Rajalakshmi Engineering College

Name: Mohamed Yahya A
Email: 240701325@rajalakshmi.edu.in
Roll no: 240701325
Phone: 9600561844
Branch: REC
Department: CSE - Section 9
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 : 2
Total Mark : 40
Marks Obtained : 40

Section 1 : Coding

1. 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 {
   int getTotalSteps(int[] steps);
}

interface CalorieCalculator {
   double calculateCalories(int totalSteps);
}

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

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

```
    }

    public int getTotalSteps() {
        return totalSteps;
    }

    public int getTotalSteps(int[] steps) {
        int total = 0;
        for (int s : steps) total += s;
        return total;
    }

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

    public double calculateCaloriesBurned(int totalSteps) {
        return calculateCalories(totalSteps);
    }
}
```

```
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***

2. 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

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;

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

```
interface BudgetCalculator {
   double calculateRemainingBudget();
}

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

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

   public void recordExpense(double expense) {
      totalExpenses += expense;
   }

   public double calculateRemainingBudget() {
      return budget - 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);

      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***

3. 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

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

```
import java.util.Scanner;

// You are using Java
import java.util.*;

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;

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

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

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

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

        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***

4. 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;

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***