

## ZeeZee Bank

Account.java

```
public class Account {  
    private long accountNumber;  
    private double balanceAmount;  
  
    public Account(long accountNumber, double balanceAmount) {  
        this.accountNumber = accountNumber;  
        this.balanceAmount = balanceAmount;  
    }  
  
    public long getAccountNumber() {  
        return accountNumber;  
    }  
  
    public void setAccountNumber(long accountNumber) {  
        this.accountNumber = accountNumber;  
    }  
  
    public double getBalanceAmount() {  
        return balanceAmount;  
    }  
  
    public void setBalanceAmount(double balanceAmount) {  
        this.balanceAmount = balanceAmount;  
    }  
  
    public void deposit(double depositAmount) {  
        balanceAmount += depositAmount;  
    }  
  
    public boolean withdraw(double withdrawAmount) {  
        if (withdrawAmount <= balanceAmount) {  
            balanceAmount -= withdrawAmount;  
            return true;  
        }  
        return false;  
    }  
}
```

Main.java

```
import java.text.DecimalFormat;
```

```

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        DecimalFormat decimalFormat = new DecimalFormat("0.00");

        System.out.println("Enter the account number:");
        long accountNumber = scanner.nextLong();

        System.out.println("Enter initial balance:");
        double balanceAmount = scanner.nextDouble();

        Account account = new Account(accountNumber, balanceAmount);

        System.out.println("Enter the amount to be deposited:");
        double depositAmount = scanner.nextDouble();
        account.deposit(depositAmount);
        double availableBalance = account.getBalanceAmount();

        System.out.println("Available balance is:" + decimalFormat.format(availableBalance));

        System.out.println("Enter the amount to be withdrawn:");
        double withdrawAmount = scanner.nextDouble();
        boolean isWithdrawn = account.withdraw(withdrawAmount);
        availableBalance = account.getBalanceAmount();

        if (!isWithdrawn) {
            System.out.println("Insufficient balance");
        }

        System.out.println("Available balance is:" + decimalFormat.format(availableBalance));
    }
}

```

### Numerology number

```

Main.java
import java.util.Scanner;

public class Main {
    private static int getSum(long num) {
        char[] chars = Long.toString(num).toCharArray();

```

```
int sum = 0;

for (char ch : chars) {
    sum += Character.digit(ch, 10);
}

return sum;
}

private static int getNumerology(long num) {
    String string = String.valueOf(num);

    while (string.length() != 1) {
        string = String.valueOf(getSum(Long.parseLong(string)));
    }

    return Integer.parseInt(string);
}

private static int getOddCount(long num) {
    int oddCount = 0;

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

    return oddCount;
}

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

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

    return evenCount;
}

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

```

Scanner scanner = new Scanner(System.in);

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

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

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

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

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

```

### **Substitution Cipher Technique**

```

Main.java
import java.util.*;

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

        System.out.println("Enter the encrypted text:");
        String text = scanner.nextLine();
        char[] chars = text.toCharArray();
        boolean flag = false;

        for (char ch : chars) {
            if (Character.isLetter(ch)) {
                flag = true;

                if (Character.isLowerCase(ch)) {
                    int sub = (int) ch - 7;

                    if (sub < 97) {
                        ch = (char) (122 - (97 - sub) + 1);
                    } else {

```

```

        ch = (char) sub;
    }
} else if (Character.isUpperCase(ch)) {
    int sub = (int) ch - 7;

    if (sub < 65) {
        ch = (char) (90 - (65 - sub) + 1);
    } else {
        ch = (char) sub;
    }
}

stringBuilder.append(ch);
} else if (Character.isWhitespace(ch)) {
    stringBuilder.append(ch);
}
}

if (flag) {
    System.out.println("Decrypted text:");
    System.out.println(stringBuilder.toString());
} else {
    System.out.println("No hidden message");
}
}
}
}

```

### Bank Account - Interface

Account.java

```

public class Account {
    private String accountNumber;
    private String customerName;
    private double balance;

    public Account(String accountNumber, String customerName, double balance) {
        this.accountNumber = accountNumber;
        this.customerName = customerName;
        this.balance = balance;
    }

    public String getAccountNumber() {
        return accountNumber;
    }
}

```

```
public void setAccountNumber(String accountNumber) {
    this.accountNumber = accountNumber;
}

public String getCustomerName() {
    return customerName;
}

public void setCustomerName(String customerName) {
    this.customerName = customerName;
}

public double getBalance() {
    return balance;
}

public void setBalance(double balance) {
    this.balance = balance;
}
}
```

#### CurrentAccount.java

```
public class CurrentAccount extends Account implements MaintenanceCharge {
    public CurrentAccount(String accountNumber, String customerName, double balance) {
        super(accountNumber, customerName, balance);
    }

    @Override
    public float calculateMaintenanceCharge(float noOfYears) {
        return (100.0f + noOfYears) + 200.0f;
    }
}
```

#### MaintenanceCharge.java

```
public interface MaintenanceCharge {
    float calculateMaintenanceCharge(float noOfYears);
}
```

#### SavingsAccount.java

```
public class SavingsAccount extends Account implements MaintenanceCharge {
    public SavingsAccount(String accountNumber, String customerName, double balance) {
        super(accountNumber, customerName, balance);
    }
}
```

```

@Override
public float calculateMaintenanceCharge(float noOfYears) {
    return (50.0f * noOfYears) + 50.0f;
}
}

UserInterface.java
import java.text.DecimalFormat;
import java.util.Scanner;

public class UserInterface {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        DecimalFormat decimalFormat = new DecimalFormat("0.0");

        System.out.println("1. Savings Account");
        System.out.println("2. Current Account");
        System.out.println("Enter your choice:");
        int choice = scanner.nextInt();

        System.out.println("Enter the Account number");
        String accountNumber = scanner.next();

        System.out.println("Enter the Customer Name");
        String customerName = scanner.next();

        System.out.println("Enter the Balance amount");
        double balance = scanner.nextDouble();

        System.out.println("Enter the number of years");
        int noOfYears = scanner.nextInt();

        System.out.println("Customer Name " + customerName);
        System.out.println("Account Number " + accountNumber);
        System.out.println("Account Balance " + decimalFormat.format(balance));

        switch (choice) {
            case 1: {
                SavingsAccount savingsAccount = new SavingsAccount(accountNumber,
                customerName, balance);
                System.out.println("Maintenance Charge for Savings Account is Rs " +
                decimalFormat.format(savingsAccount.calculateMaintenanceCharge(noOfYears)));
                break;
            }
        }
    }
}

```

```

        case 2: {
            CurrentAccount currentAccount = new CurrentAccount(accountNumber,
customerName, balance);
            System.out.println("Maintenance Charge for Current Account is Rs " +
decimalFormat.format(currentAccount.calculateMaintenanceCharge(noOfYears)));
        }
    }
}
}

```

## **Batting Average**

```

UserInterface.java
package com.ui;

import com.utility.Player;

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

public class UserInterface {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Player player = new Player();
        player.setScoreList(new ArrayList<>());
        boolean flag = true;

        while (flag) {
            System.out.println("1. Add Runs Scored");
            System.out.println("2. Calculate average runs scored");
            System.out.println("3. Exit");
            System.out.println("Enter your choice");
            int choice = scanner.nextInt();

            switch (choice) {
                case 1: {
                    System.out.println("Enter the runs scored");
                    int score = scanner.nextInt();
                    player.addScoreDetails(score);
                    break;
                }
                case 2: {
                    System.out.println("Average runs secured");
                }
            }
        }
    }
}

```

```
        System.out.println(player.getAverageRunScored());
        break;
    }
    case 3: {
        System.out.println("Thank you for use the application");
        flag = false;
        break;
    }
}
}
}
}
```

Player.java

```
package com.utility;
```

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

```
public class Player {
```

```
    private List<Integer> scoreList;
```

```
    public List<Integer> getScoreList() {
```

```
        return scoreList;
```

```
}
```

```
    public void setScoreList(List<Integer> scoreList) {
```

```
        this.scoreList = scoreList;
```

```
}
```

```
    public double getAverageRunScored() {
```

```
        if (scoreList.isEmpty()) {
```

```
            return 0.0;
```

```
}
```

```
        int size = scoreList.size();
```

```
        int totalScore = 0;
```

```
        for (int score : scoreList) {
```

```
            totalScore += score;
```

```
}
```

```
        return (double) totalScore / (double) size;
```

```
}
```

```

    public void addScoreDetails(int score) {
        scoreList.add(score);
    }
}

```

### Grade Calculation

```

Main.java
import java.util.Scanner;

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

        System.out.println("Enter the number of Threads:");
        int n = scanner.nextInt();

        GradeCalculator[] gradeCalculators = new GradeCalculator[n];
        Thread[] threads = new Thread[n];

        for (int i = 0; i < n; ++i) {
            System.out.println("Enter the String:");
            String string = scanner.next();
            String[] strings = string.split(":");
            int[] marks = new int[5];

            String studName = strings[0];

            for (int j = 1; j < 6; ++j) {
                marks[j - 1] = Integer.parseInt(strings[j]);
            }

            gradeCalculators[i] = new GradeCalculator(studName, marks);
            threads[i] = new Thread(gradeCalculators[i]);
            threads[i].start();
            threads[i].interrupt();
        }

        for (int i = 0; i < n; ++i) {
            System.out.println(gradeCalculators[i].getStudName() + ":" +
gradeCalculators[i].getResult());
        }
    }
}

```

```
Gradecalculator.java
public class GradeCalculator extends Thread {
    private String studName;
    private char result;
    private int[] marks;

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

    public String getStudName() {
        return studName;
    }

    public void setStudName(String studName) {
        this.studName = studName;
    }

    public char getResult() {
        return result;
    }

    public void setResult(char result) {
        this.result = result;
    }

    public int[] getMarks() {
        return marks;
    }

    public void setMarks(int[] marks) {
        this.marks = marks;
    }

    @Override
    public void run() {
        int totalMarks = 0;

        for (int mark : marks) {
            totalMarks += mark;
        }
    }
}
```

```

if (totalMarks <= 500 && totalMarks >= 400) {
    result = 'A';
} else if (totalMarks < 400 && totalMarks >= 300) {
    result = 'B';
} else if (totalMarks < 300 && totalMarks >= 200) {
    result = 'C';
} else if (totalMarks < 200 && totalMarks >= 0) {
    result = 'E';
}
}
}

```

### **Employees eligible for promotionCoding exercise**

Main.java

```

import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Scanner;
import java.util.stream.Collectors;
import java.util.stream.IntStream;

class Employee implements Comparable<Employee> {
    private final String id;
    private final LocalDate joiningDate;
    private boolean isEligible;

    public Employee(String id, LocalDate joiningDate) {
        this.id = id;
        this.joiningDate = joiningDate;
    }

    public void setEligible(LocalDate now) {
        isEligible = joiningDate.until(now, ChronoUnit.YEARS) >= 5;
    }

    public boolean getEligible() {
        return isEligible;
    }

    public String getId() {
        return id;
    }
}

```

```

}

@Override
public String toString() {
    return id;
}

@Override
public int compareTo(Employee employee) {
    return this.id.compareTo(employee.getId());
}
}

public class Main {
    public static void main(String[] args) throws Exception {
        Scanner scanner = new Scanner(System.in);
        DateTimeFormatter dateTimeFormatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
        LocalDate now = LocalDate.parse("01/01/2019", dateTimeFormatter);
        int n = scanner.nextInt();
        ArrayList<Employee> employees = new ArrayList<>();

        IntStream.rangeClosed(1, 4).forEach(i -> {
            String id = scanner.next();
            String joiningDateStr = scanner.next();

            try {
                LocalDate joiningDate = LocalDate.parse(joiningDateStr, dateTimeFormatter);
                Employee employee = new Employee(id, joiningDate);
                employee.setIsEligible(now);
                employees.add(employee);
            } catch (Exception ignore) {
                System.out.println("Invalid date format");
                System.exit(0);
            }
        });

        List<Employee> filteredEmployees =
        employees.stream().filter(Employee::getIsEligible).collect(Collectors.toList());

        if (filteredEmployees.isEmpty()) {
            System.out.println("No one is eligible");
        } else {
            Collections.sort(filteredEmployees);
            filteredEmployees.forEach(System.out::println);
        }
    }
}

```

```
    }
}
}
```

### Check Number Type

NumberType.java

```
public interface NumberType {
    boolean checkNumber(int num);
}
```

NumberTypeUtility.java

```
import java.util.Scanner;

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

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

        if (idOdd().checkNumber(num)) {
            System.out.println(num + " is odd");
        } else {
            System.out.println(num + " is not odd");
        }
    }
}
```

### Retrieve Flight details based on source and destination

Main.java

```
import java.util.*;
public class Main{
    public static void main(String[] args){
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter the source");
        sc.next();
        String source=sc.nextLine();
        System.out.println("Enter the destination");
        String dest=sc.nextLine();
        FlightManagementSystem obj=new FlightManagementSystem();
```

```

ArrayList<Flight> res=obj.viewFlightsBySourceDestination(source,dest);
if(res!=null)
    System.out.println(res);
else
    System.out.println("No flights available for the given source and destination");
}
}

```

```

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

public class DB {

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

    //ENSURE YOU DON'T CHANGE THE BELOW CODE WHEN YOU SUBMIT
    public static Connection getConnection() throws ClassNotFoundException,
SQLException {
        try{

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

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

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

```

```
}
```

FlightManagementSystem.java

```
import java.util.*;
import java.sql.*;
public class FlightManagementSystem{
    public ArrayList<Flight> viewFlightsBySourceDestination(String source, String destination){
        DB db=new DB();
        ArrayList<Flight> list=new ArrayList<Flight>();
        try{
            int f=0;
            Connection con=db.getConnection();
            Statement st=con.createStatement();
            String sql= "select * from Flight where source= '"+source+"' and destination=
"+destination+"'";
            ResultSet rs=st.executeQuery(sql);
            while(rs.next()){
                f=1;
                Flight x=new Flight(rs.getInt(1), rs.getString(2),rs.getString(3), rs.getInt(4),
rs.getDouble(5));
                list.add(x);
            }
            con.close();
            if(f==1)
                return list;
            else
                return null;
        }
        catch(SQLException e){
            System.out.println("SQL Error. Contact Administrator.");
            return null;
        }
        catch(Exception e){
            System.out.println("Exception. Contact Administrator.");
            return null;
        }
    }
}
```

Flight.java

```
public class Flight {

    private int flightId;
    private String source;
```

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

```
        return ("Flight ID : "+getFlightId());
    }
```

```
}
```

### Perform Calculation

```
import java.util.Scanner;

public class Calculator {

    public static void main (String[] args) {

        Scanner sc=new Scanner(System.in);

        int a = sc.nextInt();

        int b= sc.nextInt();

        Calculate Perform_addition = performAddition();

        Calculate Perform_subtraction = performSubtraction();

        Calculate Perform_product = performProduct();

        Calculate Perform_division = performDivision();

        System.out.println("The sum is "+Perform_addition.performCalculation(a,b));

        System.out.println("The difference is
"+Perform_subtraction.performCalculation(a,b));

        System.out.println("The product is "+Perform_product.performCalculation(a,b));

        System.out.println("The division value is
"+Perform_division.performCalculation(a,b));

    }

    public static Calculate performAddition(){

        Calculate Perform_calculation = (int a,int b)->a+b;

        return Perform_calculation;
```

```
}

public static Calculate performSubtraction(){

    Calculate Perform_calculation = (int a,int b)->a-b;

    return Perform_calculation;

}

public static Calculate performProduct(){

    Calculate Perform_calculation = (int a,int b)->a*b;

    return Perform_calculation;

}

public static Calculate performDivision(){

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

        float c = (float)a;

        float d = (float)b;

        return (c/d);

    };

    return Perform_calculation;

}

}

public interface Calculate {

    float performCalculation(int a,int b);

}
```

```
public class InPatient extends Patient {  
    InPatient(String patientId, String patientname, long mobileNumber,  
    String gender) {  
        super(patientId, patientname, mobileNumber, gender);  
    }  
    InPatient()  
    {  
    }  
    private double roomRent;  
    public double getrent()  
    {  
        return roomRent;  
    }  
    public void setrent(double rent)  
    {  
        roomRent=rent;  
    }  
    public double calculateTotalBill(int no,double medi)  
    {  
        return (roomRent*no)+medi;  
    }  
}  
  
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.util.Scanner;  
  
public class Main {  
    public static void main(String [] args) throws IOException {  
        Scanner sc=new Scanner(System.in);  
        BufferedReader br=new BufferedReader(new  
InputStreamReader(System.in));  
        OutPatient o1=new OutPatient();  
        InPatient o2=new InPatient();  
        System.out.println("1.In Patient\n2.Out Patient");  
        System.out.println("Enter the choice");  
        int a=sc.nextInt();  
        //sc.hasNextLine();  
        System.out.println("Enter the details\nPatient Id");  
        String pid=br.readLine();  
        System.out.println("Patient Name");  
        String pname=br.readLine();  
        System.out.println("Phone Number");  
        long mob=sc.nextLong();  
        System.out.println("Gender");  
        String gen=br.readLine();  
        if(a==1)  
        {  
            System.out.println("Room Rent");  
            double rent=sc.nextDouble();  
        }  
    }  
}
```

```
        System.out.println("Medicinal Bill");
        double med=sc.nextDouble();
        System.out.println("Number of Days of Stay");
        int no=sc.nextInt();
        c2.setrent(rent);
        System.out.println("Amount to be paid
"+c2.calculateTotalBill111(no,med));
    }
    else
    {
        System.out.println("Consultancy Fee");
        double con=sc.nextDouble();
        System.out.println("Medicinal Bill");
        double med=sc.nextDouble();
        System.out.println("Scan Pay");
        int scan=sc.nextInt();
        c1.setcon(con);
        System.out.println("Amount to be paid
"+c1.calculateTotalBill111(scan,med));
    }
}
}

public class OutPatient extends Patient {

    /*OutPatient(String patientId, String patientname, long
mobileNumber, String gender) {
    super(patientId, patientname, mobileNumber, gender);
    // TODO Auto-generated constructor stub
}*/
    OutPatient()
    {
        super();
    }
    private double consultingFee;
    public double getcon()
    {
        return consultingFee;
    }
    public void setcon(double con)
    {
        consultingFee=con;
    }
    public double calculateTotalBill111(int scan,double medi)
    {
        return (consultingFee+scan+medi);
    }
}
```

```
public class Patient {  
    private String patientId,patientname,gender;  
    private long mobileNumber;  
    Patient(String patientId, String patientname, long  
    mobileNumber, String gender)  
    {  
        this.patientId=patientId;  
        this.patientname=patientname;  
        this.gender=gender;  
        this.mobileNumber=mobileNumber;  
    }  
    Patient()  
    {  
    }  
    public String getpaid(){  
        return patientId;  
    }  
    public String getpaname(){  
        return patientname;  
    }  
    public String getpagen(){  
        return gender;  
    }  
    public long getpanob(){  
        return mobileNumber;  
    }  
  
    public void setpaid(String id){  
        patientId=id;  
    }  
    public void setpaname(String name){  
        patientname=name;  
    }  
    public void setpagen(String gen){  
        gender=gen;  
    }  
    public void setpanob(long mob){  
        mobileNumber=mob;  
    }  
}
```

## Payment Inheritance

### Bill.java

```
public class Bill {  
  
    public String processPayment(Payment obj) {  
  
        String message = "Payment not done and your due amount is "+obj.getDueAmount();  
    }  
}
```

```

if(obj instanceof Cheque ) {

    Cheque cheque = (Cheque) obj;

    if(cheque.payAmount())

        message = "Payment done successfully via cheque";

}

else if(obj instanceof Cash ) {

    Cash cash = (Cash) obj;

    if(cash.payAmount())

        message = "Payment done successfully via cash";

}

else if(obj instanceof Credit ) {

    Credit card = (Credit) obj;

    if(card.payAmount())

        message = "Payment done successfully via creditcard. Remaining amount in your
"+card.getCardType()+" card is "+card.getCreditCardAmount();

}

return message;
}
}

```

### **Cash.java**

```

public class Cash extends Payment{

    private int cashAmount;

    public int getCashAmount() {

```

```
    return cashAmount;  
}  
  
public void setCashAmount(int cashAmount) {  
    this.cashAmount = cashAmount;  
}  
  
@Override  
  
public boolean payAmount() {  
    return getCashAmount() >= getDueAmount();  
}  
}
```

### Cheque.java

```
import java.text.ParseException;  
  
import java.text.SimpleDateFormat;  
  
import java.util.Calendar;  
  
import java.util.Date;  
  
import java.util.GregorianCalendar;  
  
public class Cheque extends Payment {  
  
    private String chequeNo;  
  
    private int chequeAmount;  
  
    private Date dateOfIssue;  
  
    public String getChequeNo() {  
        return chequeNo;  
    }
```

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

public int getChequeAmount() {
    return chequeAmount;
}

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

public Date getDateOfIssue() {
    return dateOfIssue;
}

public void setDateOfIssue(Date dateOfIssue) {

    this.dateOfIssue = dateOfIssue;
}

@Override

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

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

public void generateDate(String date) {

    try {
        Date issueDate = new SimpleDateFormat("dd-MM-yyyy").parse(date);
        setDateOfIssue(issueDate);
    }
    catch (ParseException e) {
        e.printStackTrace();
    }
}
```

```
    }  
}
```

### Credit.java

```
public class Credit extends Payment {  
  
    private int creditCardNo;  
  
    private String cardType;  
  
    private int creditCardAmount;  
  
    public int getCreditCardNo(){  
  
        return creditCardNo;  
  
    }  
  
    public void setCreditCardNo(int creditCardNo) {  
  
        this.creditCardNo = creditCardNo;  
  
    }  
  
    public String getCardType() {  
  
        return cardType;  
  
    }  
  
    public void setCardType(String cardType) {  
  
        this.cardType = cardType;  
  
    }  
  
    public int getCreditCardAmount() {  
  
        return creditCardAmount;  
  
    }  
  
    public void setCreditCardAmount(int creditCardAmount) {
```

```
this.creditCardAmount = creditCardAmount;

}

@Override

public boolean payAmount() {

int tax = 0;

boolean isDeducted = false;

switch(cardType) {

case "silver":

setCreditCardAmount(10000);

tax = (int) (0.02*getDueAmount())+getDueAmount();

if(tax <= getCreditCardAmount()) {

setCreditCardAmount(getCreditCardAmount()-tax);

isDeducted = true;

}

break;

case "gold":

setCreditCardAmount(50000);

tax = (int) (0.05*getDueAmount())+getDueAmount();

if(tax <= getCreditCardAmount()) {

setCreditCardAmount(getCreditCardAmount()-tax);

isDeducted = true;

}

}
```

```
break;

case "platinum":

setCreditCardAmount(100000);

tax = (int) (0.1*getDueAmount())+getDueAmount();

if(tax <= getCreditCardAmount()) {

setCreditCardAmount(getCreditCardAmount()-tax);

isDeducted = true;

}

break;

}

return isDeducted;

}

}
```

### **Main.java**

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {

        Bill bill = new Bill();

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the due amount:");

        int dueAmount = sc.nextInt();

        System.out.println("Enter the mode of payment(cheque/cash/credit):");
```

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

switch (mode) {

    case "cash":

        System.out.println("Enter the cash amount:");

        int cashAmount = sc.nextInt();

        Cash cash = new Cash();

        cash.setCashAmount(cashAmount);

        cash.setDueAmount(dueAmount);

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

        break;

    case "cheque":

        System.out.println("Enter the cheque number:");

        String number = sc.next();

        System.out.println("Enter the cheque amount:");

        int chequeAmount = sc.nextInt();

        System.out.println("Enter the date of issue:");

        String date = sc.next();

        Cheque cheque = new Cheque();

        cheque.setChequeAmount(chequeAmount);

        cheque.setChequeNo(number);

        cheque.generateDate(date);

        cheque.setDueAmount(dueAmount);
```

```
System.out.println(bill.processPayment(cheque));
break;

case "credit":

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

int creditNumber = sc.nextInt();

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

String cardType = sc.next();

Credit credit = new Credit();

credit.setCardType(cardType);

credit.setCreditCardNo(creditNumber);

credit.setDueAmount(dueAmount);

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

default:

break;

}

sc.close();

}

}
```

### **Payment.java**

```
public class Payment {

    private int dueAmount;

    public int getDueAmount() {

        return dueAmount;
    }
}
```

```

    }

    public void setDueAmount(int dueAmount) {

        this.dueAmount = dueAmount;

    }

    public boolean payAmount() {

        return false;

    }

}

```

## HUNGER EATS

```

package com.utility;
import java.util.*;
import com.bean.FoodProduct;
public class Order{
    private double discountPercentage;
    private List<FoodProduct> foodList=new ArrayList<FoodProduct>();

    public double getDiscountPercentage() {
        return discountPercentage;
    }
    public void setDiscountPercentage(double discountPercentage) {
        this.discountPercentage = discountPercentage;
    }
    public List<FoodProduct> getFoodList() {
        return foodList;
    }
    public void setFoodList(List<FoodProduct> foodList) {
        this.foodList = foodList;
    }

    public void findDiscount(String bankName)
    {
        if(bankName.equals("HDFC")) {
            discountPercentage=15.0;
        }
        else if(bankName.equals("ICICI")) {
    }
}

```

```

        discountPercentage=25.0;
    }
else if(bankName.equals("CUB")) {
    discountPercentage=30.0;
}
else if(bankName.equals("SBI")) {
    discountPercentage=50.0;
}
else if(bankName.equals("OTHERS")) {
    discountPercentage=0.0;
}

}

public void addToCart(FoodProduct foodProductObject)
{
List<FoodProduct> f=getFoodList();
f.add(foodProductObject);
setFoodList(f);

}

public double calculateTotalBill()
{
    double bill = 0;
    List<FoodProduct> f=getFoodList();
    for(int i=0;i<f.size();i++)
    {
//        System.out.println(f.get(i).getCostPerUnit());
//        System.out.println(f.get(i).getQuantity());
        bill+=f.get(i).getQuantity()*f.get(i).getCostPerUnit()*1.0;

    }
//    System.out.println(bill);
//    System.out.println(dis);
    bill=bill-((bill*discountPercentage)/100);
    return bill;
}

}

package com.ui;

import java.util.Scanner;

```

```
import com.utility.Order;
import com.bean.FoodProduct;

public class UserInterface{

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

        System.out.println("Enter the number of items");
        itemno=sc.nextInt();
        System.out.println("Enter the item details");

        Order o=new Order();

        for(int i=0;i<itemno;i++)
        {
            FoodProduct fd=new FoodProduct();
            System.out.println("Enter the item id");
            fd.setFoodId(sc.nextInt());
            System.out.println("Enter the item name");
            fd.setFoodName(sc.next());
            System.out.println("Enter the cost per unit");
            fd.setCostPerUnit(sc.nextDouble());
            System.out.println("Enter the quantity");
            fd.setQuantity(sc.nextInt());
            o.addToCart(fd);

        }

        System.out.println("Enter the bank name to avail offer");
        bank=sc.next();
        o.findDiscount(bank);

        System.out.println("Calculated Bill Amount:"+o.calculateTotalBill());

    }
}
```

```
}

package com.bean;

public class FoodProduct {

    private int foodId;
    private String foodName;
    private double costPerUnit;
    private int quantity;

    public int getFoodId() {
        return foodId;
    }

    public void setFoodId(int foodId) {
        this.foodId = foodId;
    }

    public String getFoodName() {
        return foodName;
    }

    public void setFoodName(String foodName) {
        this.foodName = foodName;
    }

    public double getCostPerUnit() {
        return costPerUnit;
    }

    public void setCostPerUnit(double costPerUnit) {
        this.costPerUnit = costPerUnit;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

}
```

### Singapore

```
import java.util.*;

public class tourism {
    static String name;
    static String place;
    static int days;
    static int tickets;
```

```

static double price = 0.00;
static double total = 0.00;
public static void main(String[] args){
    Scanner in = new Scanner(System.in);
    System.out.println("Enter the passenger name");
    name = in.nextLine();
    System.out.println("Enter the place name");
    place=in.nextLine();
    if(place.equalsIgnoreCase("beach"))

||place.equalsIgnoreCase("pilgrimage")||place.equalsIgnoreCase("heritage")||place.equalsIgnoreCase("Hills")||place.equalsIgnoreCase("palls")||place.equalsIgnoreCase("adventure")){
        System.out.println("Enter the number of days");
        days = in.nextInt();
        if(days>0){
            System.out.println("Enter the number of Tickets");
            tickets = in.nextInt();
            if(tickets>0){
                if(place.equalsIgnoreCase("beach")){
                    price = tickets*270;
                    if(price>1000){
                        total = 85*price/100;
                        System.out.printf("Price:%.2f",total);
                    }
                    else {
                        System.out.printf("Price:%.2f",price);
                    }
                }
                else if(place.equalsIgnoreCase("prilgrimage")){
                    price = tickets*350;
                    if(price>1000){
                        total = 85*price/100;
                        System.out.printf("Price:%.2f",total);
                    }
                    else {
                        System.out.printf("Price:%.2f",price);
                    }
                }
                else if(place.equalsIgnoreCase("heritage")){
                    price = tickets*430;
                    if(price>1000){
                        total = 85*price/100;
                        System.out.printf("Price:%.2f",total);
                    }
                }
            }
        }
    }
}

```

```
else {
    System.out.printf("Price:%.2f",price);
}
}

else if(place.equalsIgnoreCase("hills")){
    price = tickets*780;
    if(price>1000){
        total = 85*price/100;
        System.out.printf("Price:%.2f",total);
    }
    else {
        System.out.printf("Price:%.2f",price);
    }
}

else if(place.equalsIgnoreCase("palls")){
    price = tickets*1200;
    if(price>1000){
        total = 85*price/100;
        System.out.printf("Price:%.2f",total);
    }
    else {
        System.out.printf("Price:%.2f",price);
    }
}

else {
    price = tickets*4500;
    if(price>1000){
        total = 85*price/100;
        System.out.printf("Price:%.2f",total);
    }
    else {
        System.out.printf("Price:%.2f",price);
    }
}

}

else{
    System.out.println(tickets+" is an Invalid no. of tickets");
}
}

else{
    System.out.println(days+" is an Invalid no. of days");
}
}
```

```

        else {
            System.out.println(place+" is an Invalid place");
        }
    }
}

```

### Prime no ending

```

import java.util.*;
public class Main
{
    public static void main (String[] args) {
        int flag=0, k=0, z=0;
        Scanner sc =new Scanner(System.in );
        System.out.println("Enter the first number");
        int f=sc.nextInt();
        System.out.println("Enter the last number");
        int l=sc.nextInt();
        for(int i=f; i<=l; i++)
        {
            for(int j=2; j<i; j++)// this loop increments flag if i is divisible by j
            {
                if(i%j==0)
                {
                    flag++;
                }
            }
            if(i==l && (flag!=0 || i%10!=1))//when last number is not a prime
            {
                while(z==0)
                {
                    for(int a=2; a<i; a++)
                    {
                        if(i%a==0)
                        {
                            flag++;
                        }
                    }
                    if(i%10==1 && flag==0)
                    {
                        System.out.print(", "+i);
                        z++;
                    }
                    flag=0;
                    i++;
                }
            }
        }
    }
}

```

```

        }
    }
    if(i%10==1 && flag==0)//to check for last digit 1 and prime
    {
        if(k==0)
        {
            System.out.print(i);
            k++;
        }
        else
        {
            System.out.print(", "+i);
        }
    }
    flag=0;
}
}

}
}

```

### Query Set

```

public class Query {

    private class DataSet{
        private String theatreId;
        private String theatreName;
        private String location;
        private int noOfScreen;
        private double ticketCost;
        public String getTheatreId() {
            return theatreId;
        }
        public void setTheatreId(String theatreId) {
            this.theatreId = theatreId;
        }
        public String getTheatreName() {
            return theatreName;
        }
        public void setTheatreName(String theatreName) {
            this.theatreName = theatreName;
        }
        public String getLocation() {
            return location;
        }
    }
}

```

```
public void setLocation(String location) {
this.location = location;
}
public int getNoOfScreen() {
return noOfScreen;
}
public void setNoOfScreen(int noOfScreen) {
this.noOfScreen = noOfScreen;
}
public double getTicketCost() {
return ticketCost;
}
public void setTicketCost(double ticketCost) {
this.ticketCost = ticketCost;
}
@Override
public String toString() {
return "Theatre id: " + theatreId + "\nTheatre name: " + theatreName + "\nLocation: " + location
+ "\nNo of Screen: " + noOfScreen + "\nTicket Cost: " + ticketCost+"\n";
}
}

private String queryId;
private String queryCategory;
private DataSet primaryDataset;
private DataSet secondaryDataSet;
public String getQueryId() {
return queryId;
}
public void setQueryId(String queryId) {
this.queryId = queryId;
}
public String getQueryCategory() {
return queryCategory;
}
public void setQueryCategory(String queryCategory) {
this.queryCategory = queryCategory;
}
public DataSet getPrimaryDataset() {
return primaryDataset;
}
public void setPrimaryDataset(DataSet primaryDataset) {
this.primaryDataset = primaryDataset;
}
```

```

public DataSet getSecondaryDataSet() {
    return secondaryDataSet;
}
public void setSecondaryDataSet(DataSet secondaryDataSet) {
    this.secondaryDataSet = secondaryDataSet;
}
@Override
public String toString() {
    return "Primary data set\n" + primaryDataset
        + "Secondary data set\n" + secondaryDataSet + "Query id: " + queryId + "\nQuery category=" +
        queryCategory;
}
}

}

```

```

import java.util.Scanner;
public class TestApplication {
    public static void main(String[] args) {
        Query query = new Query();
        Scanner sc = new Scanner(System.in);
        Query.DataSet primary = query.new DataSet();
        Query.DataSet secondary = query.new DataSet();
        System.out.println("Enter the Details of primary data set");
        System.out.println("Enter the theatre id");
        String theatreid = sc.nextLine();
        primary.setTheatreId(theatreid);
        sc.nextLine();
        System.out.println("Enter the theatre name");
        String theatrename = sc.nextLine();
        primary.setTheatreName(theatrename);
        sc.nextLine();
        System.out.println("Enter the location");
        String location = sc.nextLine();
        primary.setLocation(location);
        sc.nextLine();
        System.out.println("Enter the no of screens");
        int screens = sc.nextInt();
        primary.setNoOfScreen(screens);
        System.out.println("Enter the ticket cost");
        double cost = sc.nextDouble();
        primary.setTicketCost(cost);
    }
}

```

```

System.out.println("ENter the details of secondary data set");
System.out.println("Enter the theatre id");
theatreid = sc.nextInt();
secondary.setTheatreId(theatreid);
sc.nextLine();
System.out.println("Enter the theatre name");
theatrename = sc.next();
secondary.setTheatreName(theatrename);
sc.nextLine();
System.out.println("Enter the location");
location = sc.next();
secondary.setLocation(location);
sc.nextLine();
System.out.println("Enter the no of screens");
screens = sc.nextInt();
secondary.setNoOfScreen(screens);
System.out.println("Enter the ticket cost");
cost = sc.nextDouble();
secondary.setTicketCost(cost);
System.out.println("Enter the query id");
String queryid = sc.next();
query.setQueryId(queryid);
sc.nextLine();
System.out.println("Enter the query category");
String querycategory = sc.next();
query.setQueryCategory(querycategory);
sc.nextLine();
query.setPrimaryDataset(primary);
query.setSecondaryDataSet(secondary);

System.out.println(query);
}
}

```

### Extract book

```

import java.util.Scanner;

class ExtractBook {

    public static int extractDepartmentCode(String input) {
        return Integer.parseInt(input.substring(0, 3));
    }
}

```

```
public static String extractDepartmentName(int code) {  
  
    switch (code) {  
        case 101:  
            return "Accounting";  
        case 102:  
            return "Economics";  
        case 103:  
            return "Engineering";  
    }  
  
    throw new Error(code + " is invalid department code");  
}  
  
public static int extractDate(String input) {  
    String yearStr = input.substring(3, 7);  
    try {  
        int year = Integer.parseInt(yearStr);  
        if (year > 2020 || year < 1900) {  
            throw new NumberFormatException();  
        }  
        return year;  
    } catch (NumberFormatException e) {  
        throw new Error(yearStr + " is invalid year");  
    }  
}  
  
public static int extractNumberOfPages(String input) {  
    String pagesStr = input.substring(7, 12);  
    try {  
        int pages = Integer.parseInt(pagesStr);  
        if (pages < 10) {  
            throw new NumberFormatException();  
        }  
        return pages;  
    } catch (NumberFormatException e) {  
        throw new Error(pagesStr + " are invalid pages");  
    }  
}  
  
public static String extractBookId(String input) {  
    String id = input.substring(12, 18);  
    if (!Character.isAlphabetic(id.charAt(0)))
```

```

        throw new NumberFormatException();
    try {
        Integer.parseInt(id.substring(1));
    } catch (NumberFormatException e) {
        throw new Error(id + " is invalid book id");
    }
    return id;
}

public static void parseAndPrint(String str) {
    if (str.length() != 18) {
        System.out.println(str + " is an invalid input");
        return;
    }

    try {
        int dCode = extractDepartmentCode(str);
        String dString = extractDepartmentName(dCode);
        int year = extractDate(str);
        int pages = extractNumberOfPages(str);
        String bookId = extractBookId(str);

        System.out.println("Department Code: " + dCode);
        System.out.println("Department Name: " + dString);
        System.out.println("Year of Publication: " + year);
        System.out.println("Number of Pages: " + pages);
        System.out.println("Book Id: " + bookId);

    } catch (Error e) {
        System.out.println(e.getMessage());
    }
}

public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    String input = sc.nextLine();
    parseAndPrint(input);
    sc.close();
}
}

```

```
import java.util.*;
class FDScheme {

    private int schemeNo;
    private double depositAmt;
    private int period;
    private float rate;
    public FDScheme(int schemeNo, double depositAmt, int period) {
        super();
        this.schemeNo = schemeNo;
        this.depositAmt = depositAmt;
        this.period = period;
        calculateInterestRate();
    }
    public int getSchemeNo() {
        return schemeNo;
    }
    public void setSchemeNo(int schemeNo) {
        this.schemeNo = schemeNo;
    }
    public double getDepositAmt() {
        return depositAmt;
    }
    public void setDepositAmt(double depositAmt) {
        this.depositAmt = depositAmt;
    }
    public int getPeriod() {
        return period;
    }
    public void setPeriod(int period) {
        this.period = period;
    }
    public float getRate() {
        return rate;
    }
    public void setRate(float rate) {
        this.rate = rate;
    }

    public void calculateInterestRate()
    {
        if(period>=1 && period<=90)
        {
            this.rate=(float) 5.5;
```

```

        }
        else if(period>=91 && period<=180)
        {
            this.rate=(float) 6.25;
        }
        else if(period>=181 && period<=365)
        {
            this.rate=(float) 7.5;
        }
        System.out.println("Interest rate for "+period+" days is "+this.rate);
    }
}
public class Main{

    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter Scheme no");
        int no=sc.nextInt();
        sc.nextLine();
        System.out.println("Enter Deposit amount");
        double amt=sc.nextDouble();
        System.out.println("enter period of deposit");
        int prd=sc.nextInt();
        FDScheme obj=new FDScheme(no,amt,prd);
    }
}

```

### Annual Salary

```

import java.io.*;
public class Main
{
    public static void main(String[] args)throws IOException
    {
        // Scanner sc=new Scanner(System.in);
        //Fill the code
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter the Employee Name");
        String name=br.readLine();
        System.out.println("Enter percentage of salary");
        double percent=Double.parseDouble(br.readLine());
        if(percent>0&&percent<20)
        {

```

```

System.out.println("Enter the Year of Experience");
int time=Integer.parseInt(br.readLine());

if(time>0&&time<15)
{
    double permonth=12000+(2000*(time));
    double dayshift=permonth*6;
    double nightshift=(((permonth*percent)/100)+permonth)*6;
    double annualIncome=dayshift+nightshift;

    String str="The annual salary of "+name+" is";
    System.out.println(str+" "+annualIncome);

}
else{
    System.out.println((int)time+" is an invalid year of experience");}
}

else
    System.out.println((int)percent+" is an invalid percentage");
}
}

```

### Amity Passenger

```

import java.util.*;
public class PassengerAmenity {

public static void main(String[] args) {

Scanner sc=new Scanner(System.in);
System.out.println("Enter the number of passengers");
int no=sc.nextInt();
sc.nextLine();
int count=0;

if(no>0)
{
String name[]=new String[no];
String seat[]=new String[no];
String arr[]=new String[no];

for(int i=0;i<no;i++)

```

```

{
System.out.println("Enter the name of the passenger "+(i+1));
String str=sc.nextLine();

name[i]=str.toUpperCase();

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

if(seat[i].charAt(0)>='A' && seat[i].charAt(0)<='S')
{

int r=Integer.parseInt(seat[i].substring(1,seat[i].length()));

if(r>=10 && r<=99)
{
count++;
}

else
{
System.out.println(r+" is invalid seat number");
break;
}
}

else
{
System.out.println(seat[i].charAt(0)+" is invalid coach");
break;
}

arr[i]=name[i]+" "+seat[i];
}

if(count==seat.length)
{

Arrays.sort(seat);

for(int i=seat.length-1;i>=0;i--)
{
for(int j=0;j<arr.length;j++)
{
}
}
}

```

```

        if(arr[j].contains(seat[i]))
        {
            System.out.println(arr[j]);
        }
    }

}
}

else
{
    System.out.println(no+" is invalid input");
}
}

}

```

### Change the Case

```

import java.util.*;

public class ChangeTheCase {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String a = sc.next();
        if(a.length() < 3) {
            System.out.println("String length of " + a + " is too short");
            return;
        }
        else if(a.length() > 10) {
            System.out.println("String length of " + a + " is too long");
            return;
        }

        char[] arr = a.toCharArray();
        char[] arr1 = new char[arr.length];
        int j = 0;
        for(int i = 0; i < a.length(); i++) {
            if((arr[i]<65 || ((arr[i]>90) && (arr[i]<97)) || arr[i]>122)) {
                arr1[j++] = arr[i];
            }
        }
    }
}
```

```

}
if(j!=0) {
    System.out.print("String should not contain ");
    for(int i = 0; i<=j; i++) {
        System.out.print(arr1[i]);
    }
    return;
}
char b = sc.next().charAt(0);
int present = 0;
for(int i = 0; i<a.length(); i++) {
    if(arr[i] == Character.toUpperCase(b)) {
        arr[i] = Character.toLowerCase(b);
        present = 1;
    }
    else if(arr[i] == Character.toLowerCase(b)) {
        arr[i] = Character.toUpperCase(b);
        present = 1;
    }
}
if(present == 0) {
    System.out.println("Character " + b + " is not found");
}
else {
    for(int i = 0; i <a.length(); i++) {
        System.out.print(arr[i]);
    }
}
}
}

}

```

### **Club Member**

```

import java.util.Scanner;

public class ClubMember {
    private int memberId;
    private String memberName;
    private String memberType;
    private double membershipFees;

    public ClubMember(int memberId, String memberName, String memberType) {

```

```
super();
this.memberId = memberId;
this.memberName = memberName;
this.memberType = memberType;
calculateMembershipFees();
}
public int getMemberId() {
    return memberId;
}
public void setMemberId(int memberId) {
    this.memberId = memberId;
}
public String getMemberName() {
    return memberName;
}
public void setMemberName(String memberName) {
    this.memberName = memberName;
}
public String getMemberType() {
    return memberType;
}
public void setMemberType(String memberType) {
    this.memberType = memberType;
}
public double getMembershipFees() {
    return membershipFees;
}
public void setMembershipFees(double membershipFees) {
    this.membershipFees = membershipFees;
}

public void calculateMembershipFees() {
    if(!(memberType == "Gold"))
    {
        this.membershipFees=(double) 50000.0;
    }
    else if(!(memberType=="Premium"))
    {
        this.membershipFees=(double) 75000.0;
    }
    System.out.println("Member Id is "+this.memberId);
    System.out.println("Member Name is "+this.memberName);
    System.out.println("Member Type is "+this.memberType);
```

```
System.out.println("Membership Fees is "+this.membershipFees);

}

}

public class Main {
public static void main(String[] args) {
Scanner sc=new Scanner(System.in);
System.out.println("Enter Member Id");
int id=sc.nextInt();
sc.nextLine();
System.out.println("Enter Name");
String name=sc.next();
System.out.println("Enter Member Type");
String type=sc.next();
ClubMember club=new ClubMember(id, name, type);
//club.calculateMembershipFees();
}
}
```

## 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)**

<b>Component Name</b>	<b>Type(Class)</b>	<b>Attributes</b>	<b>Methods</b>	<b>Responsibilities</b>
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)**

<b>Component Name</b>	<b>Type(Class)</b>	<b>Attributes</b>	<b>Methods</b>	<b>Responsibilities</b>
	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 <> 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(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(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    }
37 }
```

```

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     else
54         return false;
55     }
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
11                "+obj.getDueAmount();
12        }
13        else if(obj instanceof Cash)
14        {
15            if(obj.payAmount())
16                res = "Payment done successfully via cash";
17            else
18                res = "Payment not done and your due amount is
19                "+obj.getDueAmount();
20        }
21        else if(obj instanceof Credit)
22        {
23            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.,  $3(3)$   $9(3^3)$   $27(3^3 \cdot 3)$   $81(3^3 \cdot 3^3)$   $243(3^3 \cdot 3^3 \cdot 3)$

**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
<b>Account</b>	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**

<b>Component Name</b>	<b>Type(Class)</b>	<b>Methods</b>	<b>Responsibilities</b>
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**

<b>Component Name</b>	<b>Type(Class)</b>	<b>Methods</b>	<b>Responsibilities</b>
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:**

slehsaesSea

**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@g.s.

#### 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)

<b>Component Name</b>	<b>Type(Class)</b>	<b>Attributes</b>	<b>Methods</b>	<b>Responsibilities</b>
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)

<b>Component Name</b>	<b>Type(Class)</b>	<b>Attributes</b>	<b>Methods</b>	<b>Responsibilities</b>
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

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        }
13    }
14
15}
16

```

```

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 emailld, String movieName){
4         super(ticketId, customerName, mobileNumber, emailld, 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     }  

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 mnmm=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,mnmm);
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,mnmm);
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,mnmm);
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)	Attribute	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	FlightManagement System		public ArrayList<Flight> viewFlightBySourceDestination(String source, String destination)	This method should accept a Source and a destination as parameter and retrieve all the flights with the given source and destination from the database. Return these details as ArrayList<Flight>.

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

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

Column Name	Datatype
flightId	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 nofseats=rst.getInt(4);
23         double flightfare=rst.getDouble(5);
24
25         flightList.add(new Flight(flightId, src, dst, nofseats, 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.nextLine();
9         System.out.println("Enter the destination");
10        String destination=sc.nextLine();
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                                     // load the Driver Class
22                                     Class.forName(props.getProperty("DB_DRIVER_CLASS"));
23
24                                     // create the connection now
25
26     con =
DriverManager.getConnection(props.getProperty("DB_URL"),props.getProperty("DB_USERNAME"),props.getProperty("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 <code>DisplayText</code> 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
<b>PasswordInfo</b>	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. Watican 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:

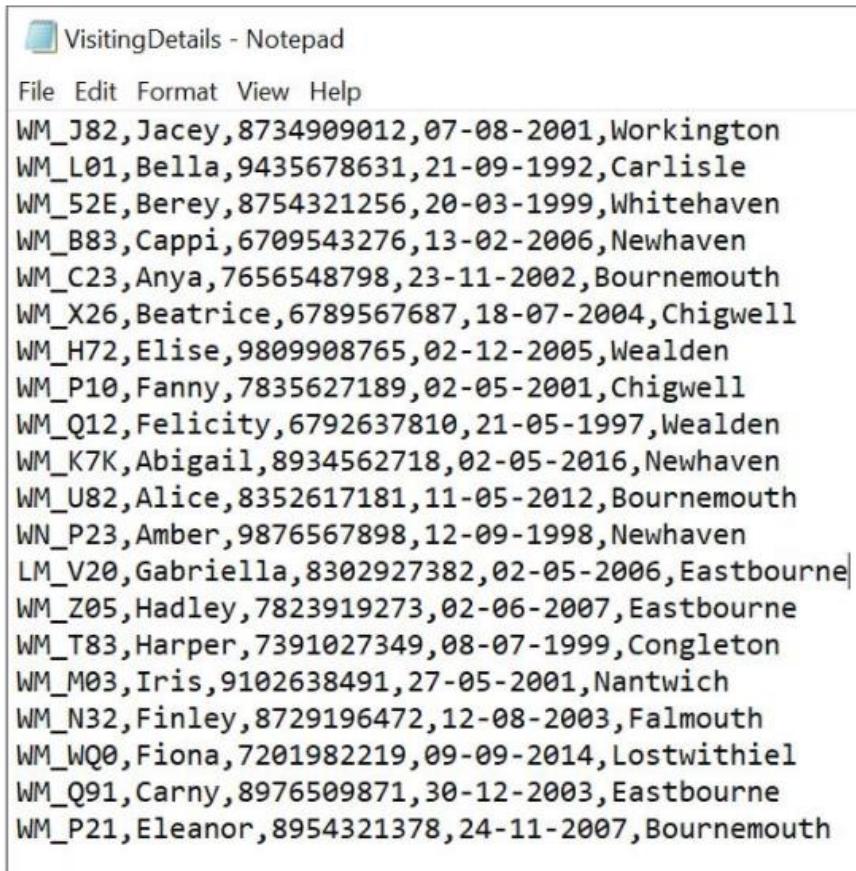
- Visitor class which includes the attributes visitorId, visitorName, mobileNumber, dateOfVisiting and address with all the getters and setters.
- VisitorUtility class which includes the following method declarations.
  - public List<Visitor> generateVisitor(String filePath)
  - public boolean isValidVisitorId(String 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



The screenshot shows a Notepad window titled "VisitingDetails - Notepad". The menu bar includes File, Edit, Format, View, and Help. The main content area contains a list of 25 visitor details, each on a new line. The data is separated by commas and includes fields such as visitor ID, name, date of birth, and place of residence.

```

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
WN_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	public List < Visitor> generateVisitor(String filePath)	<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><b>Note:</b></p> <p>Before converting the separated string into a visitor object, the identified visitorId should be validated using the</p>

		isValidVisitorId method.
VisitorUtility	<pre>public boolean isValidVisitorId (String visitorId)</pre>	<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 “&lt;visitorId&gt; is Invalid Visitor Id”.</p> <p><b>Validation Rules:</b></p> <ul style="list-style-type: none"> <li>Length of the visitorId should be exactly 6.</li> <li>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).</li> </ul> <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	<pre>public List&lt;Visitor&gt; viewVisitorDetailsByDateOfVisiting(Stream&lt;Visitor&gt; visitorStream, String fromDate, String toDate)</pre>	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 <b>particular mentioned</b> 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	<pre>public List &lt;Patient&gt; fetchPatient(String filePath)</pre>	<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 <code>isValidPatientId ()</code>.</p> <p><b>Note:</b> Make sure that the user-defined exception is handled in this method itself.</p>
PatientUtility	<pre>public boolean isValidPatientId (String patientId)</pre>	<p><b>Validation Guidelines for Valid Patient ID:</b></p> <ul style="list-style-type: none"> <li>• The length of the Patient Id should be exactly 6.</li> <li>• 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.</li> </ul> <p>Check whether the patient Id is valid or not. If invalid, this method should handle an <code>InvalidPatientIdException</code> with a message “&lt;patientid&gt; is an Invalid Patient Id”.</p>
PatientUtility	<pre>public List&lt;Patient&gt; retrievePatientRecords_ByDateOfVisit(Stream&lt;Patient&gt; patientStream, String fromDate, String toDate)</pre>	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.

PatientUtility	public Stream<Patient> retrievePatientRecords_ByAddress(Stream<Patient> patientStream, String address)	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	public InvalidPatientIdException(String message)	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,Anya,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         //FILL THE CODE HERE
19         List <Patient> patients =new ArrayList<>();
20         try{
21             File register =new File(filePath);
22             Scanner reader=new Scanner(register);
23             while(reader.hasNextLine()){
24                 Patient p = new Patient();
25                 String[] infos=reader.nextLine().split(",");
26                 try{
27                     if(isValidPatientId(infos[0])){
28                         p.setPatientId(infos[0]);
29                         p.setPatientName(infos[1]);
30                         p.setContactNumber(infos[2]);
31                         p.setDateOfVisit(infos[3]);
32                         p.setPatientAddress(infos[4]);
33                         patients.add(p);
34                     }
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
50     public boolean isValidPatientId (String patientId)throws InvalidPatientIdException
51     {
52
53         //FILL THE CODE HERE
54         Pattern p =Pattern.compile("WM_[A-Z][0-9]{2}$");
55         Matcher m=p.matcher(patientId);
56         boolean ne =m.matches();
57         if(!ne){
58             throw new InvalidPatientIdException(patientId+"is an Invalid Patient Id.");
59         }
60         //return inValid;
61         return ne;
62     }
63
64
65
66     public List<Patient> retrievePatientRecords_ByDateOfVisit(Stream<Patient> patientStream, String
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             "fetchPatient:java.util.List,isValidPatientId:boolean,retrievePatientRecords_ByDateOfVisit:java.util.List",
23             "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.
44                         + "Class name is same as
45 Please manually verify that the "
46                         + "skeleton before uploading");
47             }
48         }
49     }
50     protected final void validateMethodSignature(String methodWithExcptn, String className) {
51         Class cls = null;
52         try {
53
54             String[] actualmethods = methodWithExcptn.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
73 (!findMethod.getReturnType().getName().equals(returnType))) {
74
75                         } else {
76                             LOG.info("Method signature of "
77
+ 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                         + "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/EventManagement.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}
```

```

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
48     else if(p.getEventName().equalsIgnoreCase("quiz")){
49         p.setRegistrationFee(300);
50     }
51     else if(p.getEventName().equalsIgnoreCase("games")){
52         p.setRegistrationFee(100);
53     }
54     else{
55         throw new InvalidEventException("Event Name is Invalid");
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        }
49        else{
50            System.out.println("Number of participants for"+srch+"event is "+count);
51        }
52    }
53}
54
55
56
57
```

### *TechnologyFest/src/Participant.java*

```
1 public class Participant {
```



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
12 * smooth auto evaluation
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
skelton before uploading");
96         }
97         return iscorrect;
98
99     }
100
101    protected final void validateMethodSignature(String methodWithExcptn, String className) {
102        Class cls = null;
103        try {
104
105            String[] actualmethods = methodWithExcptn.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
123                    if
124 (!findMethod.getReturnType().getName().contains(returnType))) {
125                         errorFlag = true;
126                         LOG.log(Level.SEVERE, " You
have changed the " + "return type in " + methodName

```

```

125
126     method. Please stick to the " + "skeleton provided");
127
128 } else {
129     LOG.info("Method signature of "
130     + methodName + " is valid");
131 }
132 }
133 if (!foundMethod) {
134     errorFlag = true;
135     LOG.log(Level.SEVERE, " Unable to find the given
public method " + methodName
136                                         + ". Do not change the " + "given
public method name. " + "Verify it with the skeleton");
137 }
138 }
139 if (!errorFlag) {
140     LOG.info("Method signature is valid");
141 }
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 "
147                                         + "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 Employeeld, String EmployeeName, int yearsOfExperience,  
    String gender, double salary, int supplementaryTrainingHours, int scorePoints)  
    {  
        super(Employeeld, EmployeeName, yearsOfExperience, gender, salary);  
    }
```

```
this.supplementaryTrainingHours=supplementaryTrainingHours;  
this.scorePoints=scorePoints;  
}  
  
public double calculateIncrementedSalary(int incrementPercentage){  
double total=(supplementaryTrainingHours*500)+(scorePoints*50)+this.salary;  
double incsalary=total+(total*incrementPercentage/100);  
return incsalary;  
}  
  
}
```

#### **User Interface:**

```
import java.util.Scanner;  
public class UserInterface {  
  
public static void main(String[] args){  
  
Scanner sc=new Scanner(System.in);  
System.out.println("Enter Employee Id");  
String EmployeeId = sc.next();  
System.out.println("Enter Employee name");  
String EmployeeName = sc.next();  
System.out.println("Enter Experience in years");  
int yearsOfExperience = sc.nextInt();  
System.out.println("Enter Gender");  
String gender = sc.next();  
System.out.println("Enter Salary");  
double salary=sc.nextDouble();
```

```
double incSalary=0;

if(yearsOfExperience>=1 && yearsOfExperience <= 5)

{

System.out.println("Enter Supplementary Training Hours");

int supplementaryTrainingHours = sc.nextInt();

System.out.println("Enter Score Points");

int scorePoints = sc.nextInt();

TraineeEmployees te=new TraineeEmployees(EmployeeId, EmployeeName, yearsOfExperience, gender, salary, supplementaryTrainingHours, scorePoints);

incSalary=te.calculateIncrementedSalary(5);

System.out.println("Incremented Salary is "+incSalary);

}

else if(yearsOfExperience>=6 && yearsOfExperience <=10)

{

System.out.println("Enter Supplementary Hours");

int supplementaryHours = sc.nextInt();

System.out.println("Enter Food Allowance");

double foodAllowance = sc.nextDouble();

CasualEmployee ce=new CasualEmployee(EmployeeId, EmployeeName, yearsOfExperience, gender, salary, supplementaryHours, foodAllowance);

incSalary = ce.calculateIncrementedSalary(12);

System.out.println("Incremented Salary is "+incSalary);

}

else if(yearsOfExperience>=10 && yearsOfExperience <=25)

{

System.out.println("Enter Medical Allowance");

double medicalAllowance = sc.nextDouble();

System.out.println("Enter Vehicle Allowance");

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

```

PermanentEmployee pe = new PermanentEmployee(EmployeeId, EmployeeName,
yearsOfExperience, gender, salary, medicalAllowance, vehicleAllowance);

incSalary=pe.calculateIncrementedSalary(12);

System.out.println("Incremented Salary is "+incSalary);

}

else

System.out.println("Provide valid Years of Experience");

}

}

}

```

## 2.Dominion Cinemas

### Book Movie Ticket:

```

public class BookAMovieTicket {

protected String ticketId;

protected String customerName;

protected long mobileNumber;

protected String emailId;

protected String movieName;

public void setticketId( String ticketId){

this.ticketId=ticketId;

}

public void setcustomerName( String customerName){

this.customerName=customerName;

}

public void setmobileNumber( long mobileNumber){

this.mobileNumber=mobileNumber;

}

public void setemailId( String emailId){

}

```

```
this.emaiId=emaiId;
}

public void setmovieName( String movieName){
this.movieName=movieName;
}

public String getticketId(){
return ticketId;
}

public String getcustomerName(){
return customerName;
}

public String getemaiId(){
return emaiId;
}

public String getmovieName(){
return movieName;
}

public long getmobileNumber(){
return mobileNumber;
}

public BookAMovieTicket(String ticketId,String customerName,long mobileNumber,String emaiId,String movieName)
{
this.ticketId=ticketId;
this.customerName=customerName;
this.mobileNumber=mobileNumber;
this.emaiId=emaiId;
this.movieName=movieName;
}
```

```
}
```

### **Gold Ticket:**

```
public class GoldTicket extends BookAMovieTicket {  
    public GoldTicket(String ticketId, String customerName, long mobileNumber,  
        String emailId, String movieName) {  
        super(ticketId, customerName, mobileNumber, emailId, movieName);  
    }  
    public boolean validateTicketId(){  
        int count=0;  
        if(ticketId.contains("GOLD")){  
            count++;  
            char[] cha=ticketId.toCharArray();  
            for(int i=4;i<7;i++){  
                if(cha[i]>='1'&& cha[i]<='9')  
                    count++;  
            }  
            if(count==4)  
                return true;  
            else  
                return false;  
        }  
        public double calculateTicketCost(int numberOfTickets,String ACFacility){  
            double amount;  
            if(ACFacility.equals("yes")){  
                amount=500*numberOfTickets;  
            }  
            else{  
                amount=350*numberOfTickets;  
            }  
        }  
    }  
}
```

```
}

return amount;

}

}
```

### **Platinum Ticket:**

```
public class PlatinumTicket extends BookAMovieTicket

{

public PlatinumTicket(String ticketId, String customerName, long mobileNumber, String
emailId, String movieName)

{

super(ticketId, customerName, mobileNumber, emailId, movieName);

}

public boolean validateTicketId(){

int count=0;

if(ticketId.contains("PLATINUM")){

count++;

char[] cha=ticketId.toCharArray();

for(int i=8;i<11;i++){

if(cha[i]>='1'&& cha[i]<='9')

count++;

}

if(count==4)

return true;

else

return false;

}

public double calculateTicketCost(int numberOfTickets, String ACFacility){

double amount;

if(ACFacility.equals("yes")){

```

```
amount=750*numberOfTickets;  
}  
  
else{  
amount=600*numberOfTickets;  
}  
  
return amount;  
}  
  
}
```

### **Silver Ticket:**

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

```
public double calculateTicketCost(int numberOfTickets, String ACFacility){  
    double amount;  
    if(ACFacility.equals("yes")){  
        amount=250*numberOfTickets;  
    }  
    else{  
        amount=100*numberOfTickets;  
    }  
    return amount;  
}
```

### **User Interface:**

```
import java.util.*;  
  
public class UserInterface {  
    public static void main(String[] args) {  
        Scanner sc=new Scanner(System.in);  
        System.out.println("Enter Ticket Id");  
        String tid=sc.next();  
        System.out.println("Enter Customer Name");  
        String cnm=sc.next();  
        System.out.println("Enter Mobile Number");  
        long mno=sc.nextLong();  
        System.out.println("Enter Email id");  
        String email=sc.next();  
        System.out.println("Enter Movie Name");  
        String mnm=sc.next();  
        System.out.println("Enter number of tickets");  
        int tno=sc.nextInt();
```

```
System.out.println("Do you want AC or not");

String choice =sc.next();

if(tid.contains("PLATINUM")){
    PlatinumTicket PT=new PlatinumTicket(tid,cnm,mno,email,mnm);
    boolean b1=PT.validateTicketId();
    if(b1==true){
        double cost =PT.calculateTicketCost(tno, choice);
        System.out.println("Ticket cost is "+ cost);
    }
    else if(b1==false){
        System.out.println("Provide valid Ticket Id");
        System.exit(0);
    }
}

else if(tid.contains("GOLD")){
    GoldTicket GT=new GoldTicket(tid,cnm,mno,email,mnm);
    boolean b2=GT.validateTicketId();
    if(b2==true){
        double cost=GT.calculateTicketCost(tno, choice);
        System.out.println("Ticket cost is "+cost);
    }
    else if (b2==false){
        System.out.println("Provide valid Ticket Id");
        System.exit(0);
    }
}

else if(tid.contains("SILVER")){
    SilverTicket ST=new SilverTicket(tid,cnm,mno,email,mnm);
    boolean b3=ST.validateTicketId();
```

```

if(b3==true){

double cost=ST.calculateTicketCost(tno, choice);

System.out.println("Ticket cost is "+cost);

}

else if(b3==false){

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

System.exit(0);

}

}

}

}

}

```

### 3.Little Innovators

**Main:**

```

import java.util.*;

public class Main {

    public static void main(String args[])

    {

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

        Scanner sc=new Scanner(System.in);

        String st=sc.nextLine();

        String op=st;

        st=st.replaceAll(" ","");

        boolean d=st.matches("[a-zA-Z]+");

        if(!d)

        {

            System.out.println("Invalid Slogan");

        }

    }

}

```

```

        else
        {
            char a[]=st.toCharArray();
            char b[]=new char[100];
            b[0]='0';
            int same=0,i=a.length,j=0,l=0;
            while(j<i)

            {
                int count=0;
                for(int k=0;k<i;k++)
                {
                    if(a[j]==a[k])
                    {
                        count++;
                    }
                }
                if(count==1)
                {
                    l++;
                    b[l]=a[j];
                    j++;
                }
            }
            else
            {
                j++;
            }
        }

        if(l==(i-l))
            System.out.println("All the guidelines
are satisfied for "+op);
    }
}

```

```

        else
            System.out.println(op+" does not satisfy
the guideline");
    }
}
}

```

#### 4.Kidsor Home appliances

##### Air Conditioner:

```

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

```

```
this.capacity = capacity;  
}  
  
public double calculateProductPrice(){  
    double price = 0;  
  
    if(airConditionerType.equalsIgnoreCase("Residential")){  
        if (capacity == 2.5){  
            price = 32000;  
        }  
        else if(capacity == 4){  
            price = 40000;  
        }  
        else if(capacity == 5.5){  
            price = 47000;  
        }  
    }  
  
    else if(airConditionerType.equalsIgnoreCase("Commercial")){  
        if (capacity == 2.5){  
            price = 40000;  
        }  
        else if(capacity == 4){  
            price = 55000;  
        }  
        else if(capacity == 5.5){  
            price = 67000;  
        }  
    }  
  
    else if(airConditionerType.equalsIgnoreCase("Industrial")){  
        if (capacity == 2.5){  
            price = 47000;
```

```
}

else if(capacity == 4){

price = 60000;

}

else if(capacity == 5.5){

price = 70000;

}

}

return price;

}

}
```

### **Electronic Products:**

```
public class ElectronicProducts {

protected String productId;

protected String productName;

protected String batchId;

protected String dispatchDate;

protected int warrantyYears;

public ElectronicProducts(String productId, String productName, String batchId,
String dispatchDate, int warrantyYears) {

this.productId = productId;

this.productName = productName;

this.batchId = batchId;

this.dispatchDate = dispatchDate;

this.warrantyYears = warrantyYears;

}

public String getProductId() {

return productId;
```

```
}

public void setProductId(String productId) {
    this.productId = productId;
}

public String getProductName() {
    return productName;
}

public void setProductName(String productName) {
    this.productName = productName;
}

public String getBatchId() {
    return batchId;
}

public void setBatchId(String batchId) {
    this.batchId = batchId;
}

public String getDispatchDate() {
    return dispatchDate;
}

public void setDispatchDate(String dispatchDate) {
    this.dispatchDate = dispatchDate;
}

public int getWarrantyYears() {
    return warrantyYears;
}

public void setWarrantyYears(int warrantyYears) {
    this.warrantyYears = warrantyYears;
}
```

**LED TV:**

```
public class LEDTV extends ElectronicProducts {  
    private int size;  
    private String quality;  
    public LEDTV(String productId, String productName, String batchId, String  
dispatchDate, int warrantyYears, int size, String quality) {  
        super(productId, productName, batchId, dispatchDate, warrantyYears);  
        this.size = size;  
        this.quality = quality;  
    }  
    public int getSize() {  
        return size;  
    }  
    public void setSize(int size) {  
        this.size = size;  
    }  
    public String getQuality() {  
        return quality;  
    }  
    public void setQuality(String quality) {  
        this.quality = quality;  
    }  
    public double calculateProductPrice(){  
        double price = 0;  
        if(quality.equalsIgnoreCase("Low")){  
            price = size * 850;  
        }  
        else if(quality.equalsIgnoreCase("Medium")){  
            price = size * 1250;  
        }  
    }  
}
```

```
}

else if(quality.equalsIgnoreCase("High")){
    price = size * 1550;
}

return price;
}

}
```

### **Microwave Oven:**

```
public class MicrowaveOven extends ElectronicProducts{

    private int quantity;

    private String quality;

    public MicrowaveOven(String productId, String productName, String batchId, String
dispatchDate, int warrantyYears, int quantity, String quality) {
        super(productId, productName, batchId, dispatchDate, warrantyYears);
        this.quantity = quantity;
        this.quality = quality;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

    public String getQuality() {
        return quality;
    }

    public void setQuality(String quality) {
        this.quality = quality;
    }
}
```

```
}

public double calculateProductPrice(){

double price = 0;

if(quality.equalsIgnoreCase("Low")){
price = quantity * 1250;

}

else if(quality.equalsIgnoreCase("Medium")){
price = quantity * 1750;

}

else if(quality.equalsIgnoreCase("High")){
price = quantity * 2000;

}

return price;
}

}
```

### **User Interface:**

```
import java.util.Scanner;

public class UserInterface {

public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
System.out.println("Enter Product Id");
String productId = sc.next();
System.out.println("Enter Product Name");
String productName = sc.next();
System.out.println("Enter Batch Id");
String batchId = sc.next();
System.out.println("Enter Dispatch Date");
```

```
String dispatchDate = sc.next();
System.out.println("Enter Warranty Years");
int warrantyYears = sc.nextInt();
double price;
String quality;
switch(productName){
case "AirConditioner":
System.out.println("Enter type of Air Conditioner");
String type = sc.next();
System.out.println("Enter quantity");
double capacity = sc.nextDouble();
AirConditioner ac = new AirConditioner(productId, productName, batchId,
dispatchDate, warrantyYears, type, capacity);
price = ac.calculateProductPrice();
System.out.printf("Price of the product is %.2f", price);
break;
case "LEDTV":
System.out.println("Enter size in inches");
int size = sc.nextInt();
System.out.println("Enter quality");
quality = sc.next();
LEDTV l = new LEDTV(productId, productName, batchId, dispatchDate,
warrantyYears, size, quality);
price = l.calculateProductPrice();
System.out.printf("Price of the product is %.2f", price);
break;
case "MicrowaveOven":
System.out.println("Enter quantity");
int quantity = sc.nextInt();
```

```

System.out.println("Enter quality");
quality = sc.next();

MicrowaveOven m = new MicrowaveOven(productId, productName, batchId,
dispatchDate, warrantyYears, quantity, quality);

price = m.calculateProductPrice();

System.out.printf("Price of the product is %.2f", price);

break;

default:

System.out.println("Provide a valid Product name");

System.exit(0);

}

}

}

```

## 5. Reverse a word

```

import java.util.*;

class HelloWorld {

    public static void main(String[] args) {

        String[] words ;
        Scanner myObj = new Scanner(System.in);

        String sentence= myObj.nextLine();
        words=sentence.split(" ");
        if(words.length<3)
            System.out.println("Invalid Sentence");
        else{

```

```
String a=words[0].substring(0,1);
String b=words[1].substring(0,1);
String c=words[2].substring(0,1);
if(a.equalsIgnoreCase(b)&&b.equalsIgnoreCase(c))
{
    StringBuilder input1 = new StringBuilder();

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

    input1=input1.reverse();
    input1.append(words[0]);

    System.out.println(input1);
}
else {

    StringBuilder input1 = new StringBuilder();

    input1.append(words[0]);
    // reverse StringBuilder input1

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

    System.out.println(input1);
}

}
```

}

}

```
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));
    }
}
```



Edit with WPS Office

```
        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));
    }
}
```



Edit with WPS Office

```
import java.util.*;
public class tourism {
    static String name;
    static String place;
    static int days;
    static int tickets;
    static double price = 0.00;
    static double total = 0.00;
    public static void main(String[] args){
        Scanner in = new Scanner(System.in);
        System.out.println("Enter the passenger name");
        name = in.nextLine();
        System.out.println("Enter the place name");
        place=in.nextLine();
        if(place.equalsIgnoreCase("beach")
            ||place.equalsIgnoreCase("pilgrimage")||
place.equalsIgnoreCase("heritage")||place.equalsIgnoreCase("Hills")||
place.equalsIgnoreCase("palls")||place.equalsIgnoreCase("adventure")){
            System.out.println("Enter the number of days");
            days = in.nextInt();
            if(days>0){
                System.out.println("Enter the number of Tickets");
                tickets = in.nextInt();
                if(tickets>0){
                    if(place.equalsIgnoreCase("beach")){
                        price = tickets*270;
                        if(price>1000){
                            total = 85*price/100;
                            System.out.printf("Price:%.2f",total);
                        }
                        else {
                            System.out.printf("Price:%.2f",price);
                        }
                    }
                    else if(place.equalsIgnoreCase("prilgrimage")){
                        price = tickets*350;
                        if(price>1000){
                            total = 85*price/100;
                            System.out.printf("Price:%.2f",total);
                        }
                        else {
                            System.out.printf("Price:%.2f",price);
                        }
                    }
                    else if(place.equalsIgnoreCase("heritage")){
                        price = tickets*430;
                        if(price>1000){
                            total = 85*price/100;
                            System.out.printf("Price:%.2f",total);
                        }
                        else {
                            System.out.printf("Price:%.2f",price);
                        }
                    }
                    else if(place.equalsIgnoreCase("hills")){
                        price = tickets*780;
                        if(price>1000){
                            total = 85*price/100;
                            System.out.printf("Price:%.2f",total);
                        }
                        else {
                            System.out.printf("Price:%.2f",price);
                        }
                    }
                }
            }
        }
    }
}
```

```
else if(place.equalsIgnoreCase("palls")){
    price = tickets*1200;
    if(price>1000){
        total = 85*price/100;
        System.out.printf("Price: %.2f", total);
    }
    else {
        System.out.printf("Price: %.2f", price);
    }
}
else {
    price = tickets*4500;
    if(price>1000){
        total = 85*price/100;
        System.out.printf("Price: %.2f", total);
    }
    else {
        System.out.printf("Price: %.2f", price);
    }
}
else{
    System.out.println(tickets+" is an Invalid no. of tickets");
}
}
else{
    System.out.println(days+" is an Invalid no. of days");
}
}
else {
System.out.println(place+" is an Invalid place");
}
}
```

```

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:");

```



Edit with WPS Office

```
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));
}
```



Edit with WPS Office

# 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

```
<!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:#FFFFFF;
        margin-left: auto;
        margin-right: auto;
    }

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

```

```
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 Pedal 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()">

Name	<input id="name" required="" type="text"/>
Phone no	<input id="phno" required="" type="tel"/>
Email ID	<input id="email" required="" type="email"/>
Number of Persons	<input id="noOfPersons" required="" type="number"/>
Prefer Stay	<input id="yes" name="preferStay" onchange="disableNoOfDaysStay()" required="" type="radio" value="Yes"/> Yes <input id="no" name="preferStay" onchange="disableNoOfDaysStay()" required="" type="radio" value="No"/> No

```

</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:#FFAAAC;
    }

    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;
    }

</style>
</head>
<body>
<div>
<h3>Monthly Instalment Estimator</h3>
<form>
<input type="text" placeholder="Principal Amount" name="principal">
<input type="text" placeholder="Interest Rate" name="interest">
<input type="text" placeholder="Loan Term (in years)" name="term">
<input type="button" value="Estimate" onclick="calculate()"/>
</form>
</div>
</body>
</html>
```

```
margin-left: auto;
margin-right: auto;
background-color: #770080;
color: #FFFFFF;
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 ;
}

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

#submit{
    margin-right: auto;
    margin-left: auto;
    color: #FFFFFF;
    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=XXXXXXXX;">
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 validateCustomerId(){

        // Fill the code

        boolean b=true;

        String s1 = this.customerId.toUpperCase();

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

        if(s1.matches(regex)){

            b=true;

        }

        else{

            b=false;

        }

        return b;

    }

}
```

```
}

public double calculateDiscount(double purchaseAmount){

    // Fill the code

    double discount=purchaseAmount*0.45;

    double updateamount=purchaseAmount-discount;

    return updateamount;

}
```

}

GoldMembers.java

```
public class GoldMembers extends Members {

    public GoldMembers(String customerId,String customerName,long mobileNumber,String
memberType,String emailld){

        super(customerId,customerName,mobileNumber,memberType,emailld);

    }
```

// 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
```



```
        double discount=purchaseAmount*0.3;
        double updateamount=purchaseAmount-discount;
        return updateamount;
    }

}

UserInterface.java

import java.util.Scanner;

public class UserInterface {

    public static void main(String[] args){

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

```

        double res=0.0;

        if(d.validateCustomerId()){

            res= d.calculateDiscount(amount);

            System.out.println("Name :" +d.getCustomerName());

            System.out.println("Id :" +d.getCustomerId());

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

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

        } else if(g.validateCustomerId()){

            res= g.calculateDiscount(amount);

            System.out.println("Name :" +g.getCustomerName());

            System.out.println("Id :" +g.getCustomerId());

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

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

        } else if(p.validateCustomerId()){

            res= p.calculateDiscount(amount);

            System.out.println("Name :" +p.getCustomerName());

            System.out.println("Id :" +p.getCustomerId());

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

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

        } else{

            System.out.println("Provide a valid Customer Id");

        }

    }

    // Fill the code
}


```

**Batting Average**

```
UserInterface.java

package com.ui;

import com.utility.Player;

import java.util.ArrayList;

import java.util.Scanner;

public class UserInterface {

    public static void main(String[] args) {

        Scanner sc=new Scanner(System.in);

        Player player=new Player();

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

        boolean flag=true;

        while(flag)

        {

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

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

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

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

            int choice=sc.nextInt();

            switch(choice)

            {

                case 1: {

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

                    int runScored=sc.nextInt();

                    player.addScoreDetails(runScored);

                    break;

                }

                case 2: {

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

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

```

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

## Player.java

```
package com.utility;
```

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

```
public class Player {
```

private List<

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

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

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

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

}
```

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

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

For Example:

List contains[150,50,50]

average runs secured=(150+50+50)/3=83.33333333333333

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"));


```

```
        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");

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

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

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

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

}

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

protected final boolean validateClassName(String className) {

    boolean iscorrect = false;

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

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

```

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

return iscorrect;

}

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

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

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

            methodName = methodSignature[0];
            returnType = methodSignature[1];
            cls = Class.forName(className);

            Method[] methods = cls.getMethods();
            for (Method findMethod : methods) {
                if (methodName.equals(findMethod.getName())) {
                    foundMethod = true;
                }
            }
            if (!foundMethod) {
                errorFlag = true;
            }
        }
        if (errorFlag) {
            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");
        }
    } 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");
    }
}

```

```

        if
        (!(findMethod.getReturnType().getName().equals(returnType))) {

            errorFlag = true;

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

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

        }

    } else {

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

    }

}

if (!foundMethod) {

    errorFlag = true;

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

        + ". Do not change the " + "given public
method name. " + "Verify it with the skeleton");

}

}

if (!errorFlag) {

    LOG.info("Method signature is valid");

}

}

} catch (Exception e) {

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

        + "Method signature is same as the skeleton
before uploading");
}

```

```
    }
```

```
}
```

```
}
```

```
DatabaseUtil.java
```

```
package com.cts.paymentProcess.util;
```

```
import java.io.FileInputStream;
```

```
import java.io.IOException;
```

```
import java.io.InputStream;
```

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
```

```
import java.sql.SQLException;
```

```
import java.util.Properties;
```

```
import java.util.ResourceBundle;
```

```
public class DatabaseUtil {
```

```
    private DatabaseUtil() {
```

```
    }
```

```
    private static Connection con = null;
```

```
    private static Properties props = new Properties();
```

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

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

```
        try{
```

```
            FileInputStream fis = null;
```

```

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

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

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

}
}

```

App.java

```
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 rowsCount = 0;

    //Write your logic here...

    try{
        Connection con = DatabaseUtil.getConnection();
        for(Assessment a:assessments)
        {
            PreparedStatement st = con.prepareStatement("insert into assessment
values(?,?,?,?,?,?)");
            st.setString(1,a.getExamCode());
            st.setString(2,a.getExamTitle());
            st.setString(3,a.getExamDate().toString());
            st.setString(4,a.getExamTime().toString());
            st.setString(5,a.getExamDuration().toString());
            st.setString(6,a.getEvalDays().toString());
            int rs=st.executeUpdate();
            if(rs!=-1)
                rowsCount=rowsCount+1;
        }
    } catch(SQLException e){
    }

    return rowsCount;
}

public Assessment findAssessment(String code) throws Exception {
    Assessment assessment = null;
    Connection conn = DatabaseUtil.getConnection();
    String sql = "SELECT * FROM assessment where code=?";
    PreparedStatement ps = conn.prepareStatement(sql);
}

```

```

        ps.setString(1, code);

        ResultSet rs = ps.executeQuery();

        if(rs.next()) {

            assessment = new Assessment();

            assessment.setExamCode(rs.getString(1));

            assessment.setExamTitle(rs.getString(2));

            assessment.setExamDate(LocalDate.parse(rs.getString(3)));

            assessment.setExamTime(LocalTime.parse(rs.getString(4)));

            assessment.setExamDuration(Duration.parse(rs.getString(5)));

            assessment.setEvalDays(Period.parse(rs.getString(6)));

        }

        return assessment;
    }
}

```

### GenerateAssessmentFunction.java

```

package com.cts.cc.functions;

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

import com.cts.cc.model.Assessment;

public class GenerateAssessmentFunction implements Function<String, Assessment>{

    @Override
    public Assessment apply(String t) {

```

```
//Write your code here...
String temp[] = t.split(",");
Assessment a = new
Assessment(temp[0],temp[1],LocalDate.parse(temp[2]),LocalTime.parse(temp[3]),Duration.parse(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"),
    props.getProperty("DB_PASSWORD"));
}

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

return con;
}

}

FileUtil.java

package com.cts.cc.util;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;
import java.util.function.Function;
import java.util.stream.Collectors;
import java.util.stream.Stream;
import java.io.*;
```

```
import java.util.*;

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

public class FileUtils {

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

    return list;
}

}
```

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

```
import java.util.logging.Logger;

import com.cts.cc.model.Assessment;

public class SkeletonValidator {

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

    public SkeletonValidator() {

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

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

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

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

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

```

        testMethods(assessmentDAOClass, "findAssessment", new Class[] { String.class },
Assessment.class);

        testMethods(funritionalClass, "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");
    }
}
}

```

### **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;

    }

}
```

```
 }  
  
 }
```

### 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");
    }
}

```

## 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])/([10-2]|0[1-9])/[0-9]{4}$";
    Pattern pattern=Pattern.compile(regex);
    Matcher matcher=pattern.matcher((CharSequence)date);
    return matcher.matches();
}

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

public static String futureDate(String date,int days){

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

```

**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"));

            con = DriverManager.getConnection("jdbc:mysql://localhost:3306/travelrequest", "root", "root");
        }
        catch(Exception e){
            e.printStackTrace();
        }
    }

}
```

```

        // 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,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 (!foundMethod) {
                errorFlag = true;
            }
        }
        if (errorFlag) {
            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");
        }
    }
}

```

```

        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 destinationCity  
 */  
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]==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");
}
}
}

```

### 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[]={};  
        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];  
                }  
            }  
        }  
        System.out.println("The maximum difference is "+max+" at index "+n1);  
    }  
}
```

```

    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,getBlockListNumb  
ers>List","com.cts.services.CreditCardService");  
    }  
}
```

```

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

protected final boolean validateClassName(String className) {

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

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

```

```

String[] methodSignature;
String methodName = null;
String returnType = null;

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

```

```

        + ". Do not change the " + "given public
method name. " + "Verify it with the skeleton");

    }

}

if (!errorFlag) {

    LOG.info("Method signature is valid");

}

}

} catch (Exception e) {

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

}

}

CreditCardValidatorMain

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);

}
}

```

### Eshooping

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 linelitem
 * @return boolean
 */

public boolean isBulkOrder(OrderLineItem linelitem) {

boolean result=false;

if(linelitem.quantity>5){

    result = true;

}

else if(linelitem.quantity<=5 && linelitem.quantity>=1){

    result=false;

}

return result; // TODO change this return value

}

/***
 * Count the number of line items which are ordered in bulk
 *
 */

```

```
* @param orderLineItems
* @return
*/
public int countOfBulkOrderLineItems(OrderLineItem[] orderLineItems) {
    int count = 0;

    for(int i=0;i<orderLineItems.length;i++){
        if(isBulkOrder(orderLineItems[i])){
            count++;
        }
    }

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

```
}
```

SkeletonValidator.java

```
package com.cts.eshopping.skeletonvalidator;
```

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

```
/***
 * @author 222805
 *
 * This class is used to verify if the Code Skeleton is intact and not modified by participants thereby
 * ensuring smooth auto evaluation
 *
 */
public class SkeletonValidator {
```

```

public SkeletonValidator() {
    validateClassName("com.cts.eshopping.orderservice.CartService");
    validateClassName("com.cts.eshopping.vo.OrderLineItem");
    validateMethodSignature(
        "calculateOrderTotalAmount:double,calculateDiscount:double,isBulkOrder:boolean,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 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");
}
}

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 grade obtained is A, its equivalent grade points is 9 has to be added into the gradePointList\*/

```

public void addGradePoint(char gradeObtained) {

    if(gradeObtained == 'S')
    {
        gradePointList.add(10);
    }
    else if(gradeObtained == 'A')
    {

```

```

        gradePointList.add(9);

    }

    else if(gradeObtained == 'B')
    {

        gradePointList.add(8);

    }

    else if(gradeObtained == 'C')
    {

        gradePointList.add(7);

    }

    else if(gradeObtained == 'D')
    {

        gradePointList.add(6);

    }

    else
    {

        gradePointList.add(5);

    }

}

```

/\* This method should return the GPA of all grades scored in the semester  
GPA can be calculated based on the following formula  
GPA= (gradePoint1 + gradePoint2 + ... + gradePointN) / (size of List)

For Example:

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

GPA = (9 + 10 + 8 + 5) / (4)= 8.0

\*/

```
public double calculateGPAScored() {
```

```

        double gpa=-1;

        double total=0,value=0,size=0;

        size = gradePointList.size();

        if(size < 1)

        {

            return 0;

        }

        // fill the code

        Iterator i = gradePointList.iterator();

        while(i.hasNext())

        {

            value = (Integer)i.next();

            total += value;

        }

        gpa = total/size;

        return gpa;

    }

}

```

### **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.getHouseholdValuation();

    int propertyAge = LocalDate.now().getYear()-propertyDetails.getBuiltYear();
    propertyDetails.setSumAssured(sumAssured);
}

```

```

        double premium = 0;
        if(propertyAge>15) {
            premium =
            Constants.MIN_PREMIUM_AMOUNT+(propertyDetails.getSumAssured()*0.35);
        }
        else if(propertyAge>=6) {
            premium =
            Constants.MIN_PREMIUM_AMOUNT+(propertyDetails.getSumAssured()*0.2);
        }
        else {
            premium =
            Constants.MIN_PREMIUM_AMOUNT+(propertyDetails.getSumAssured()*0.1);
        }
        return premium;
    }

```

```

    public double calculatePremiumForBurglaryCoverage(PropertyDetails propertyDetails,
double amount) {
    //write your code here based on business rules
    if(propertyDetails.getBurglaryCoverReqd().equalsIgnoreCase(Constants.YES)) {
        return amount*.01;
    }
    return 0;
}

```

```

    public double calculatePremiumForPoliticalUnrestCoverage(PropertyDetails propertyDetails,
double amount) {
    //Write your code here based on business rules
    //Ex:-
    propertyDetails.getPoliticalUnrestCoverReqd().equalsIgnoreCase(Constants.YES) to check condition
    if(propertyDetails.getPoliticalUnrestCoverReqd().equalsIgnoreCase(Constants.YES)) {
        return amount*.01;
    }
}

```

```
        return 0;

    }

}

SkeletonValidator.java

package com.cts.insurance.skeleton;

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

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

    public SkeletonValidator() {
        validateClassName("com.cts.insurance.entity.PropertyDetails");
        validateClassName("com.cts.insurance.misc.Constants");
        validateClassName("com.cts.insurance.services.CalculatePremiumService");
        validateClassName("com.cts.insurance.InsurancePremiumGeneratorApp");
        validateMethodSignature(
            "checkOwnerDetails:boolean,getPremiumAmount:double,validatePropertyParameters:boolean,calculatePremiumByPropertyAge:double,calculatePremiumForBurglaryCoverage:double,calculatePremiumForPoliticalUnrestCoverage:double","com.cts.insurance.services.CalculatePremiumService");
    }
}
```

```

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

protected final boolean validateClassName(String className) {

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

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

```

```

String[] methodSignature;
String methodName = null;
String returnType = null;

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

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

```

```

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

                + ". Do not change the " + "given public
method name. " + "Verify it with the skeleton");

        }

    }

    if (!errorFlag) {

        LOG.info("Method signature is valid");

    }

}

} catch (Exception e) {

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

            + "Method signature is same as the skeleton
before uploading");

    }

}

}

```

```

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);

        }

    }

}

```

### 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 getemailld(){

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 mnmm=sc.next();

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

int tno=sc.nextInt();

System.out.println("Do you want AC or not");

String choice =sc.next();

if(tid.contains("PLATINUM")){

PlatinumTicket PT=new PlatinumTicket(tid,cnm,mno,email,mnmm);

boolean b1=PT.validateTicketId();

if(b1==true){

double cost =PT.calculateTicketCost(tno, choice);

System.out.println("Ticket cost is "+ cost);

}

else if(b1==false){

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

System.exit(0);

}

}

else if(tid.contains("GOLD")){

GoldTicket GT=new GoldTicket(tid,cnm,mno,email,mnmm);

boolean b2=GT.validateTicketId();

if(b2==true){

double cost=GT.calculateTicketCost(tno, choice);

System.out.println("Ticket cost is "+cost);

}

else if (b2==false){
```

```
System.out.println("Provide valid Ticket Id");

System.exit(0);

}

}

else if(tid.contains("SILVER")){

SilverTicket ST=new SilverTicket(tid,cnm,mno,email,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 iscorrect = false;
    try {
        Class.forName(className);
        iscorrect = true;
        LOG.info("Class Name " + className + " is correct");

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

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

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

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

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

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

```

```

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

```

#### USER INTERFACE-

```

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

public class UserInterface{
    public static PassengerCategorization searchPassengerRecord(){

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

if((pass.getSource().toLowerCase()).equals(source.toLowerCase())){

```

```

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

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

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

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

```

ZEE LAPTOP AGENCY

INVALID LAPTOP-

package com.cts.zeelaptopagency.exception;

```

public class InvalidLaptopIdException extends Exception{
    public InvalidLaptopIdException() {
    }

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

}

```

#### MAIN.JAVA-

```

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

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

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

        //Add your code here to retreive file object from Service
        class
        //Add Code here to print valid LaptopDetails returned by
        Service Method
            LaptopService l=new LaptopService();
            File f=l.accessFile();
            List<Laptop> lap=l.readData(f);
            System.out.println("The Valid Laptop Details are:-");
            for(Laptop la:lap)
            {
                try{
                    if(l.validate(la.getLaptopId())==true){
                        System.out.println(la.toString());
                    }
                }
                catch(InvalidLaptopIdException e)
                {
                    e.printStackTrace();
                }
            }
        }

    }
}

```

#### LAPTOP SERVICE.JAVA-

```

package com.cts.zeelaptopagency.service;

```

```

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

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

public class LaptopService {

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

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

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

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

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

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

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

```

```

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

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

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

return lap; //TODO change this return value
}

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

```

```
    }  
}  
  
}
```

#### SKELETON VALIDATOR-

```
package com.cts.zeelaptopagency.skeletonvalidator;  
  
import java.lang.reflect.Method;  
import java.util.logging.Level;  
import java.util.logging.Logger;  
  
public class SkeletonValidator {  
    public SkeletonValidator() {  
        validateClassName("com.cts.zeelaptopagency.service.LaptopService");  
        validateClassName("com.cts.zeelaptopagency.vo.Laptop");  
        validateMethodSignature(  
            "accessFile:java.io.File,validate:boolean,readData:java.util.List",  
            "com.cts.zeelaptopagency.service.LaptopService");  
    }  
  
    private static final Logger LOG =  
        Logger.getLogger("SkeletonValidator");  
    protected final boolean validateClassName(String className) {  
        boolean iscorrect = false;  
        try {  
            Class.forName(className);  
            iscorrect = true;  
            LOG.info("Class Name " + className + " is correct");  
        } catch (ClassNotFoundException e) {  
            LOG.log(Level.SEVERE, "You have changed either the " +  
                "class name/package. Use the correct package "  
                + "and class name as provided in the  
                skeleton");  
        } catch (Exception e) {  
            LOG.log(Level.SEVERE,  
                "There is an error in validating the " + "Class  
                Name. Please manually verify that the "  
                + "Class name is same as skeleton  
                before uploading");  
        }  
        return iscorrect;  
    }  
  
    protected final void validateMethodSignature(String methodWithExcptn,  
        String className) {
```

```

        Class cls = null;
        try {

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

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

                methodName = methodSignature[0];
                returnType = methodSignature[1];
                cls = Class.forName(className);
                Method[] methods = cls.getMethods();
                for (Method findMethod : methods) {
                    if (methodName.equals(findMethod.getName())) {
                        foundMethod = true;
                        if (!
(findMethod.getReturnType().getName()).equals(returnType)) {
                            errorFlag = true;
                            LOG.log(Level.SEVERE, " You have
changed the " + "return type in '" + methodName
+ "' method. Please
stick to the " + "skeleton provided");
                        } else {
                            LOG.info("Method signature of " +
methodName + " is valid");
                        }
                    }
                }
                if (!foundMethod) {
                    errorFlag = true;
                    LOG.log(Level.SEVERE, " Unable to find the
given public method " + methodName
+ ". Do not change the " + "given
public method name. " + "Verify it with the skeleton");
                }
            }
            if (!errorFlag) {
                LOG.info("Method signature is valid");
            }
        } catch (Exception e) {
            LOG.log(Level.SEVERE,
                    " There is an error in validating the " +
"method structure. Please manually verify that the "
+ "Method signature is same as the
skeleton before uploading");
        }
    }
}

```

LAPTOP.JAVA-

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

    public Laptop()
    {
    }

    public String toString()
    {
        return "Laptop [laptopId="+this.getLaptopId()+",
customerName="+this.getCustomerName()+", basicCost="+this.getBasicCost()+",
noOfDays="+this.getNoOfDays()+", totalAmount="+this.getTotalAmount()+"]";
    }
    public String getLaptopId() {
        return laptopId;
    }

    public void setLaptopId(String laptopId) {
        this.laptopId = laptopId;
    }
    public String getCustomerName() {
        return customerName;
    }
    public void setCustomerName(String customerName) {
        this.customerName = customerName;
    }
    public double getBasicCost() {
        return basicCost;
    }
    public void setBasicCost(double basicCost) {
        this.basicCost = basicCost;
    }
    public int getNoOfDays() {
        return noOfDays;
    }
    public void setNoOfDays(int noOfDays) {
        this.noOfDays = noOfDays;
    }
    public double getTotalAmount() {
        return totalAmount;
    }
    public void setTotalAmount(double totalAmount) {
```

```
        this.totalAmount = totalAmount;
    }
```

```
}
```

#### LAPTOP DETAILS-

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

#### DOLLAR CITY THEME PARK

#### USER INTERFACE-

```
package com.ui;  
  
import java.util.Scanner;  
  
import com.utility.ThemeParkBO;  
  
public class UserInterface {  
    public static void main(String[] args) {  
        Scanner sc = new Scanner(System.in);  
  
        // Fill the UI code  
        boolean flag = true;  
        int choice = 0;  
        ThemeParkBO park = new ThemeParkBO();  
        while(flag){  
  
            System.out.println("1.Add booking details");  
            System.out.println("2.Average customer booked");  
            System.out.println("3.Exit");  
            System.out.println("Enter your choice");  
            choice = sc.nextInt();  
  
            switch(choice){  
                case 1:  
                    System.out.println("Enter the day");  
                    String day = sc.next();  
                    System.out.println("Enter the customer count");  
                    int cc = sc.nextInt();  
                    park.addBookingDetails(cc);  
  
                    break;  
                case 2:  
                    double res = park.findAverageCustomerBooked();  
                    if(res==0){
```

```

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

}
}

```

#### THEMEPARKBO.JAVA-

```

package com.utility;

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

public class ThemeParkBO {

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

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

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

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

    public void addBookingDetails(int customerCount) {

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

    }

    /*
     * This method should return the average customer booked based on the
     * customerCount values available in the bookingList.
     */
    public double findAverageCustomerBooked() {
        double avg;

        // Fill the Code here
        double count = 0;

```

```

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

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

```

## PASSENGER

### PASSENGER UTILITY-

```

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

public class PassengerUtility {

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

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

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

        return list;
    }

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

```

```
        }
    return false;
}

}
```

#### PASSENGER CATEGORIZATION-

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

}
```

#### PASSENGER SKELETON-

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

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

    public SkeletonValidator() {

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

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

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

```

protected final boolean validateClassName(String className) {

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

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

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

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

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

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

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

                } else {
                    LOG.info("Method signature of " +
methodName + " is valid");
                }
            }
        }
    }
}

```

```

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

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

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

```

#### PASSENGER USER INTERFACE-

```

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

public class UserInterface{
    public static PassengerCategorization searchPassengerRecord(){

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

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

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

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

```

```

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

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

```

#### INVALID CARRIER EXEMPTION -

```

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

```

#### EMPLOYEE SALARY

##### EMPLOYEE-

```

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

```

```

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

```

#### MAIN .JAVA-

```

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

```

```

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

```

## HOME APPLIANCES

### USER INTERFACE-

```

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

```

```

        case "LEDTV":
        {
            System.out.println("Enter Batch Id");
            String batch = sc.nextLine();
            System.out.println("Enter Dispatch
date");
            String date = sc.nextLine();
            System.out.println("Enter Warranty
Years");

            inches");
            date, years, size, quality);
            ob2.calculateProductPrice();
            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");

            MicrowaveOven(id, name, batch, date, years,
            quantity, quality);
            ob3.calculateProductPrice();
            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 mnmm=sc.next();
System.out.println("Enter number of tickets");
int tno=sc.nextInt();
System.out.println("Do you want AC or not");
String choice =sc.next();
if(tid.contains("PLATINUM")){
PlatinumTicket PT=new PlatinumTicket(tid,cnm,mno,email,mnmm);
boolean b1=PT.validateTicketId();
if(b1==true){
double cost =PT.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,mnmm);
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,mnmm);
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");

}

}

}
```