==false){
41                System.out.println("Provide valid Ticket Id");
42                System.exit(0);
43            }
44        }
45        else if(tid.contains("SILVER")){
46            SilverTicket ST= new SilverTicket(tid,cnm,mno,email,mnm);
47            boolean b3=ST.validateTicketId();
48            if(b3==true){
49                double cost=ST.calculateTicketCost(tno,choice);
50                System.out.println("Ticket cost is "+String.format("%.2f",cost));
51            }
52            else if (b3==false){
53                System.out.println("Provide valid Ticket Id");
54                System.exit(0);
55            }
56        }
57    }
58 }
59
60

```

Grade

Reviewed on Monday, 7 February 2022, 4:18 PM by Automatic grade

Grade 100 / 100

Assessment report

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

=====

Group-2

1. Flight record retrieval

Grade settings: Maximum grade: 100

Based on: [JAVA CC JDBC - MetaData V1 - ORACLE \(w/o Proj Struc\)](#)

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)	Attributes	Methods	Responsibilities
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	FlightManagementSystem		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	integer
source	varchar2(30)
destination	varchar2(30)
noofseats	integer
flightfare	double

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
2 public class Flight {
```

```

3
4     private int flightId;
5     private String source;
6     private String destination;
7     private int noOfSeats;
8     private double flightFare;
9     public int getFlightId() {
10         return flightId;
11     }
12     public void setFlightId(int flightId) {
13         this.flightId = flightId;
14     }
15     public String getSource() {
16         return source;
17     }
18     public void setSource(String source) {
19         this.source = source;
20     }
21     public String getDestination() {
22         return destination;
23     }
24     public void setDestination(String destination) {
25         this.destination = destination;
26     }
27     public int getNoOfSeats() {
28         return noOfSeats;
29     }
30     public void setNoOfSeats(int noOfSeats) {
31         this.noOfSeats = noOfSeats;
32     }
33     public double getFlightFare() {
34         return flightFare;
35     }
36     public void setFlightFare(double flightFare) {
37         this.flightFare = flightFare;
38     }
39     public Flight(int flightId, String source, String destination,
40                 int noOfSeats, double flightFare) {
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.util.ArrayList;
2 import java.sql.*;
3
4
5 public class FlightManagementSystem {
6
7     public ArrayList<Flight> viewFlightBySourceDestination(String source, String destination){
8         ArrayList<Flight> flightList = new ArrayList<Flight>();
9         try{
10             Connection con = DB.getConnection();
11
12             String query="SELECT * FROM flight WHERE source= " + source + " AND destination= " +
destination + " ";
13
14             Statement st=con.createStatement();

```

```

15
16     ResultSet rst= st.executeQuery(query);
17
18     while(rst.next()){
19         int flightId= rst.getInt(1);
20         String src=rst.getString(2);
21         String dst=rst.getString(3);
22         int noofseats=rst.getInt(4);
23         double flightfare=rst.getDouble(5);
24
25         flightList.add(new Flight(flightId, src, dst, noofseats, flightfare));
26     }
27 }catch(ClassNotFoundException | SQLException e){
28     e.printStackTrace();
29 }
30 return flightList;
31 }
32
33 }

```

Main.java

```

1 import java.util.Scanner;
2 import java.util.ArrayList;
3
4 public class Main{
5     public static void main(String[] args){
6         Scanner sc=new Scanner(System.in);
7         System.out.println("Enter the source");
8         String source=sc.next();
9         System.out.println("Enter the destination");
10        String destination=sc.next();
11
12        FlightManagementSystem fms= new FlightManagementSystem();
13        ArrayList<Flight> flightList=fms.viewFlightBySourceDestination(source,destination);
14        if(flightList.isEmpty()){
15            System.out.println("No flights available for the given source and destination");
16            return;
17        }
18        System.out.println("Flightid Noofseats Flightfare");
19        for(Flight flight : flightList){
20            System.out.println(flight.getFlightId()+" "+flight.getNoOfSeats()+" "+flight.getFlightFare());
21        }
22
23    }
24 }

```

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 =
DriverManager.getConnection(props.getProperty("DB_URL"),props.getProperty("DB_USERNAME"),props.getPr
operty("DB_PASSWORD"));
27     }
28     catch(IOException e){
29         e.printStackTrace();
30     }
31     return con;
32 }
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=oracle.jdbc.driver.OracleDriver
8 DB_URL=jdbc:oracle:thin:@127.0.0.1:1521:XE
9 DB_USERNAME=${sys:db_username}
10 DB_PASSWORD=${sys:db_password}
11

```

Grade

Reviewed on Monday, 7 February 2022, 6:33 PM by Automatic grade

Grade 100 / 100

Assessment report

Assessment Completed Successfully

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

=====

2. Get Text and Display Welcome Message

Grade settings: Maximum grade: 100

Run: Yes **Evaluate:** Yes

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

Amir owns "Bouncing Babies" an exclusive online store for baby toys.

He desires to display a welcome message whenever a customer visits his online store and makes a purchase.

Help him do this by incorporating the customer name using the Lambda expression.

Requirement 1: Display Welcome message

Amir wants to display a welcome message for his customers. The method `displayText` is used to display the name of the customer who made an online purchase from his store.

Component Specification: DisplayText Interface – This is a Functional Interface.

Type(Interface)	Methods	Responsibilities
DisplayText	<code>public void displayText(String text)</code>	The purpose of this method is to display the welcome message by including the text provided as an argument by using Lambda expression.
DisplayText	<code>public default String getInput()</code>	This method should get a String (name of the customer) as input from the user and return the same. This method should be a default method.

Annotate the interface with the appropriate annotation

Component Specification: Main class

Component Name	Type(Class)	Methods	Responsibilities
Display welcome message	Main	<code>public static DisplayText welcomeMessage()</code>	This method should return a DisplayText object. To do this, implement the lambda expression to print the text received as a parameter in the <code>displayText</code> method as "Welcome <text>".

In the Main class write the main method and perform the given steps :

- Invoke the static method `welcomeMessage()`. It returns a DisplayText object.
- Capture the DisplayText object in a reference variable.
- Using that reference, invoke the default method `getInput`.
- It will return a String. Capture that String in a variable.
- Using the reference of DisplayText, invoke the `displayText` method by passing the String as a parameter.
- The output should be as shown in the sample data mentioned below.

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 name for classes, interfaces and methods as specified in the question.

Adhere to the code template, if provided.

Sample Input 1 :

Watson

Sample Output 1 :

Welcome Watson

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

DisplayText.java

```
1 import java.util.*;
2 @FunctionalInterface
3 public interface DisplayText
4 {
5     public void displayText(String text);
6     public default String getInput()
7     {
8         Scanner read = new Scanner(System.in);
9         String str = read.next();
10        return str;
11        //return null;
12    }
13 }
```

Main.java

```
1 public class Main
2 {
3     public static DisplayText welcomeMessage()
4     {
5
6         DisplayText dis = (str)->{
7
8             System.out.println("Welcome "+str);
9         };
10        return dis;
11    }
12    public static void main(String args[])
13    {
14        DisplayText dis=welcomeMessage();
15        String text = dis.getInput();
16        dis.displayText(text);
17    }
18 }
19 }
```

Grade

Reviewed on Wednesday, 1 December 2021, 10:14 PM by Automatic grade

Grade 100 / 100

Assessment report

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

=====

3. Generate Password

Grade settings: Maximum grade: 100

Run: Yes **Evaluate:** Yes

Automatic grade: Yes

Important Instructions:

- Please read the document thoroughly before you code.
- Import the given skeleton code into your Eclipse.(if provided)
- Do not change the Skeleton code or the package structure, method names, variable names, return types, exception clauses, access specifiers etc.
- You can create any number of private methods inside the given class.
- You can test your code from main() method of the program

The system administrator of an organization wants to set password for all the computers for security purpose. To generate a strong password, he wants to combine the username of each user of the system with the reverse of their respective usernames. Help them by using Lambda expressions that caters to their requirement.

Requirement 1: PasswordInfo

The Administrator wants to generate password for each system by making use of the passwordGeneration method based on the username which is passed as a string.

Component Specification: Password Info Interface – This is a Functional Interface.

Type(Interface)	Methods	Responsibilities
PasswordInfo	public String passwordGeneration(String username)	This method is used to generate the password based on the username and hence returns the generated password

Component Specification: Computer Class

Type(Class)	Methods	Responsibilities
Computer	public static PasswordInfo passwordPropagation()	This method should return a PasswordInfo object. To do this, implement the lambda expression to get the password.
	public static void displayUserDetails(String systemNo,String username>PasswordInfo passwordInfoObj)	This method is used to print the Password Info such as the systemNo, password along with the message, “Your password is generated successfully!!!” based

		on the systemNo, username, passwordInfoObj which is passed as an argument.
--	--	---

In the Computer class write the main method and perform the given steps:

- Get the systemNo and username from the user.
- Invoke the static method passwordPropagation(). It returns a passwordInfo object with the definition of the passwordGeneration method.
- Capture the PasswordInfo object in a reference variable.
- Invoke the displayUserDetails method by passing systemNo, username and passwordInfoObj as parameters.
- Inside the userDetails method, you should invoke the passwordGeneration method using the passwordInfo object and the output should be displayed as shown in the sample input/output.
- The output should be as shown in the sample data mentioned below.

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 use the lambda expression.
- Ensure to follow the object oriented specifications provided in the question.
- Ensure to provide the name for classes, interfaces and methods as specified in the question.
- Adhere to the code template, if provided.

Sample Input 1:

Enter system no

Tek/1234

Enter username

Manoj Kumar

Sample Output 1:

Password Info

System no: Tek/1234

Password: Manoj KumarramuK jonaM

Your password is generated successfully!!!

=====

4. Vatican Museum Manipulation

Grade settings: Maximum grade: 100

Run: Yes **Evaluate:** Yes

Automatic grade: Yes **Maximum execution time:** 60 s **Maximum memory used:** 64

MiB Maximum execution file size: 320 KiB

Important Instructions:

- Please read the document thoroughly before you code.
- Import the given skeleton code into your Eclipse.(if provided)
- Do not change the Skeleton code or the package structure, method names, variable names, return types, exception clauses, access specifiers etc.
- You can create any number of private methods inside the given class.
- You can test your code from the main() method of the program.

Vatican Museum is one of the famous museums, they have collections of houses paintings, and sculptures from artists. The Museum management stores their visitor's details in a text file. Now, they need an application to analyze and manipulate the visitor details based on the visitor visit date and the visitor address.

You are provided with a text file – VisitorDetails.txt, which contains all the visitor details like the visitor Id, visitor name, mobile number, date of visiting and address. Your application should satisfy the following requirements.

1. View visitor details within two given dates.
2. View visitor details which are above a particular mentioned visitor address.

You are provided with a code template which includes the following:

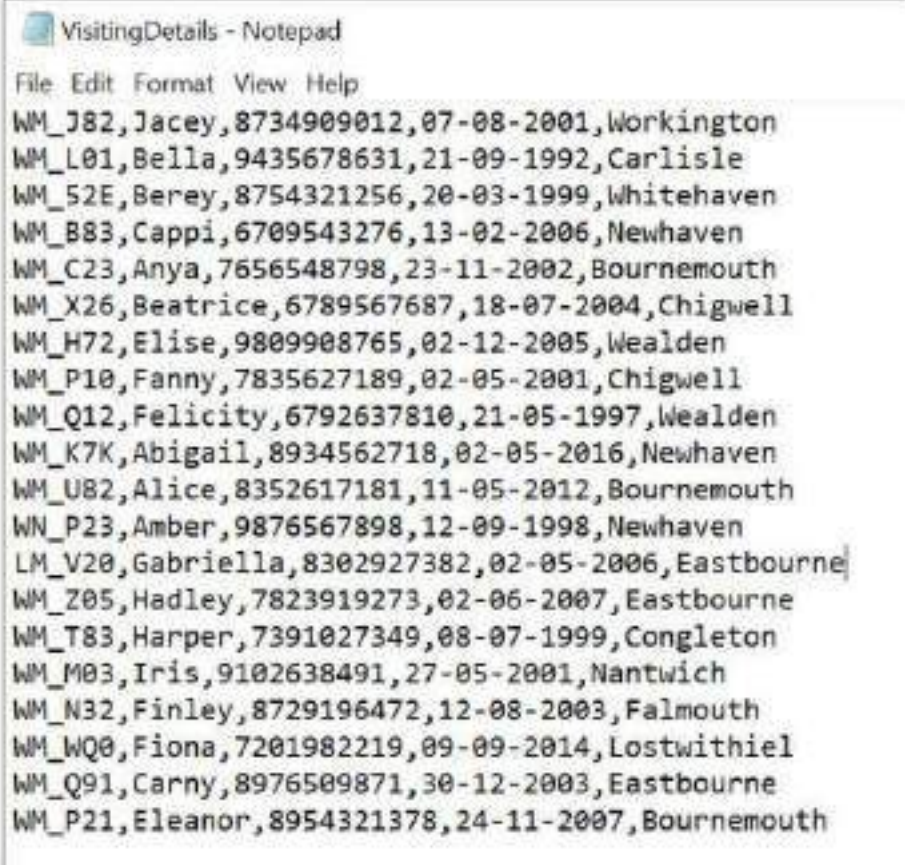
- Visitor class which includes the attributes visitorId, visitorName, mobileNumber, dateOfVisiting and address with all the getters and setters.
- VisitorUtility class which includes the following method declarations.
 - public List<Visitor> generateVisitor(String filePath)
 - public boolean isValidVisitorId(String visitorId)
 - public List<Visitor> viewVisitorDetailsByDateOfVisiting(Stream<Visitor> visitorStream, String fromDate, String toDate)
 - public Stream<Visitor> viewVisitorDetailsByAddress (Stream<Visitor> visitorStream, double address)
- InvalidVisitorIdException class which inherits the Exception class.
- Main class with a main method which creates the required user interface for the application.
- VisitorDetails.txt which contains all the visitor details like visitor id, visitor name, mobile number, date of visiting and address.

Note:

- The Visitor class and the Main class will be provided with all the necessary codes. Please do not edit or delete any line of the code in these two classes.
- Fill your code in the InvalidVisitorIdException class to create a constructor as described in the functional requirements below.
- Fill your code in the respective methods of VisitorUtility class to fulfil all the functional requirements.

- In the VisitorDetails.txt file, each visitor detail has information separated by a comma, and it is given as one customer detail per line.

Sample data in VisitorDetails.txt file



```

File Edit Format View Help
WM_J82,Jacey,8734909012,07-08-2001,Workington
WM_L01,Bella,9435678631,21-09-1992,Carlisle
WM_52E,Berey,8754321256,20-03-1999,Whitehaven
WM_B83,Cappi,6709543276,13-02-2006,Newhaven
WM_C23,Anya,7656548798,23-11-2002,Bournemouth
WM_X26,Beatrice,6789567687,18-07-2004,Chigwell
WM_H72,Elise,9809908765,02-12-2005,Wealden
WM_P10,Fanny,7835627189,02-05-2001,Chigwell
WM_Q12,Felicity,6792637810,21-05-1997,Wealden
WM_K7K,Abigail,8934562718,02-05-2016,Newhaven
WM_U82,Alice,8352617181,11-05-2012,Bournemouth
WM_P23,Amber,9876567898,12-09-1998,Newhaven
LM_V20,Gabriella,8302927382,02-05-2006,Eastbourne
WM_Z05,Hadley,7823919273,02-06-2007,Eastbourne
WM_T83,Harper,7391027349,08-07-1999,Congleton
WM_M03,Iris,9102638491,27-05-2001,Nantwich
WM_N32,Finley,8729196472,12-08-2003,Falmouth
WM_WQ0,Fiona,7201982219,09-09-2014,Lostwithiel
WM_Q91,Carny,8976509871,30-12-2003,Eastbourne
WM_P21,Eleanor,8954321378,24-11-2007,Bournemouth
  
```

Functional Requirements:

Fill your code in the respective class and method declarations based on the required functionalities as given below.

Class	Attributes/ Methods	Rules/ Responsibility
VisitorUtility	<pre> public List < Visitor> generateVisitor(String filePath) </pre>	<p>Read the text file and convert each line in the text file as String and store it in a List. Each String from the List should be converted into a visitor object and each visitor object should be stored in a List. Return the List of visitors.</p> <p>Note:</p> <p>Before converting the separated string into a visitor object, the identified visitorId should be validated using the</p>

		isValidVisitorId method.
VisitorUtility	public boolean isValidVisitorId (String visitorId)	<p>Should check whether the provided visitorId is valid or not.</p> <p>If valid, this method should return true.</p> <p>If invalid, this method should handle an InvalidVisitorIdException with a message "<visitorId> is Invalid Visitor Id".</p> <p>Validation Rules:</p> <ul style="list-style-type: none"> · Length of the visitorId should be exactly 6. · The visitorId should start with "WM_" and the next letter should be an alphabet (A-Z) in upper case and the last two letters should be positive integers(0-9). <p>Example.</p> <p>WM_A23</p>
InvalidVisitorIdException	Create a constructor with a single String argument and pass it to the parent class constructor.	This class Should inherit the Exception class. The constructor should pass the String message which is thrown to it by calling the parent class constructor.

Requirement 1: View visitor details between the dates of visiting

Class	Attributes/ Methods	Rules/ Responsibility
VisitorUtility	public List<Visitor> viewVisitorDetailsByDateOfVisiting(Stream<Visitor> visitorStream, String fromDate, String toDate)	From the provided Stream of Visitor, separate the visitor details which has the date of visiting between fromDate and toDate (both inclusive). Return the

		separated visitor details as a list.
--	--	--------------------------------------

Requirement 2: View visitor details which are above a particular mentioned address

Class	Attributes/ Methods	Rules/ Responsibility
VisitorUtility	public Stream<Visitor> viewVisitorDetailsByAddress(Stream<Visitor> visitorStream, String address)	From the given Stream of Visitor, separate the visitor details based on address, which has a particular mentioned address as provided. Return the separated Stream of visitor.

Note:

1. All inputs/ outputs for processing the functional requirements should be case sensitive.
2. Adhere to the Sample Inputs/ Outputs
3. In the Sample Inputs/ Outputs provided, the highlighted text in bold corresponds to the input given by the user and the rest of the text represents the output.
4. All the Date values used in this application must be in "dd-MM-yyyy" format.
5. Adhere to the code template.
6. Fill all your required codes in the respective blocks. Do not edit or delete the codes provided in the code template.
7. The Sample Inputs/ Outputs given below are generated based on the Sample data given in the VisitorDetails.txt file.
8. Please do not hard code the output.

Sample Input/ Output 1:

WM_52E is Invalid Visitor Id

WM_K7K is Invalid Visitor Id

WN_P23 is Invalid Visitor Id

LM_V20 is Invalid Visitor Id

WM_WQ0 is Invalid Visitor Id

1. ViewVisitorDetailsByDateOfVisiting

2. ViewVisitorDetailsByAddress

Enter your choice

1

Enter the starting date

19-05-2004

Enter the ending date

07-04-2012

WM_B83 Cappi 6709543276 13-02-2006 Newhaven

WM_X26 Beatrice 6789567687 18-07-2004 Chigwell

WM_H72 Elise 9809908765 02-12-2005 Wealden

WM_Z05 Hadley 7823919273 02-06-2007 Eastbourne

WM_P21 Eleanor 8954321378 24-11-2007 Bournemouth

Sample Input/ Output 2:

WM_52E is Invalid Visitor Id

WM_K7K is Invalid Visitor Id

WN_P23 is Invalid Visitor Id

LM_V20 is Invalid Visitor Id

WM_WQ0 is Invalid Visitor Id

1. viewVisitorDetailsByDateOfVisiting

2. viewVisitorDetailsByAddress

Enter your choice

2

Enter the address

Eastbourne

WM_Z05 Hadley 7823919273 02-06-2007 Eastbourne

WM_Q91 Carny 8976509871 30-12-2003 Eastbourne

5. Hospital Management_Streams

Grade settings: Maximum grade: 100

Run: Yes **Evaluate:** Yes

Automatic grade: Yes

Laxmi Hospital is a world-class health care institution providing patient treatment with specialized medical and nursing staff and medical equipment. It typically provides an emergency department to treat urgent health problems ranging from fire and accident victims to sudden illness. The hospital maintains a register to maintain the records of the patients who enter the emergency department. The receptionist at the helpdesk would like to filter the patients based on a criterion. Develop a java application for the same using Streams.

Requirements:

1. Read the patient records from the file.
2. Retrieve the patient details for the specified date interval.
3. Retrieve the patient details which are from a particular area (address).

Component Specification: Patient (POJO Class)

Type (Class)	Attributes	Methods
Patient	String patientId String patientName String contactNumber String dateOfVisit String patientAddress	Getters and Setters are given in the code skeleton.

Component Specification: PatientUtility

Type (Class)	Methods	Responsibilities
-----------------	---------	------------------

PatientUtility	public List <Patient> fetchPatient(String filePath)	<p>Read the file using File I/O or Java Streams and return the validated list of patient records. It should filter the valid patient records based on the valid patient Id using the method isValidPatientId ().</p> <p>Note: Make sure that the user-defined exception is handled in this method itself.</p>
PatientUtility	public boolean isValidPatientId (String patientId)	<p>Validation Guidelines for Valid Patient ID:</p> <ul style="list-style-type: none"> • The length of the Patient Id should be exactly 6. • The Patient Id should start with “WM_” and the next letter should be an alphabet (A-Z) in upper case and the last two letters should be positive integers(0-9). Example. WM_A10. <p>Check whether the patient Id is valid or not. If invalid, this method should handle an InvalidPatientIdException with a message “<patientid> is an Invalid Patient Id”.</p>
PatientUtility	public List<Patient> retrievePatientRecords_ByDateOfVisit(Stream<Patient> patientStream, String fromDate, String toDate)	<p>From the provided stream of patient, separate the patient details which has the date of visit between fromDate and toDate (both inclusive) and return the resultant patient records as a list.</p>

PatientUtility	<pre>public Stream<Patient> retrievePatientRecords_ByAddress(Stream<Patient> patientStream, String address)</pre>	From the given stream of patient, filter the patient details based on the user input address, and return the separated Stream of patients.
----------------	---	--

Component Specification: InvalidPatientIdException (User defined Exception)

Type (Class)	Methods	Responsibilities
InvalidPatientIdException	<pre>public InvalidPatientIdException(String message)</pre>	This constructor should set the message to the superclass.

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

You are provided with a text file –PatientRegister.txt, which contains all the patient details like the patient Id, patient name, contact number, date of visit, and patient address. You can add any number of records in the text file to test your code.

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 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:

Invalid Patient Id are:

WM_52E is an Invalid Patient Id

WM_K7K is an Invalid Patient Id

WN_P23 is an Invalid Patient Id

LM_V20 is an Invalid Patient Id

WM_WQ0 is an Invalid Patient Id

Retrieve Patient Details

1. By Date of Visit

2. By Address

Enter your choice:

1

Enter the start date

02-03-2003

Enter the end date

02-12-2005

WM_X26 Beatrice 6789567687 18-07-2004 Texas

WM_H72 Elise 9809908765 02-12-2005 Washington

WM_N32 Finley 8729196472 12-08-2003 Pennsylvania

WM_Q91 Carny 8976509871 30-12-2003 Virginia

Sample Input/Output 2:

Invalid Patient Id are:

WM_52E is an Invalid Patient Id

WM_K7K is an Invalid Patient Id

WN_P23 is an Invalid Patient Id

LM_V20 is an Invalid Patient Id

WM_WQ0 is an Invalid Patient Id

Retrieve Patient Details

1. By Date of Visit

2. By Address

Enter your choice:

2

Enter the address

Carolina

WM_C23 Anya 7656548798 23-11-2002 Carolina

WM_T83 Harper 7391027349 08-07-1999 Carolina

WM_P21 Eleanor 8954321378 24-11-2007 Carolina

Sample Input/Output 3:

Invalid Patient Id are:

WM_52E is an Invalid Patient Id

WM_K7K is an Invalid Patient Id

WN_P23 is an Invalid Patient Id

LM_V20 is an Invalid Patient Id

WM_WQ0 is an Invalid Patient Id

Retrieve Patient Details

1. By Date of Visit

2. By Address

Enter your choice:

1

Enter the start date

03-02-2020

Enter the end date

02-02-2021

No patient records available during this interval

Sample Input/Output 4:

Invalid Patient Id are:

WM_52E is an Invalid Patient Id

WM_K7K is an Invalid Patient Id

WN_P23 is an Invalid Patient Id

LM_V20 is an Invalid Patient Id

WM_WQ0 is an Invalid Patient Id

Retrieve Patient Details

1. By Date of Visit

2. By Address

Enter your choice:

3

Invalid Option

Automatic evaluation[+]

HospitalManagement/PatientRegister.txt

```
1 WM_J82,Jacey,8734909012,07-08-2001,Colorado
2 WM_L01,Bella,9435678631,21-09-1992,Connecticut
3 WM_52E,Berey,8754321256,20-03-1999,Indiana
4 WM_B83,Cappi,6709543276,13-02-2006,Pennsylvania
5 WM_C23,Any,7656548798,23-11-2002,Carolina
6 WM_X26,Beatrice,6789567687,18-07-2004,Texas
7 WM_H72,Elise,9809908765,02-12-2005,Washington
8 WM_P10,Fanny,7835627189,02-05-2001,Virginia
9 WM_Q12,Felicity,6792637810,21-05-1997,Colorado
10 WM_K7K,Abigail,8934562718,02-05-2016,Indiana
11 WM_U82,Alice,8352617181,11-05-2012,Indiana
12 WN_P23,Amber,9876567898,12-09-1998,Pennsylvania
13 LM_V20,Gabriella,8302927382,02-05-2006,Connecticut
14 WM_Z05,Hadley,7823919273,02-06-2007,Connecticut
15 WM_T83,Harper,7391027349,08-07-1999,Carolina
16 WM_M03,Iris,9102638491,27-05-2001,Texas
17 WM_N32,Finley,8729196472,12-08-2003,Pennsylvania
18 WM_WQ0,Fiona,7201982219,09-09-2014,Washington
19 WM_Q91,Carny,8976509871,30-12-2003,Virginia
20 WM_P21,Eleanor,8954321378,24-11-2007,Carolina
```

HospitalManagement/src/InvalidPatientIdException.java

```
1
2 //public class InvalidPatientIdException{
3     //FILL THE CODE HERE
4     public class InvalidPatientIdException extends Exception{
5         public InvalidPatientIdException(String message){
6             super(message);
7         }
8     }
9
10
11
12
```

HospitalManagement/src/Main.java

```
1 public class Main {
2
3     public static void main(String[] args){
4
5         // CODE SKELETON - VALIDATION STARTS
6         // DO NOT CHANGE THIS CODE
7
8         new SkeletonValidator();
9         // CODE SKELETON - VALIDATION ENDS
10
11         // FILL THE CODE HERE
12
13     }
14
```

```
15     }
16
17
```

HospitalManagement/src/Patient.java

```
1 //DO NOT ADD/EDIT THE CODE
2 public class Patient {
3
4     private String patientId;
5     private String patientName;
6     private String contactNumber;
7     private String dateOfVisit;
8     private String patientAddress;
9
10    //Setters and Getters
11
12    public String getPatientId() {
13        return patientId;
14    }
15    public void setPatientId(String patientId) {
16        this.patientId = patientId;
17    }
18    public String getPatientName() {
19        return patientName;
20    }
21    public void setPatientName(String patientName) {
22        this.patientName = patientName;
23    }
24    public String getContactNumber() {
25        return contactNumber;
26    }
27    public void setContactNumber(String contactNumber) {
28        this.contactNumber = contactNumber;
29    }
30    public String getDateOfVisit() {
31        return dateOfVisit;
32    }
33    public void setDateOfVisit(String dateOfVisit) {
34        this.dateOfVisit = dateOfVisit;
35    }
36    public String getPatientAddress() {
37        return patientAddress;
38    }
39    public void setPatientAddress(String patientAddress) {
40        this.patientAddress = patientAddress;
41    }
42
43
44
45
46 }
47
```

HospitalManagement/src/PatientUtility.java

```
1 import java.util.List;
2 import java.util.stream.Stream;
3 import java.util.ArrayList;
4 import java.io.File;
5 import java.io.FileNotFoundException;
6 import java.util.Scanner;
7 import java.util.regex.*;
8 import java.util.stream.Collectors;
9 import java.text.ParseException;
10 import java.text.SimpleDateFormat;
11 import java.util.Date;
12
13
```

```

14 public class PatientUtility {
15
16     public List <Patient> fetchPatient(String filePath) {
17
18
19         //FILL THE CODE HERE
20         List <Patient> patients =new ArrayList<>();
21         try{
22             File register =new File(filePath);
23             Scanner reader=new Scanner(register);
24             while(reader.hasNextLine()){
25                 Patient p = new Patient();
26                 String[] infos=reader.nextLine().split(",");
27                 try{
28                     if(isValidPatientId(infos[0])){
29                         p.setPatientId(infos[0]);
30                         p.setPatientName(infos[1]);
31                         p.setContactNumber(infos[2]);
32                         p.setDateOfVisit(infos[3]);
33                         p.setPatientAddress(infos[4]);
34                         patients.add(p);
35                     }
36                 }
37                 catch(InvalidPatientIdException e1){
38                     System.out.println(e1.getMessage());
39                 }
40             }
41             reader.close();
42         }
43         catch(FileNotFoundException e){}
44         return patients;
45
46         //return null;
47     }
48
49     public boolean isValidPatientId (String patientId)throws InvalidPatientIdException
50     {
51
52         //FILL THE CODE HERE
53         Pattern p =Pattern.compile("WM_[A-Z][0-9]{2}$");
54         Matcher m=p.matcher(patientId);
55         boolean ne =m.matches();
56         if(!ne){
57             throw new InvalidPatientIdException(patientId+"is an Invalid Patient Id.");
58         }
59
60         //return inValid;
61         return ne;
62     }
63
64
65     public List<Patient> retrievePatientRecords_ByDateOfVisit(Stream<Patient> patientStream, String
66 fromDate, String toDate)
67     {
68         //FILL THE CODE HERE
69         SimpleDateFormat simpleDateFormat=new SimpleDateFormat("dd-MM-yyyy");
70         return patientStream
71             .filter((p)->{
72                 try{
73                     Date start=simpleDateFormat.parse(fromDate);
74                     Date end= simpleDateFormat.parse(toDate);
75                     Date current =simpleDateFormat.parse(p.getDateOfVisit());
76                     return start.compareTo(current)*current.compareTo(end)>=0;
77                 }
78                 catch(ParseException e){}
79                 return false;

```

```

80         }).collect(Collectors.toList());
81         //         return null;
82     }
83
84
85
86     public Stream<Patient> retrievePatientRecords_ByAddress(Stream<Patient> patientStream, String
address)
87     {
88
89         //FILL THE CODE HERE
90         return patientStream.filter(p->address.equals(p.getPatientAddress()));
91         //return null;
92
93
94
95     }
96
97 }
98

```

HospitalManagement/src/SkeletonValidator.java

```

1  import java.lang.reflect.Method;
2  import java.util.List;
3  import java.util.logging.Level;
4  import java.util.logging.Logger;
5  import java.util.stream.Stream;
6
7  /**
8   * @author TJ
9   *
10  * This class is used to verify if the Code Skeleton is intact and not modified by participants thereby ensuring
smooth auto evaluation
11  *
12  */
13  public class SkeletonValidator {
14
15      public SkeletonValidator() {
16
17
18          validateClassName("Patient");
19          validateClassName("PatientUtility");
20          validateClassName("InvalidPatientIdException");
21          validateMethodSignature(
22
23              "fetchPatient:java.util.List,isValidPatientId:boolean,retrievePatientRecords_ByDateOfVisit:java.util.List
,retrievePatientRecords_ByAddress:java.util.stream.Stream",
24              "PatientUtility");
25      }
26
27      private static final Logger LOG = Logger.getLogger("SkeletonValidator");
28
29      protected final boolean validateClassName(String className) {
30
31          boolean iscorrect = false;
32          try {
33              Class.forName(className);
34              iscorrect = true;
35              LOG.info("Class Name " + className + " is correct");
36
37          } catch (ClassNotFoundException e) {
38              LOG.log(Level.SEVERE, "You have changed either the " + "class
name/package. Use the correct package "
39                  + "and class name as provided in the skeleton");
40
41          } catch (Exception e) {

```



```

42             LOG.log(Level.SEVERE,
43                 "There is an error in validating the " + "Class Name.
Please manually verify that the "
44                 + "Class name is same as
skeleton before uploading");
45         }
46         return incorrect;
47     }
48 }
49
50     protected final void validateMethodSignature(String methodWithExcpn, String className) {
51         Class cls = null;
52         try {
53
54             String[] actualMethods = methodWithExcpn.split(",");
55             boolean errorFlag = false;
56             String[] methodSignature;
57             String methodName = null;
58             String returnType = null;
59
60             for (String singleMethod : actualMethods) {
61                 boolean foundMethod = false;
62                 methodSignature = singleMethod.split(":");
63
64                 methodName = methodSignature[0];
65                 returnType = methodSignature[1];
66                 cls = Class.forName(className);
67                 Method[] methods = cls.getMethods();
68                 for (Method findMethod : methods) {
69                     if (methodName.equals(findMethod.getName())) {
70                         foundMethod = true;
71                         if
72                         (!findMethod.getReturnType().getName().equals(returnType))) {
73                             errorFlag = true;
74                             LOG.log(Level.SEVERE, " You
have changed the " + "return type in " + methodName
75                             + "
method. Please stick to the " + "skeleton provided");
76                         } else {
77                             LOG.info("Method signature of "
+ methodName + " is valid");
78                         }
79                     }
80                 }
81             }
82             if (!foundMethod) {
83                 errorFlag = true;
84                 LOG.log(Level.SEVERE, " Unable to find the given
public method " + methodName
85                 + ". Do not change the " + "given
public method name. " + "Verify it with the skeleton");
86             }
87         }
88     }
89     if (!errorFlag) {
90         LOG.info("Method signature is valid");
91     }
92 }
93     } catch (Exception e) {
94         LOG.log(Level.SEVERE,
95             " There is an error in validating the " + "method
structure. Please manually verify that the "
96             + "Method signature is same as
the skeleton before uploading");
97     }
98 }

```

99
100 }

Grade

Reviewed on Monday, 7 February 2022, 6:04 PM by Automatic grade

Grade 100 / 100

Assessment report

Assessment Completed Successfully

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

6. Technology Fest

Grade settings: Maximum grade: 100

Run: Yes **Evaluate:** Yes

Automatic grade: Yes

Institute of Technology is organizing an All-India Technology Fest for various engineering colleges across the country. The management would like to automate the registration so that it is easier and more systematic while conducting the fest. Create a java application for the same using Threads.

Component Specification: Participant (POJO Class)

Type (Class)	Attributes	Methods
Participant	String name String yearofstudy String department String collegeName String eventName double registrationFee	Getters, Setters, and a five-argument constructor in the given order - name, yearofstudy, department, collegeName, eventName are included in the code Skeleton.

Requirements:

- To calculate the registration fee of the participant based on the event name.
- To calculate the number of participants registered for a particular event.

Sl No	Event Name	Registration Fee
1	Robocar	1000
2	PaperTalk	500
3	Quiz	300
4	Games	100

***Note that Event name is case in- sensitive**

Component Specification: EventManagement (Thread Class)

Type (Class)	Attributes	Methods	Responsibilities
EventManagement	List<Participant> TechList String searchEvent int counter		Include getters and setter methods for all the attributes.
EventManagement		public void calculateRegistrationFee(List<Participant> list)	Calculate the registration fee of the participant based on the event name. If the event name doesn't exist, throw an InvalidEventException with an error message "Event Name is invalid".
EventManagement		public void run()	Calculate the number of participants registered for a particular event. Increment the counter attribute based on the search.

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

Component Specification: InvalidEventException

Type (Class)	Methods	Responsibilities
InvalidEventException	public InvalidEventException (String message)	To set the message string to the superclass.

Create a class called Main with the main method and perform the tasks are given below:

- Get the inputs as provided in the sample input.
- Call the calculateRegistrationFee () method to calculate the registration fee of the participant based on the event name.
- Print the list of Participant objects with the registration fee.
- Get the event type to search to find the number of the participants registered for that particular event.
- Handle the user-defined exception in the main method.
- Display the output as shown in the sample input/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 represent the output.
- Ensure to follow the 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 entries

3

Enter the Participant Name/Yearofstudy/Department/CollegeName/EventName

rinu/4/EEE/mnm/robocar

fina/3/EEE/psg/papertalk

rachel/4/civil/kcg/quiz

Print participant details

ParticipantName=rinu, Yearofstudy=4, Department=EEE, CollegeName=mnm,
EventName=robocar, RegistrationFee=1000.0

ParticipantName=fina, Yearofstudy=3, Department=EEE, CollegeName=psg,
EventName=papertalk, RegistrationFee=500.0

ParticipantName=rachel, Yearofstudy=4, Department=civil, CollegeName=kcg,
EventName=quiz, RegistrationFee=300.0

Enter the event to search

robocar

Number of participants for ROBOCAR event is 1

Sample Input/Output 2:

Enter the number of entries

3

Enter the Participant Name/Yearofstudy/Department/CollegeName/EventName

rinu/4/EEE/mnm/robocar

fina/3/EEE/psg/papertalk

rachel/4/civil/kcg/quiz

Print participant details

ParticipantName=rinu, Yearofstudy=4, Department=EEE, CollegeName=mnm,
EventName=robocar, RegistrationFee=1000.0

ParticipantName=fina, Yearofstudy=3, Department=EEE, CollegeName=psg,
EventName=papertalk, RegistrationFee=500.0

ParticipantName=rachel, Yearofstudy=4, Department=civil, CollegeName=kcg,
EventName=quiz, RegistrationFee=300.0

Enter the event to search

games

No participant found

Sample Input/Output 3:

Enter the number of entries

2

Enter the Participant Name/Yearofstudy/Department/CollegeName/EventName

vishal/4/mech/vjc/flyingrobo

vivek/3/mech/hdl/games

Event Name is invalid

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

TechnologyFest/src/EventManager.java

```
1 import java.util.List;
2
3 public class EventManagement implements Runnable {
4     private List<Participant> TechList;
5     private String searchEvent;
6     private int counter=0;
7     public List<Participant>getTechList()
8     {
9         return TechList;
10    }
11
12    public void setTechList(List<Participant>techList)
13    {
14        TechList=techList;
15    }
16    public String getSearchEvent()
17    {
18        return searchEvent;
```

```

19     }
20     public void setSearchEvent(String searchEvent)
21     {
22         this.searchEvent=searchEvent;
23     }
24     public int getCounter()
25     {
26         return counter;
27     }
28     public void setCounter(int counter)
29     {
30         this.counter=counter;
31     }
32     //FILL THE CODE HERE
33
34     public void calculateRegistrationFee(List<Participant> list) throws InvalidEventException
35     {
36         for(Participant p:list)
37         {
38             if(p.getEventName().equalsIgnoreCase("robocar"))
39             {
40                 p.setRegistrationFee(1000);
41             }
42             else if(p.getEventName().equalsIgnoreCase("papertalk")){
43                 p.setRegistrationFee(500);
44             }
45             }
46         }
47         else if(p.getEventName().equalsIgnoreCase("quiz")){
48             p.setRegistrationFee(300);
49         }
50         else if(p.getEventName().equalsIgnoreCase("games")){
51             p.setRegistrationFee(100);
52         }
53         else{
54             throw new InvalidEventException("Event Name is Invalid");
55         }
56     }
57     }
58     //FILL THE CODE HERE
59     setTechList(list);
60 }
61
62 public void run()
63 {
64     String str="robocarpapertalkquizgames";
65     if(str.contains(this.getSearchEvent())){
66         for(Participant P:this.getTechList()){
67             if(this.getSearchEvent().equals(P.getEventName())){
68                 counter++;
69             }
70         }
71     }
72     setCounter(counter);
73
74     //FILL THE CODE HERE
75
76 }
77 }
78

```

TechnologyFest/src/InvalidEventException.java

```

1 public class InvalidEventException extends Exception{
2     //FILL THE CODE HERE
3 public InvalidEventException(String str){
4     super(str);
5

```

```

6 }
7
8 }
9

```

TechnologyFest/src/Main.java

```

1
2 import java.util.Scanner;
3 import java.util.*;
4 public class Main {
5     public static void main(String [] args)
6     {
7         // CODE SKELETON - VALIDATION STARTS
8         // DO NOT CHANGE THIS CODE
9
10        new SkeletonValidator();
11
12        // CODE SKELETON - VALIDATION ENDS
13
14        Scanner sc=new Scanner(System.in);
15        System.out.println("Enter the number of entries");
16        int n=sc.nextInt();
17        System.out.println("Enter the Participant
Name/Yearofstudy/Department/CollegeName/EventName");
18        List<Participant> list=new ArrayList<Participant>();
19        String strlist[]=new String[n];
20        for(int i=0;i<n;i++)
21        {
22            strlist[i]=sc.next();
23            String a[]=strlist[i].split("/");
24            Participant pt=new Participant(a[0],a[1],a[2],a[3],a[4]);
25            list.add(pt);
26        }
27        EventManagement em=new EventManagement();
28        try {
29            em.calculateRegistrationFee(list);
30        }
31        catch(InvalidEventException e)
32        {
33            e.printStackTrace();
34        }
35        System.out.println("Print participant details");
36        for(Participant p:list)
37        {
38            System.out.println(p);
39        }
40        System.out.println("Enter the event to search");
41        String srch=sc.nextLine();
42        em.setSearchEvent(srch);
43        em.run();
44        int count=em.getCounter();
45        if(count<=0){
46            System.out.println("No participant found");
47        }
48        else{
49            System.out.println("Number of participants for"+srch+"event is "+count);
50        }
51    }
52 }
53
54
55
56
57

```

TechnologyFest/src/Participant.java

```

1 public class Participant {

```

```

2     private String name;
3     private String yearofstudy;
4     private String department;
5     private String collegeName;
6     private String eventName;
7     private double registrationFee;
8
9     //5 argument Constructor
10    public Participant(String name, String yearofstudy, String department, String collegeName, String
eventName) {
11        super();
12        this.name = name;
13        this.yearofstudy = yearofstudy;
14        this.department = department;
15        this.collegeName = collegeName;
16        this.eventName = eventName;
17    }
18
19    public String getName() {
20        return name;
21    }
22    public void setName(String name) {
23        this.name = name;
24    }
25    public String getYearofstudy() {
26        return yearofstudy;
27    }
28    public void setYearofstudy(String yearofstudy) {
29        this.yearofstudy = yearofstudy;
30    }
31    public String getDepartment() {
32        return department;
33    }
34    public void setDepartment(String department) {
35        this.department = department;
36    }
37    public String getCollegeName() {
38        return collegeName;
39    }
40    public void setCollegeName(String collegeName) {
41        this.collegeName = collegeName;
42    }
43    public String getEventName() {
44        return eventName;
45    }
46    public void setEventName(String eventName) {
47        this.eventName = eventName;
48    }
49    public double getRegistrationFee() {
50        return registrationFee;
51    }
52    public void setRegistrationFee(double registrationFee) {
53        this.registrationFee = registrationFee;
54    }
55
56    @Override
57    public String toString() {
58        return "Participant [name=" + name + ", yearofstudy=" + yearofstudy + ", department=" +
department
59        + ", collegeName=" + collegeName + ", eventName=" +
eventName + ", registrationFee=" + registrationFee
60        + "];"
61    }
62
63
64
65

```



```

66 }
67
TechnologyFest/src/SkeletonValidator.java
1
2 import java.lang.reflect.Method;
3 import java.util.List;
4 import java.util.logging.Level;
5 import java.util.logging.Logger;
6 import java.util.stream.Stream;
7
8 /**
9  * @author TJ
10  *
11  * This class is used to verify if the Code Skeleton is intact and not modified by participants thereby ensuring
smooth auto evaluation
12  *
13  */
14 public class SkeletonValidator {
15
16     public SkeletonValidator() {
17
18         //classes
19         validateClassName("Main");
20         validateClassName("EventManager");
21         validateClassName("Participant");
22         validateClassName("InvalidEventException");
23         //functional methods
24         validateMethodSignature(
25             "calculateRegistrationFee:void", "EventManager");
26         validateMethodSignature(
27             "run:void", "EventManager");
28
29         //setters and getters of HallHandler
30         validateMethodSignature(
31             "getTechList:List", "EventManager");
32         validateMethodSignature(
33             "setTechList:void", "EventManager");
34
35         validateMethodSignature(
36             "getCounter:int", "EventManager");
37         validateMethodSignature(
38             "setCounter:void", "EventManager");
39
40         validateMethodSignature(
41             "getSearchEvent:String", "EventManager");
42         validateMethodSignature(
43             "setSearchEvent:void", "EventManager");
44
45         //setters and getters of Hall
46         validateMethodSignature(
47             "getName:String", "Participant");
48         validateMethodSignature(
49             "setName:void", "Participant");
50
51         validateMethodSignature(
52             "getYearofstudy:String", "Participant");
53         validateMethodSignature(
54             "setYearofstudy:void", "Participant");
55
56         validateMethodSignature(
57             "getDepartment:String", "Participant");
58         validateMethodSignature(
59             "setDepartment:void", "Participant");
60
61         validateMethodSignature(
62             "getCollegeName:String", "Participant");

```

```

63         validateMethodSignature(
64             "setCollegeName:void", "Participant");
65
66         validateMethodSignature(
67             "getEventName:String", "Participant");
68         validateMethodSignature(
69             "setEventName:void", "Participant");
70
71         validateMethodSignature(
72             "getRegistrationFee:double", "Participant");
73         validateMethodSignature(
74             "setRegistrationFee:void", "Participant");
75
76     }
77
78     private static final Logger LOG = Logger.getLogger("SkeletonValidator");
79
80     protected final boolean validateClassName(String className) {
81
82         boolean incorrect = false;
83         try {
84             Class.forName(className);
85             incorrect = true;
86             LOG.info("Class Name " + className + " is correct");
87
88         } catch (ClassNotFoundException e) {
89             LOG.log(Level.SEVERE, "You have changed either the " + "class
name/package. Use the correct package "
90                 + "and class name as provided in the skeleton");
91
92         } catch (Exception e) {
93             LOG.log(Level.SEVERE,
94                 "There is an error in validating the " + "Class Name.
Please manually verify that the "
95                 + "Class name is same as
skeleton before uploading");
96         }
97         return incorrect;
98
99     }
100
101     protected final void validateMethodSignature(String methodWithExcpn, String className) {
102         Class cls = null;
103         try {
104
105             String[] actualMethods = methodWithExcpn.split(",");
106             boolean errorFlag = false;
107             String[] methodSignature;
108             String methodName = null;
109             String returnType = null;
110
111             for (String singleMethod : actualMethods) {
112                 boolean foundMethod = false;
113                 methodSignature = singleMethod.split(":");
114
115                 methodName = methodSignature[0];
116                 returnType = methodSignature[1];
117                 cls = Class.forName(className);
118                 Method[] methods = cls.getMethods();
119                 for (Method findMethod : methods) {
120                     if (methodName.equals(findMethod.getName())) {
121                         foundMethod = true;
122                         if
(!findMethod.getReturnType().getName().contains(returnType)) {
123                             errorFlag = true;
124                             LOG.log(Level.SEVERE, " You
have changed the " + "return type in " + methodName

```

```

125                                     + ""
method. Please stick to the " + "skeleton provided");
126
127                                     } else {
128                                     LOG.info("Method signature of "
+ methodName + " is valid");
129                                     }
130
131                                     }
132                                     }
133                                     if (!foundMethod) {
134                                     errorFlag = true;
135                                     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");
137                                     }
138
139                                     }
140                                     if (!errorFlag) {
141                                     LOG.info("Method signature is valid");
142                                     }
143
144                                     } catch (Exception e) {
145                                     LOG.log(Level.SEVERE,
146                                     " There is an error in validating the " + "method
structure. Please manually verify that the "
+ "Method signature is same as
the skeleton before uploading");
148                                     }
149     }
150
151 }

```

Grade

Reviewed on Monday, 7 February 2022, 6:34 PM by Automatic grade

Grade 74 / 100

Assessment report

```

Fail 1 -- test4CheckTheOutput::
$Expected output:"[Print participant details
ParticipantName=Weni
Yearofstudy=3
Department=civil
CollegeName=vjc
EventName=robocar
RegistrationFee=1000.0
ParticipantName=gina
Yearofstudy=2
Department=mech
CollegeName=vjc
EventName=quiz
RegistrationFee=300.0
ParticipantName=jos
Yearofstudy=4
Department=ece
CollegeName=vjec
EventName=games
RegistrationFee=100.0
ParticipantName=fida
Yearofstudy=1
Department=eee

```

```

CollegeName=vjec
EventName=papertalk
RegistrationFee=500.0
Enter the event to search
Number of participants for PAPERTALK event is 1]" Actual output:"[Enter the number of
entries
Enter the Participant Name/Yearofstudy/Department/CollegeName/EventName
Print participant details
Participant [name=Weni
yearofstudy=3
department=civil
collegeName=vjc
eventName=robocar
registrationFee=1000.0]
Participant [name=gina
yearofstudy=2
department=mech
collegeName=vjc
eventName=quiz
registrationFee=300.0]
Participant [name=jos
yearofstudy=4
department=ece
collegeName=vjec
eventName=games
registrationFee=100.0]
Participant [name=fida
yearofstudy=1
department=eee
collegeName=vjec
eventName=papertalk
registrationFee=500.0]
Enter the event to search
No participant found]"$
Check your code with the input :Weni/3/civil/vjc/robocar
gina/2/mech/vjc/quiz
jos/4/ece/vjec/games
fida/1/eee/vjec/papertalk

```

```

Fail 2 -- test6CheckTheOutputfor_NCount::
$Expected output:"[Print participant details
ParticipantName=philip
Yearofstudy=4
Department=eee
CollegeName=mvc
EventName=robocar
RegistrationFee=1000.0
ParticipantName=susan
Yearofstudy=4
Department=eee
CollegeName=mvc
EventName=robocar
RegistrationFee=1000.0
ParticipantName=vivek
Yearofstudy=3
Department=civil
CollegeName=mvc
EventName=quiz
RegistrationFee=300.0
ParticipantName=vishal
Yearofstudy=3
Department=civil
CollegeName=mvc

```

```

EventName=papertalk
RegistrationFee=500.0
Enter the event to search
Number of participants for ROBOCAR event is 2]" Actual output:"[Enter the number of
entries
Enter the Participant Name/Yearofstudy/Department/CollegeName/EventName
Print participant details
Participant [name=philip
yearofstudy=4
department=eee
collegeName=mvc
eventName=robocar
registrationFee=1000.0]
Participant [name=susan
yearofstudy=4
department=eee
collegeName=mvc
eventName=robocar
registrationFee=1000.0]
Participant [name=vivek
yearofstudy=3
department=civil
collegeName=mvc
eventName=quiz
registrationFee=300.0]
Participant [name=vishal
yearofstudy=3
department=civil
collegeName=mvc
eventName=papertalk
registrationFee=500.0]
Enter the event to search
No participant found]"$
Check your code with the input :philip/4/eee/mvc/robocar
susan/4/eee/mvc/robocar
vivek/3/civil/mvc/quiz
vishal/3/civil/mvc/papertalk
robocar

```

Obtained Pass Percentage. Still few testcases failed . Kindly revisit the Solution

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

=====

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)	<p>This method should take the amount to be withdrawn as an argument.</p> <p>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.</p>

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
2 public class Call {
3     private int callId;
4     private long calledNumber;
5     private float duration;
6     public Call(){
7     }
8     public int getCallId(){
9         return callId;
10    }
11    public long getCalledNumber(){
12        return calledNumber;
13    }
14    public float getDuration(){
15        return duration;
16    }
17    public void setCallId(int callId){
18        this.callId=callId;
19    }
20    public void setCalledNumber(long calledNumber){
21        this.calledNumber=calledNumber;
22    }
23    public void setDuration(float duration){
24        this.duration=duration;
25    }
26    public void parseData(String calld){
27        callId=Integer.parseInt(calld.split(":")[0]);
28        setCallId(callId);
29        calledNumber=Long.parseLong(calld.split(":")[1]);
30        setCalledNumber(calledNumber);
31        duration=Float.parseFloat(calld.split(":")[2]);
32        setDuration(duration);
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        Main obj=new Main();
14        int ra=obj.rvs(a);
15        int rb=obj.rvs(b);
16        if(a*b==ra*rb){
17            System.out.println(a+" and "+b+" are correct pair");
18        }
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 tempIp;
11         while(memberCount>0){
12             System.out.println("Enter the member details");
13             tempIp = scan.next();
14             String tempArr[] = tempIp.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
-------------	-----------	---------	-----------------

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

```

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 =
DriverManager.getConnection(props.getProperty("DB_URL"),props.getProperty("DB_USERNAME"),props.getPr
operty("DB_PASSWORD"));
27         }
28         catch(IOException e){
29             e.printStackTrace();
30         }
31         return con;
32     }
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());
}
}
```

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.c
ts.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 incorrect;

    }

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

        String[] actualmethods = methodWithExcpn.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);

    }

}

```


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 methodWithExcpn, String className) {

    Class cls = null;

    try {

        String[] actualmethods = methodWithExcpn.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");
}

}
}

```

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 setItemQuantity(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 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;
}
}

```

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;

}

}

```

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(propertDetails) 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 validatePropertyParams(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 incorrect = false;
```

```
        try {
```

```
            Class.forName(className);
```

```
            incorrect = true;
```

```

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

    } catch (ClassNotFoundException e) {

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

    } catch (Exception e) {

        LOG.log(Level.SEVERE,

            "There is an error in validating the " + "Class Name. Please manually verify that the "

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

    }

    return incorrect;

}

```

```

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

    Class cls = null;

    try {

        String[] actualmethods = methodWithExcpn.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");

}

}

}

```

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

}

}

```


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();

}

}

```

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++;

            }

        }

    }

}
```

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("Decrypted Text:\n"+decryptMessage);  
}  
}
```

Account.java*

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

```

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

```

Main.java*

```

import java.util.Scanner;

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

```

THE NEXT RECHARGE DATE

Main.java*

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

public class Main {

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

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

    public static boolean isValidFormat(String date){
        String regex="^(3[01]|[12][0-9]|0[1-9])/(1[0-2]|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 sdformat=new SimpleDateFormat("dd/MM/yyyy");

    Date d1=sdformat.parse(date1);

    Date d2=sdformat.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 sdformat=new SimpleDateFormat("dd/MM/yyyy");

    try{

        Date mydate=sdformat.parse(date);

        c.setTime(mydate);

        c.add(Calendar.DATE, days);

    }catch(ParseException e){

        e.printStackTrace();

    }

    String toDate=sdformat.format(c.getTime());

    return toDate;

}

}

```

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 validateCusomerId(){

        // 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 validateCusomerId(){

        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 validateCusomerId(){
    // 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.validateCusomerId()){
    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.validateCusomerId()){
    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.validateCusomerId()){
    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;
                }
            }
        }
    }
}
```



```

        // 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= $(150+50+50)/3=83.33333333333333$

so this method should return 83.33333333333333

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;

    }

    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]);

    }

}

}

```

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 methodWithExcpn, String className) {

    Class cls = null;

    try {

        String[] actualmethods = methodWithExcpn.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.getPropert
y("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");

    }

    }while(true);

}

}

```

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] | [1-2][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");
}
}

```

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.LocalDateTime;
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 rowCount = 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 FileUtil {

    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.LocalDateTime;

```

```

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 funtionalClass = "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(funtionalClass, 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(funtionalClass, "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);

        }

    }

}
```

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());
    }

}

}

```

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);
    }
}
```

```
}
```

```
}
```

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());  
  
}  
}
```

Retrive 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());
}

}
}

```

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("DB_URL"),props.getProperty("DB_USERNAME"),props.getPropert
y("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";

        }

    }

    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");
        }
    }

}
```

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-1))
                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.calculateTotalBill1(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.calculateTotalBill1(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");
}
}
```

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 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);
            }
        }
    }
}
```

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 succesfully via cheque";
        }
        else if(obj instanceof Cash ) {
            Cash cash = (Cash) obj;
            if(cash.payAmount())
                message = "Payment done succesfully via cash";
        }
        else if(obj instanceof Credit ) {
            Credit card = (Credit) obj;
            if(card.payAmount())
                message = "Payment done succesfully via creditcard. Remainig 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();
}
}

```

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[]=new int[9];
        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[]=new int[10];
        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));
            }
        }
    }
}
```

```

    }
    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 Inavlid 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]);
        }
    }
}
```

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;
        }
        return sum;
    }
}
```

```

        }
        n=n/10;
    }
    if(sum%2==0)
    {
        r=-1;
    }
    else
    {
        r=1;
    }
    return r;
}
}

```

2. Number Validation

Write a program to read a string of 10 digit number , check whether the string contains a 10 digit number in the format XXX-XXX-XXXX where 'X' is a digit.

Include a class **UserMainCode** with a static method **validateNumber** which accepts a string as input .

The return type of the output should be 1 if the string meets the above specified format . In case the number does not meet the specified format then return -1 as output.

Create a class **Main** which would get the input as a String of numbers and call the static method **validateNumber** present in the UserMainCode.

Input and Output Format:

Input consists of a string.

Output is a string specifying the given string is valid or not .

Refer sample output for formatting specifications.

Sample Input 1:

123-456-7895

Sample Output 1:

Valid number format

Sample Input 2:

-123-12344322

Sample Output 2:

Invalid number format

MAIN:

```

import java.util.*;
public class Main {
    public static void main(String[] args)
    {
        Scanner s=new Scanner(System.in);
        String number=s.next();
        int r=UserMainCode.validateNumber(number);
        if(r==1)
        {
            System.out.println("Valid number format");
        }
    }
}

```

```

        else
        {
            System.out.println("Invalid number format");
        }
        s.close();
    }
}

```

USERMAINCODE:

```

import java.util.*;
public class UserMainCode {
    public static int validateNumber(String number)
    {
        int b;
        if(number.matches("[0-9]{3}[-]{1}[0-9]{3}[-]{1}[0-9]{4}"))
        {
            b=1;
        }
        else
        {
            b=0;
        }
        return b;
    }
}

```

3. Sum of Squares of Even Digits

Write a program to read a number , calculate the sum of squares of even digits (values) present in the given number.

Include a class **UserMainCode** with a static method **sumOfSquaresOfEvenDigits** which

accepts a positive integer . The return type (integer) should be the sum of squares of the even digits.

Create a class **Main** which would get the input as a positive integer and call the static method **sumOfSquaresOfEvenDigits** present in the **UserMainCode**.

Input and Output Format:

Input consists of a positive integer n.

Output is a single integer .

Refer sample output for formatting specifications.

Sample Input 1:

56895

Sample Output 1:

100

MAIN:

```

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

```

```

        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        System.out.println(UserMainCode.sumOfSquaresOfEvenDigits(n));
        s.close();
    }
}

```

USERMAINCODE:

```

public class UserMainCode {
    public static int sumOfSquaresOfEvenDigits(int n)
    {
        int n1=0;
        int sum=0;
        while(n!=0)
        {
            n1=n%10;
            if(n1%2==0)
            {
                sum+=n1*n1;
            }
            n=n/10;
        }
        return sum;
    }
}

```

4. Fetching Middle Characters from String

Write a program to read a string of even length and to fetch two middle most characters from the input string and return it as string output.

Include a class **UserMainCode** with a static method **getMiddleChars** which accepts a string

of even length as input . The return type is a string which should be the middle characters of the string.

Create a class **Main** which would get the input as a string and call the static method **getMiddleChars** present in the UserMainCode.

Input and Output Format:

Input consists of a string of even length.

Output is a string .

Refer sample output for formatting specifications.

Sample Input 1:

this

Sample Output 1:

hi

Sample Input 1:

Hell

Sample Output 1:

el

MAIN:

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

USERMAINCODE:

```
import java.util.*;
public class UserMainCode {
    public static String getMiddleChars(String str)
    {
        StringBuffer sb=new StringBuffer();
        if(str.length()%2==0)
        {
            sb.append(str.substring((str.length())/2-1,(str.length())/2+1));
        }
        return sb.toString();
    }
}
```

5. Check Characters in a String

Write a program to read a string and to test whether first and last character are same. The string is said to be valid if the 1st and last character are the same. Else the string is said to be invalid.

Include a class **UserMainCode** with a static method **checkCharacters** which accepts a string as input .

The return type of this method is an int. Output should be 1 if the first character and last character are same . If they are different then return -1 as output.

Create a class **Main** which would get the input as a string and call the static method **checkCharacters** present in the UserMainCode.

Input and Output Format:

Input consists of a string.

Output is a string saying characters are same or not .

Refer sample output for formatting specifications.

Sample Input 1:

the picture was great

Sample Output 1:

Valid

Sample Input 1:

this

Sample Output 1:

Invalid

MAIN:

```
import java.util.*;
public class main {
    public static void main(String[] args)
    {
        Scanner s=new Scanner(System.in);
        String input=s.nextLine();
        int r=UserMainCode.checkCharacters(input);
        if(r==1)
        {
            System.out.println("Valid");
        }
        else
        {
            System.out.println("Invalid");
        }
        s.close();
    }
}
```

USERMAINCODE:

```
import java.util.*;
public class UserMainCode {
    public static int checkCharacters(String input)
    {
        int r;
        StringTokenizer t = new StringTokenizer(input," ");
        String s = t.nextToken();
        String s1 =s ;
        while(t.hasMoreTokens())
        {
            s1 = t.nextToken();
        }
        if(s.charAt(0) == s1.charAt(s1.length()-1))
            r=1;
        else
            r=0;
        return r;
    }
}
```

6. Forming New Word from a String

Write a program to read a string and a positive integer n as input and construct a string with first n and last n characters in the given string.

Include a class **UserMainCode** with a static method **formNewWord** which accepts a string and positive integer .

The return type of the output should be a string (value) of first n character and last n character.

Create a class **Main** which would get the input as a string and integer n and call the static method **formNewWord** present in the UserMainCode.

Input and Output Format:

Input consists of a string of even length.

Output is a string .

Note: The given string length must be $\geq 2n$.

Refer sample output for formatting specifications.

Sample Input 1:

California

3

Sample Output 1:

Calnia

Sample Input 2:

this

1

Sample Output 2:

Ts

MAIN:

```
import java.util.*;
public class Main {
    public static void main(String[] args)
    {
        Scanner s=new Scanner(System.in);
        String s1=s.nextLine();
        int n1=s.nextInt();
        System.out.println(UserMainCode.formNewWord(s1,n1));
        s.close();
    }
}
```

USERMAINCODE:

```
import java.util.*;
public class UserMainCode {
    public static String formNewWord(String s1,int n)
    {
        String s = new String();
        if(s1.length()>n)
        {
            s = s1.substring(0,n) + s1.substring(s1.length()-n, s1.length());
            return s;
        }
        else
            return null;
    }
}
```

7. Reversing a Number

Write a program to read a positive number as input and to get the reverse of the given number and return it as output.

Include a class **UserMainCode** with a static method **reverseNumber** which accepts a positive integer .

The return type is an integer value which is the reverse of the given number.

Create a **Main** class which gets the input as a integer and call the static method **reverseNumber** present in the **UserMainCode**

Input and Output Format:

Input consists of a positive integer.

Output is an integer .

Refer sample output for formatting specifications.

Sample Input 1:

543

Sample Output 1:

345

Sample Input 1:

1111

Sample Output 1:

1111

MAIN:

```
import java.util.*;
public class Main {
    public static void main(String[] args)
    {
        Scanner s=new Scanner(System.in);
        int n=s.nextInt();
        System.out.println(UserMainCode.reverseNumber(n));
        s.close();
    }
}
```

USERMAINCODE:

```
public class UserMainCode {
    public static int reverseNumber(int n)
    {
        int a,r=0;
        while(n!=0)
        {
            a=n%10;
            r=r*10+a;
            n=n/10;
        }
        return r;
    }
}
```

8. Array List Sorting and Merging

Write a code to read two int array lists of size 5 each as input and to merge the two arrayLists, sort the merged arraylist in ascending order and fetch the elements at 2nd, 6th and 8th index into a new arrayList and return the final ArrayList.

Include a class **UserMainCode** with a static method **sortMergedArrayList** which accepts 2

ArrayLists.

The return type is an ArrayList with elements from 2,6 and 8th index position .Array index starts from position 0.

Create a **Main** class which gets two array list of size 5 as input and call the static method **sortMergedArrayList** present in the **UserMainCode**.

Input and Output Format:

Input consists of two array lists of size 5.

Output is an array list .

Note - The first element is at index 0.

Refer sample output for formatting specifications.

Sample Input 1:

3
1
17
11
19
5
2
7
6
20

Sample Output 1:

3
11
19

Sample Input 2:

1
2
3
4
5
6
7
8
9
10

Sample Output 2:

3
7
9

Main:

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

```

{
    Scanner s=new Scanner(System.in);
    ArrayList<Integer> list1=new ArrayList<Integer>();
    ArrayList<Integer> list2=new ArrayList<Integer>();
    ArrayList<Integer> newList=new ArrayList<Integer>();
    for (int i = 0; i < 5; i++)
    {
        list1.add(s.nextInt());
    }
    for (int i = 0; i < 5; i++)
    {
        list2.add(s.nextInt());
    }
    newList=UserMainCode.sortMergedArrayList(list1,list2);
    for (int i = 0; i < 3; i++)
    {
        System.out.println(newList.get(i));
    }
    s.close();
}
}

```

USERMAINCODE:

```

import java.util.*;
public class UserMainCode {
    public static ArrayList<Integer> sortMergedArraylist(ArrayList<Integer>
list1,ArrayList<Integer> list2)
    {
        list1.addAll(list2);
        Collections.sort(list1);
        ArrayList<Integer> ans=new ArrayList<Integer>();
        ans.add(list1.get(2));
        ans.add(list1.get(6));
        ans.add(list1.get(8));
        return ans;
    }
}

```

9. Validating Date Format

Obtain a date string in the format dd/mm/yyyy. Write code to validate the given date against the given format.

Include a class **UserMainCode** with a static method **validateDate** which accepts a string .

The return type of the validateDate method is 1 if the given date format matches the specified format , If the validation fails return the output as -1.

Create a **Main** class which gets date string as an input and call the static method **validateDate** present in the **UserMainCode**.

Input and Output Format:

Input is a string .

Refer sample output for formatting specifications

Sample Input 1:

12/06/1987

Sample Output 1:

Valid date format

Sample Input 2:

03/1/1987

Sample Output 2:

Invalid date format

Main:

```
import java.util.*;
public class Main {
public static void main(String[] args) {
Scanner sc=new Scanner(System.in);
String s1=sc.nextLine();
int b=UserMainCode.ValidateDate(s1);
if(b==1){
    System.out.println("Valid date format");
}
else{
    System.out.println("Invalid date format");
}
sc.close();
}}
```

UserMainCode:

```
import java.util.*;
import java.text.*;
public class UserMainCode{
public static int ValidateDate(String s1) {
if(s1.matches("[0-9]{2}[/]{1}[0-9]{2}[/]{1}[0-9]{4}"))
{
SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");
sdf.setLenient(false);
try {
Date d1=sdf.parse(s1);
return 1;
} catch (ParseException e) {
return -1;
}
}
else{
return -1;}}}
```

10. Validate Time

Obtain a time string as input in the following format 'hh:mm am' or 'hh:mm pm'. Write code to validate it using the following rules:

- It should be a valid time in 12 hrs format
- It should have case insensitive AM or PM

Include a class **UserMainCode** with a static method **validateTime** which accepts a string. If the given time is as per the given rules then return 1 else return -1.If the value returned is 1 then print as valid time else print as Invalid time.

Create a **Main** class which gets time(string value) as an input and call the static method **validateTime** present in the **UserMainCode**.

Input and Output Format:

Input is a string .

Output is a string .

Sample Input 1:

09:59 pm

Sample Output 1:

Valid time

Sample Input 2:

10:70 AM

Sample Output 2:

Invalid time

Main:

```
import java.util.*;
public class Main{
public static void main(String []args){
Scanner sc=new Scanner(System.in);
String str=sc.nextLine();
int b=UserMainCode.ValidateTime(str);
if(b==1){
    System.out.println("Valid time");
}
else{
    System.out.println("Invalid time");
}
sc.close();
}}
```

UserMainCode:

```
import java.text.*;
import java.util.*;
public class UserMainCode{
public static int ValidateTime(String str){
StringTokenizer st=new StringTokenizer(str,".");
if(st.countTokens()==3)
{
SimpleDateFormat sdf1 = new SimpleDateFormat("h:mm:ss a");
sdf1.setLenient(false);
try
{
Date d2=sdf1.parse(str);
return 1;
}
catch(Exception e)
{
return -1;
}
}
else
{
SimpleDateFormat sdf = new SimpleDateFormat("h:mm a");
sdf.setLenient(false);
try
{
Date d1=sdf.parse(str);
return 1;
}
}
```

```

catch(Exception e){
    return -1;
}}}}

```

11. String Encryption

Given an input as string and write code to encrypt the given string using following rules and return the encrypted string:

1. Replace the characters at odd positions by next character in alphabet.
2. Leave the characters at even positions unchanged.

Note:

- If an odd position character is 'z' replace it by 'a'.
- Assume the first character in the string is at position 1.

Include a class **UserMainCode** with a static method **encrypt** which accepts a string.

The return type of the output is the encrypted string.

Create a **Main** class which gets string as an input and call the static method **encrypt** present in the **UserMainCode**.

Input and Output Format:

Input is a string .
Output is a string.

Sample Input 1:
curiosity

Sample Output 1:
dusipsjtz

Sample Input 2:
zzzz

Sample Output 2:
Azaz

Main:

```

import java.util.*;
public class Main {
public static void main(String[] args) {
    Scanner s=new Scanner(System.in);
    String s1=s.next();
    System.out.println(UserMainCode.encrypt(s1));
    s.close();
}
}

```

UserMainCode:

```

public class UserMainCode{
public static String encrypt(String s1) {
StringBuffer sb=new StringBuffer();
for(int i=0;i<s1.length();i++){
char c=s1.charAt(i);
if(i%2==0){
if(c==122)
if((c==122)&&(i==0)){
c='A';}
else
c=(char) (c-25);
else{

```

```

c=(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:

cmmitmnt

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[]=new int[n];
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[]=new int[n];
        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 position 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 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

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 $nCr = \frac{n!}{r!(n-r)!}$ where first input being n and second input being r.

Note1 : n! factorial can be achieved using given formula $n! = n \times (n-1) \times (n-2) \times \dots \times 2 \times 1$.

Note2 : $0! = 1$.

Example $5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$

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 to get a password as string input and validate using the rules specified below.

Apply following validations:

1. Minimum length should be 8 characters
2. Must contain any one of these three special characters @ or _ or #
3. May contain numbers or alphabets.
4. Should not start with special character or number
5. Should not end with special character

Include a class **UserMainCode** with a static method **validatePassword** which accepts password string as input and returns an integer. The method returns 1 if the password is

valid. Else it returns -1.

Create a class **Main** which would get the input and call the static method **validatePassword** present in the UserMainCode.

Input and Output Format:

Input consists of a string.

Output is a string Valid or Invalid.

Refer sample output for formatting specifications.

Sample Input 1:

ashok_23

Sample Output 1:

Valid

Sample Input 2:

1980_200

Sample Output 2:

Invalid

Main

```
import java.util.*;

public class Main{

    public static void main(String[] args)

    {

        Scanner sc=new Scanner(System.in);

        String a=sc.next();

        int e=UserMainCode.validatePassword(a);

        if(e==1){

            System.out.println("Valid");

        }

    }

}
```

```

else
{
    System.out.println("Invalid");
}
sc.close();
}}

```

UserMainCode

```

public class UserMainCode {

    public static int validatePassword(String a){

        int d=0;

        if(a.length()>=8){

            if(a.contains("#") || a.contains("@") || a.contains("_"))

            {

                char c= a.charAt(0);

                //System.out.println(c);

                if(Character.isAlphabetic(c))

                {

                    char dd=a.charAt(a.length()-1);

                    //System.out.println(dd);

                    if((Character.isAlphabetic(dd))||(Character.isDigit(dd)))

                    {

                        if(a.matches(".*[0-9]{1,}.*")||a.matches(".*[a-zA-Z]{1,}.*")){

                            d=1;

                        }

                    }

                }

            }

        }

    }
}

```

```

    }

    }

    else

    d=-1;

    return d;

}}

```

28.iD Validation

Write a program to get two string inputs and validate the ID as per the specified format.

Include a class **UserMainCode** with a static method **validateIDLocations** which accepts two

strings as input.

The return type of the output is a string Valid Id or Invalid Id.

Create a class **Main** which would get the input and call the static

method **validateIDLocations** present in the UserMainCode.

Input and Output Format:

Input consists of two strings.

First string is ID and second string is location. ID is in the format CTS-LLL-XXXX where LLL is

the first three letters of given location and XXXX is a four digit number.

Output is a string Valid id or Invalid id.

Refer sample output for formatting specifications.

Sample Input 1:

CTS-hyd-1234

hyderabad

Sample Output 1:

Valid id

Sample Input 2:

CTS-hyd-123

hyderabad

Sample Output 2:

Invalid id

Main

```
import java.util.*;

public class Main3 {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        String s1=sc.next();

        String s2=sc.next();

        boolean b=UserMainCode3.validateIDLocations(s1,s2);

        if(b==true)

            System.out.println("Valid id");

        else

            System.out.println("Invalid id");

        sc.close();
    }
}
```



```
}  
}
```

-

UserMainCode

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

```
public class UserMainCode3 {  
    public static boolean validateIDLocations(String s1, String s2) {  
        String s3=s2.substring(0, 3);  
        boolean b=false;  
        StringTokenizer t=new StringTokenizer(s1,"-");  
        String s4=t.nextToken();  
        String s5=t.nextToken();  
        String s6=t.nextToken();  
        if(s4.equals("CTS") && s5.equals(s3) && s6.matches("[0-9]{4}"))  
            b=true;  
        else{  
            b=false;}  
        return b;  
    }  
}
```

29.Remove Elements

Write a program to remove all the elements of the given length and return the size of the final array as output. If there is no element of the given length, return the size of the same

array as output.

Include a class **UserMainCode** with a static method **removeElements** which accepts a string

array, the number of elements in the array and an integer. The return type (integer) should return the size of the final array as output.

Create a Class Main which would be used to accept Input String array and a number and call the static method present in UserMainCode.

Assume maximum length of array is 20.

Input and Output Format:

Input consists of a integers that corresponds to n, followed by n strings and finally m which corresponds to the length value.

Output consists of a single Integer.

Refer sample output for formatting specifications.

Sample Input 1:

```
5
a
bb
b
ccc
ddd
2
```

Sample Output 1:

```
4
```

Main

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

```

public class Main
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        int n=Integer.parseInt(sc.nextLine());
        String[] a=new String[n];
        for(int i=0;i<n;i++)
        a[i]=sc.nextLine();
        int m=Integer.parseInt(sc.nextLine());
        System.out.println(UserMainCode.removeElements(a,m));
        sc.close();
    }
}

```

UserMainCode

```

public class UserMainCode {

    public static int removeElements(String[] a,int m){

        int u=a.length;

        for(int i=0;i<a.length;i++)
        {
            if(a[i].length()==m)
                u--;
        }

        return u;
    }
}

```

30.Find the difference between Dates in months

Given a method with two date strings in yyyy-mm-dd format as input. Write code to find the difference between two dates in months.

Include a class **UserMainCode** with a static method **getMonthDifference** which accepts two

date strings as input.

The return type of the output is an integer which returns the difference between two dates in months.

Create a class **Main** which would get the input and call the static method **getMonthDifference** present in the UserMainCode.

Input and Output Format:

Input consists of two date strings.

Format of date : yyyy-mm-dd.

Output is an integer.

Refer sample output for formatting specifications.

Sample Input 1:

2012-03-01

2012-04-16

Sample Output 1:

1

Sample Input 2:

2011-03-01

2012-04-16

Sample Output 2:

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[]=new int[n];

        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 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
```

Main

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

aaaxxyzAAx

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.consecutiveRepetitionOfChar(input1));
}
}

```

```
}  
}
```

UserMainCode

```
class UserMainCode  
{  
  
    public static int consecutiveRepeatitionOfChar(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

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 **getFormatedString** which accepts a string. The return type (String) should return the shrunked 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.getFormattedString(s1));
    }
}
```

UserMainCode

```
class UserMainCode
{

    public static String getFormattedString(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:

Cognizantantantant

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("EEEEEE");
        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=11*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 average 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' && 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}[/]{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

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 :

6

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 Searchtring. 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 Input 2:

265-65-7777

Sample Output 1:

Valid

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 argument 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("[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 Input 2:

4534

Sample Output 1:

2

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:

XXxxtest

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>=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));
    }
}

```

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[]=new String[n];
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(x1,y1,x2,y2) as the coordinates of two points.

Calculate the distance between the two points using the below formula.

Formula : square root of((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2))

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:

```
Llollollo
```

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[]=new int[20];
        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.

Sample Input :

This is a cat

Sample Output :

S\$\$SA\$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(UserMainode.getMaxKeyValue(hm));
    }
}
```

Hashmap:

```
import java.util.*;
import java.util.HashMap;
import java.util.Iterator;
public class UserMainode {
    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);
            }
        }
    }
}
```

```

        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 inplst=sc.nextLine();
        System.out.println(UserMainCode.calculateWordSum(inplst));
    }

}

```

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 should 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> all=new ArrayList<String>();
        all=UserMainCode.getName(hm1);
        Iterator<String> it=all.iterator();
        while(it.hasNext())
        {
```



```

        System.out.println(it.next());
    }
}
}

```

Usermaincode:

```

import java.util.*;
import java.text.*;
public class UserMainCode {
    public static ArrayList<String> getName(HashMap<Integer,String> hm1)
    {
        ArrayList<String> al2=new ArrayList<String>();
        Iterator<Integer> it =hm1.keySet().iterator();
        while(it.hasNext())
        {
            int id=it.next();
            String name=hm1.get(id);
            if(name.matches("[a-z]{1,}.*[0-9]{1,}.*[A-Z]{1}"))
                al2.add(name);
        }
        return al2;
    }
}

```

19.Max Admissions

Write a program that reads details about number of admissions per year of a particular college, return the year which had maximum admissions. The details are stored in an arraylist with the first index being year and next being admissions count.

Include a class UserMainCode with a static method **getYear** which accepts a arraylist. The return type is an integer indicating the year of max admissions.

Create a Class Main which would be used to accept Input string and call the static method present in UserMainCode.

Input and Output Format:

Input consists of 2n+1 values. The first value corresponds to size of the data (year & admissions). The next n pair of numbers contains the year and admissions count.

Output consists of an integer as mentioned in the problem statement.

Refer sample output for formatting specifications.

Sample Input 1:

```

4
2010
200000
2011
300000
2012
45000
2013
25000

```

Sample Output 1:

2011

USERMAINCODE:

```
import java.util.ArrayList;

public class UserMainCode
{
    public static int year (ArrayList<Integer> a1)
    {
        int max=0,pos=0;
        for(int i=1;i<a1.size();i+=2)
        {
            if(a1.get(i)>max)
            {
                max=a1.get(i);
                pos=i;
            }
        }
        return a1.get(pos-1);
    }
}

MAIN:

import java.util.*;

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

        ArrayList<Integer> a1=new ArrayList<Integer>();
```

```

int n=s.nextInt();

n=n*2;

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

{

a1.add(s.nextInt());

}

System.out.println(UserMainCode.year(a1));

s.close();

}

}

```

20.Sum Non Prime Numbers

Write a program to calculate the sum of all the non prime positive numbers less than or equal to the given number.

Note: prime is a natural number greater than 1 that has no positive divisors other than 1 and itself

Example:

input = 9

Prime numbers = 2,3,5 and 7

output = 1+4+6+8+9=28

Include a class **UserMainCode** with a static method “**addNumbers**” that accepts an integer

argument and returns an integer.

Create a class **Main** which would get an integer as input and call the static method **validateNumber** present in the UserMainCode.

Input and Output Format:

Input consists of an integer.

Output consists of an integer.

Sample Input:

9

Sample Output:

28

Main:

```

import java.util.*;

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

```

```

    }
}

Usermaincode:

public class UserMainCode {
    public static int addNumbers(int n) {
        int sum=0;int k=0;int sum1=0;
        for(int i=1; i<=n; i++)
        { k=0;
        for(int j=1; j<=i; j++)
        {
            if(i%j==0)
            k++;
        }
        if(k!=2)
        {
            sum=sum+i;
        }
        }
        return sum;
    }
}

```

21.Date Format Conversion

Given a date string in the format dd/mm/yyyy, write a program to convert the given date to the format dd-mm-yy.

Include a class **UserMainCode** with a static method “**convertDateFormat**” that accepts a

String and returns a String.

Create a class **Main** which would get a String as input and call the static method **convertDateFormat** present in the UserMainCode.

Input and Output Format:

Input consists of a String.

Output consists of a String.

Sample Input:

12/11/1998

Sample Output:

12-11-98

Main:

```

import java.util.*;
import java.text.*;
public class Main {
    public static void main(String[] args) {
        Scanner s=new Scanner(System.in);
        String s1=s.next();
        UserMainCode.convertDateFormate(s1);
    }
}

```

Usermaincode:

```

import java.util.*;
import java.text.*;
public class UserMainCode {
    public static void convertDateFormat(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}/[0-9]{2}/[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}-[0-9]{2}-[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 Occurances - 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 **checkTripplets** 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.checkTripplets(a);

        if(s==true)

            System.out.println("TRUE");

        else

            System.out.println("FALSE");

    }

}
```

UserMainCode:-

```
import java.util.*;

public class UserMainCode {

    public static boolean checkTripplets(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[] arg)
    {
        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 pg.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.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[]=new int[n];
```

```

    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:

2

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 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.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

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 $1+2+5$ 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

```
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:Conside 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[]=new 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 **getStateld** 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 `getStateId` returns the `HashMap` to the Main Class.

Include `UserMainCode` Class With static method **`getStateId`** 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

```

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 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

```

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.getTheTotalCostOfPheripherals
(m1,s));
    }
}
```

USERMAINCODE:

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

public class UserMainCode {
    public static float getTheTotalCostOfPheripherals(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[]=new int[n];
        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;}
            }
        return s2;
    }
}
}
}

```

5. String Occurences

Obtain two strings from user as input. Your program should count the number of occurences 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 **countNoOfWords** 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>a11,ArrayList<Integer>a12)

    {

        ArrayList<Integer>a13=new ArrayList<Integer>();

        for(int i=0;i<a11.size();i++)

        {

            if(i%2==0)

                a13.add(a12.get(i));

            else

                a13.add(a11.get(i));

        }

        return a13;

    }

}
```

```
}
```

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>();

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 calculated 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();
    }
}

```

```

        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;
}

}

```

31.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

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.next();

    }

    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[]=new String[n];

        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 n2 = sc.nextInt();

    String[] s2 = new String[n2];

    for (int i = 0; i < n2; i++) {

        s2[i] = sc.next();

    }

    System.out.println(UserMainCode.countCommonStrings(s1,s2));

}

}

```

USERMAINCODE:

```

import java.util.ArrayList;

public class UserMainCode {

    public static int countCommonStrings(String[] s1,String[] s2)
    {

        int count=0;

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

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

            for (int j = 0; j < s2.length; j++) {

                if(s1[i].equals(s2[j])){

                    if(!al.contains(s1[i])){

                        count++;

                        al.add(s1[i]);

                    }

                }

            }

        }

    }

}

```

```

        }
    }
}

return count;
}

}

```

35.Sequence Sum

Write a program to read a non-negative integer n, and find sum of fibonacci 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 Fibonacci 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:

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.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;
```



```

        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.next();

        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. $[(A \cup B) - (A \cap B)]$

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 $B = \{11, 26, 7, 14\}$.

$A \cup B = \{7, 11, 12, 14, 24, 26, 36\}$ and

$A \cap B = \{7, 14\}$

Symmetric difference of A and B after sorting = $[A \cup B] - [A \cap B] = \{11, 12, 24, 26, 36\}$.

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

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 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 5 year then discount rate is 10% of transaction amount.

c. If the transaction amount less than 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 20000 and registration less than 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++)
```

```

{
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=t3/1;

//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;

}

}

```

