Rajalakshmi Engineering College

Name: Kaaviya Sri PS
Email: 240701222@rajalakshmi.edu.in
Roll no: 240701222
Phone: 8838174850
Branch: REC
Department: CSE - Section 6
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 : 40

Section 1 : Coding

1. Problem Statement:

Sam is developing a geometry application and needs a class for trapezoid calculations. Create a "Trapezoid" class implementing a "ShapeInput" interface with a method to input trapezoid dimensions.

Also, implement a "ShapeCalculator" interface with methods to compute area and perimeter. In the "Main" class, instantiate Trapezoid, gather user input, and display the calculated area and perimeter with two decimal places.

Note

Area of Trapezoid = (1/2) * (base1 + base2) * height

Perimeter of Trapezoid = base1 + base2 + side1 + side2

240701222

Input Format

The first line of input is a double-point value representing base1 of the trapezoid.

The second line of input is a double-point value representing base2 of the trapezoid.

The third line of input is a double-point value representing the height of the trapezoid.

The fourth line of input is a double-point value representing side1 of the trapezoid.

The fifth line of input is a double-point value representing side2 of the trapezoid.

Output Format

The output displays the two lines of the calculated area (double type) and perimeter (double type) of the trapezoid, each rounded to two decimal places in the following format:

"Area of the Trapezoid: <<calculated area>>".

Perimeter of the Trapezoid: <<calculated perimeter>>".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 1.0
2.0
1.0
3.0
1.0

Output: Area of the Trapezoid: 1.50
Perimeter of the Trapezoid: 7.00

Answer

240701222

```
import java.util.Scanner;

interface ShapeInput {
   void getInput();
}

interface ShapeCalculator {
   double calculateArea();
   double calculatePerimeter();
}

class Trapezoid implements ShapeInput, ShapeCalculator {
   private double base1;
   private double base2;
   private double height;
   private double side1;
   private double side2;

   @Override
   public void getInput() {
      Scanner sc = new Scanner(System.in);
      base1 = sc.nextDouble();
      base2 = sc.nextDouble();
      height = sc.nextDouble();
      side1 = sc.nextDouble();
      side2 = sc.nextDouble();
   }

   @Override
   public double calculateArea() {
      return 0.5 * (base1 + base2) * height;
   }

   @Override
   public double calculatePerimeter() {
      return base1 + base2 + side1 + side2;
   }
}

public class Main {
   public static void main(String[] args) {
      Trapezoid trapezoid = new Trapezoid();
      trapezoid.getInput();
```

240701222

```
        double area = trapezoid.calculateArea();
        double perimeter = trapezoid.calculatePerimeter();

        System.out.println("Area of the Trapezoid: " + String.format("%.2f", area));
        System.out.println("Perimeter of the Trapezoid: " + String.format("%.2f",
perimeter));
    }
}
```

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

2. 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

240701222

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

240701222

Answer

```
import java.util.Scanner;

import java.util.Scanner;

interface TaxCalculator {
   int calculateTax(int salary);
}

class SimpleTaxCalculator implements TaxCalculator {
   @Override
   public int calculateTax(int salary) {
      int tax = 0;

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

      return 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();
   }
}
```

240701222

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

3. 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

240701222

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;

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

   @Override
   public int getTotalSteps() {
      return totalSteps;
   }

   @Override
   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();
```

240701222 240701222 240701222 240701222

```
        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***

4. Problem Statement

A developer aims to create a budget management system using two interfaces, ExpenseRecorder for recording expenses and BudgetCalculator for calculating remaining budgets.

The ExpenseTracker class implements these interfaces, allowing users to input an initial budget and record expenses iteratively until entering 0.0 as a sentinel value.

The program then computes and displays the remaining budget or notifies of budget exceedance.

Example

Input

100.0

20.0 30.0 10.0 0.0

Output

240701222

Remaining budget: Rs. 40.00

Explanation

The initial budget is 100.0. Expenses of 20.0, 30.0, and 10.0 are recorded.

Remaining budget is calculated (100.0 - 20.0 - 30.0 - 10.0 = 40.0).

Input Format

The first line of input is the initial budget as a double-point number (double type). The budget is a positive number.

The second line of input consists of individual expenses as double-point numbers. Each expense is separated by space.

To end the input, an expense of 0.0 is used.

Output Format

The output displays the remaining budget, formatted to two decimal places, in the following format:

If the remaining budget (double type) is non-negative, it prints "Remaining budget: Rs. [remainingBudget]".

If the remaining budget is negative, it prints "No remaining budget, You've exceeded your budget!".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 100.0
20.0 30.0 10.0 0.0

Output: Remaining budget: Rs. 40.00

Answer

import java.util.Scanner;

240701222

```
interface ExpenseRecorder {
   void recordExpense(double amount);
}

interface BudgetCalculator {
   double calculateRemainingBudget();
}

class ExpenseTracker implements ExpenseRecorder, BudgetCalculator {
   private double initialBudget;
   private double totalExpenses;

   public ExpenseTracker(double initialBudget) {
      this.initialBudget = initialBudget;
      this.totalExpenses = 0.0;
   }

   @Override
   public void recordExpense(double amount) {
      if (amount != 0.0) {
         totalExpenses += amount;
      }
   }

   @Override
   public double calculateRemainingBudget() {
      return initialBudget - totalExpenses;
   }
}
class Main {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      double budget = scanner.nextDouble();


      ExpenseTracker tracker = new ExpenseTracker(budget);

      double expense;
      do {
         expense = scanner.nextDouble();
         tracker.recordExpense(expense);
      } while (expense != 0.0);
```

240701222

```
        double remainingBudget = tracker.calculateRemainingBudget();
        if (remainingBudget >= 0) {
            System.out.printf("Remaining budget: Rs. %.2f", remainingBudget);
        } else {
            System.out.println("No remaining budget, You've exceeded your
budget!");
        }
    }
}
```

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

240701222