# Rajalakshmi Engineering College

Name: Shreenidhi T
Email: 240701503@rajalakshmi.edu.in
Roll no: 240701503
Phone: 9150942326
Branch: REC
Department: CSE - Section 10
Batch: 2028
Degree: B.E - CSE

Scan to verify results

## 2024_28_III_OOPS Using Java Lab

## REC_2028_OOPS using Java_Week 7_CY

Attempt : 1
Total Mark : 40
Marks Obtained : 37.5

## Section 1 : Coding

1. Problem Statement

John is developing a car loan calculator and has structured his program using two interfaces, Principal and InterestRate, defining methods for principal and interest rate retrieval.

The Loan class implements these interfaces, taking principal and annual interest rates as parameters. User input is solicited for these values, and the program ensures their validity before performing calculations. If input values are invalid (less than or equal to zero), an error message is displayed.

Note: Total interest = principal * interest rate * years

***Input Format***

The first line of input consists of a double value P, representing the principal.

The second line consists of a double value R, representing the annual interest rate.

The third line consists of an integer value N, representing the loan duration in years.

*Output Format*

If the input values are valid, print "Total interest paid: Rs. " followed by a double value, representing the total interest paid, rounded off to two decimal places.

If the input values are invalid (negative or zero values for principal, annual interest rate, or loan duration), print "Invalid input values!".

Refer to the sample output for formatting specifications.

*Sample Test Case*

Input: 20000.00
0.05
5
Output: Total interest paid: Rs.5000.00

*Answer*

```
import java.util.Scanner;


interface Principal {
   double getPrincipal();
}

interface InterestRate {
   double getInterestRate();
}

class Loan implements Principal, InterestRate {
   private double principal;
   private double rate;
```

```
    public Loan(double principal, double rate) {
        this.principal = principal;
        this.rate = rate;
    }

    public double getPrincipal() {
        return principal;
    }

    public double getInterestRate() {
        return rate;
    }

    public double calculateTotalInterest(int years) {
        return principal * rate * years;
    }
}

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

        double carPrice = scanner.nextDouble();

        double annualInterestRate = scanner.nextDouble();

        int loanDuration = scanner.nextInt();

        if (carPrice <= 0 || annualInterestRate <= 0 || loanDuration <= 0) {
            System.out.println("Invalid input values!");
            return;
        }

        Loan carLoan = new Loan(carPrice, annualInterestRate);
        double totalInterest = carLoan.calculateTotalInterest(loanDuration);

        System.out.printf("Total interest paid: Rs.%.2f%n", totalInterest);
    }
}
```

***Status :*** Correct ***Marks : 10/10***

2. Problem Statement

Jeevan is developing a fitness-tracking application to monitor daily physical activity.

The application incorporates a FitnessTracker class that implements two interfaces: StepCounter for tracking the number of steps taken and CalorieCalculator for estimating total calories burned based on total steps.

Jeevan needs your help creating a program.

Note

The calorie calculation formula is: Total caloriesBurned = (total steps / 100.0) * 20.0.

***Input Format***

The first line of input is an integer n, representing the number of days Jeevan wants to input data.

The second line consists of space-separated integers, representing the number of steps Jeevan took on each day.

***Output Format***

The first line of output prints: "Total Steps: <totalSteps>", where '<totalSteps>' is the sum of steps (integer) taken over 'n' days.

The second line prints: "Calories Burned: <caloriesBurned>", where '<caloriesBurned>' is the estimated total calories (double-point number) burned based on the total steps taken rounded off to two decimal places.

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: 3
340 234 987
Output: Total Steps: 1561

Calories Burned: 312.20

***Answer***

```
import java.util.Scanner;


interface StepCounter {
   void countSteps(int steps);
   int getTotalSteps();
}

interface CalorieCalculator {
   double calculateCaloriesBurned(int totalSteps);
}

class FitnessTracker implements StepCounter, CalorieCalculator {
   private int totalSteps = 0;

   public void countSteps(int steps) {
      totalSteps += steps;
   }

   public int getTotalSteps() {
      return totalSteps;
   }

   public double calculateCaloriesBurned(int totalSteps) {
      return (totalSteps / 100.0) * 20.0;
   }
}

class Main
{

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

      FitnessTracker tracker = new FitnessTracker();

      int n = scanner.nextInt();

      for (int i = 0; i < n; i++) {
         int steps = scanner.nextInt();
```

```
            tracker.countSteps(steps);
        }

        int totalSteps = tracker.getTotalSteps();
        System.out.println("Total Steps: " + totalSteps);

        double caloriesBurned = tracker.calculateCaloriesBurned(totalSteps);
        System.out.printf("Calories Burned: %.2f%n", caloriesBurned);

        scanner.close();
    }
}
```

***Status :*** Correct ***Marks : 10/10***

3. Problem Statement:

Rathish is planning a road trip and needs a program to convert speeds between miles per hour (MPH) and kilometers per hour (KPH).

Create an interface, SpeedConverter, with a method convertSpeed(double mph). Implement the interface with MPHtoKPHConverter class, allowing Rathish to input MPH and receive the converted speed in KPH, rounded to two decimal points.

Formula: Speed in KPH = 1.60934 * Speed in MPH.

***Input Format***

The input consists of a single double-point number representing the speed in miles per hour (MPH).

***Output Format***

The output displays the converted speed (double-point number) in kilometers per hour (KPH) rounded off to two decimal points in the following format:

"Speed in KPH: <<converted speed>>".

Refer to the sample output for the formatting specifications.

***Sample Test Case***

Input: 1.0
Output: Speed in KPH: 1.61

***Answer***

```
import java.util.Scanner;

// You are using Java
interface SpeedConverter {
   double convertSpeed(double mph);
}

class MPHtoKPHConverter implements SpeedConverter {
   public double convertSpeed(double mph) {
      return 1.60934 * mph;
   }
}

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

      double speedInMPH = scanner.nextDouble();

      SpeedConverter converter = new MPHtoKPHConverter();

      double speedInKPH = converter.convertSpeed(speedInMPH);

      System.out.printf("Speed in KPH: %.2f\n", speedInKPH);

      scanner.close();
   }
}
```

***Status :*** Correct ***Marks : 10/10***

4. Problem Statement:

Ray is developing a tax calculation program in Java. The program includes an interface named TaxCalculator with a method to calculate tax based on salary. The SimpleTaxCalculator class implements this interface and determines the tax to be paid based on the salary amount using progressive tax slabs.

Your task is to implement this system. The program first takes an integer T representing the number of test cases, followed by T salary values. For each salary, calculate the total tax to be paid based on the following progressive tax rules:

For the first  50,000 of salary, the tax rate is 5%.For the next  50,000 (i.e., from  50,001 to  1,00,000), the tax rate is 10%.For any amount above  1,00,000, the tax rate is 20%. (That is, only the amount above  1,00,000 is taxed at 20%.)

Example

Input

3

78000

110000

23000

Output

5300

9500

1150

Explanation

For Salary Rs. 78,000

Tax =  0.1 * (78,000 - 50,000) + 0.05 * 50,000 = 5,300

For Salary Rs. 1,10,000

Tax = 0.2 * (110000 - 100000)+ 0.1 * 50,000 + 0.05 * 50,000 = 9,500

For Salary Rs. 23,000

Tax = 0.05 * 23,000 = 1,150

*Input Format*

The first line of the input consists of an integer, T, representing the number of test cases.

The next T lines of the input consist of a single integer, representing the annual salary of an individual, separated by a line.

*Output Format*

The output displays the calculated tax as an integer for each test case, separated by a line.

Refer to the sample output for the formatting specifications.

*Sample Test Case*

Input: 2
100
300

Output: 5
15

*Answer*

```
import java.util.Scanner;


interface TaxCalculator {
   int calculateTax(int salary);
}

class SimpleTaxCalculator implements TaxCalculator {

   public int calculateTax(int salary) {
      double tax = 0.0;

      if (salary <= 50000) {
         tax = 0.05 * salary;
```

```
        } else if (salary <= 100000) {
            tax = (0.05 * 50000) + (0.10 * (salary - 50000));
        } else {
            tax = (0.05 * 50000) + (0.10 * 50000) + (0.20 * (salary - 100000));
        }

        return (int) Math.round(tax);
    }
}

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

        TaxCalculator taxCalculator = new SimpleTaxCalculator();

        for (int i = 0; i < T; i++) {
            int salary = scanner.nextInt();
            int tax = taxCalculator.calculateTax(salary);
            System.out.println(tax);
        }

        scanner.close();
    }
}
```

***Status :*** Partially correct ***Marks : 7.5/10***