Rajalakshmi Engineering College

Name: Mithran VT
Email: 240701312@rajalakshmi.edu.in
Roll no:
Phone: 9952919350
Branch: REC
Department: CSE - Section 8
Batch: 2028
Degree: B.E - CSE

Scan to verify results

2024_28_III_OOPS Using Java Lab

REC_2028_OOPS using Java_Week 6_PAH

Attempt : 1
Total Mark : 40
Marks Obtained : 0

Section 1 : Coding

1. Problem Statement

In a company, each manager has a unique employee ID and a monthly salary. You are required to design a program that will calculate and display the annual(12 months) salary of a manager based on the input details provided by the user.

Implement the solution using a single inheritance approach.

Employee: The base class with attributes name and employeeID.

Manager: The derived class inheriting from Employee, with an additional attribute salary.

Input Format

The first line of input consists of a string name, representing the manager's name.

The second line of input consists of an integer employeeID, representing the manager's employee ID.

The third line of input consists of a double salary, representing the manager's monthly salary.

Output Format

The first line of output prints: Name: <name>

The second line of output prints: Annual Salary: Rs. <annual_salary> (rounded to two decimal places).

Refer to the sample output for formatting specifications.

Sample Test Case

Input: Davis
234
28750.75

Output: Name: Davis
Annual Salary: Rs. 345009.00

Answer

-

Status : Skipped ***Marks : 0/10***

2. Problem Statement

Sharon, a software developer, is working on a project to automate velocity calculations for various objects. She wants to create a class named VelocityCalculator with overloaded methods calculateVelocity to calculate the velocity. One method will accept distance in meters and time in seconds as integers, while another will accept distance and time as doubles.

Help her in completing the project.

Formula: Velocity = distance / time

Input Format

The first line of input consists of an integer, representing the distance in meters (for the integer method).

The second line consists of an integer, representing the time in seconds (for the integer method).

The third line consists of a double value, representing the distance in meters (for the double method).

The fourth line consists of a double value, representing the time in seconds (for the double method).

Output Format

The first line prints the velocity calculated using the integer inputs in the format:

Velocity with integer inputs: <velocity> m/s

The second line prints the velocity calculated using the double inputs in the format:

Velocity with double inputs: <velocity> m/s

Note:

The velocity for the double inputs should be printed with two decimal places.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 100
10
100.5

10.2

Output: Velocity with integer inputs: 10 m/s
Velocity with double inputs: 9.85 m/s

Answer

```
import java.util.Scanner;

// You are using Java
class VelocityCalculator {
    public int calculateVelocity(int distance, int time) {
        return distance / time;
    }

    public double calculateVelocity(double distance, double time) {
        return distance / time;
    }
}

Scanner scanner = new Scanner(System.in);

int distanceInt = scanner.nextInt();
int timeInt = scanner.nextInt();
double distanceDouble = scanner.nextDouble();
double timeDouble = scanner.nextDouble();

VelocityCalculator vc = new VelocityCalculator();

int velocityInt = vc.calculateVelocity(distanceInt, timeInt);
double velocityDouble = vc.calculateVelocity(distanceDouble, timeDouble);

System.out.println("Velocity with integer inputs: " + velocityInt + " m/s");
System.out.printf("Velocity with double inputs: %.2f m/s", velocityDouble);

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

        int distanceInt = scanner.nextInt();
        int timeInt = scanner.nextInt();

        double distanceDouble = scanner.nextDouble();
        double timeDouble = scanner.nextDouble();
```

```
        int velocityInt = VelocityCalculator.calculateVelocity(distanceInt, timeInt);
        double velocityDouble =
VelocityCalculator.calculateVelocity(distanceDouble, timeDouble);

        System.out.println("Velocity with integer inputs: " + velocityInt + " m/s");
        System.out.printf("Velocity with double inputs: %.2f m/s", velocityDouble);

        scanner.close();
    }
}
```

Status : Wrong ***Marks : 0/10***

3. Problem Statement

John is planning a long road trip and wants to calculate the distance his car can travel based on its speed and fuel capacity. As John knows that different cars have different fuel efficiencies, he wants a program that can help him estimate the travel distance for any given car.

To do this, you are tasked with creating a program that calculates the travel distance of a car based on its speed and fuel capacity. The calculation is simple and follows the formula:

Travel Distance = Speed * Fuel Capacity

You need to model this system using a Vehicle class and a Car class. The Vehicle class will have attributes for the speed and fuel capacity, while the Car class will inherit from the Vehicle class and include a method to calculate the travel distance.

Input Format

The first line of input consists of a double value representing the speed of the car in km/h.

The second line of input consists of a double value representing the fuel capacity of the car in liters.

Output Format

The first line should print "Speed: X km/h", where X is the speed of the car, rounded to two decimal places.

The second line should print "Fuel Capacity: Y liters", where Y is the fuel capacity of the car, rounded to two decimal places.

The third line should print "Travel Distance: Z km", where Z is the total travel distance the car can cover, rounded to two decimal places.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 10.0
1.0

Output: Speed: 10.00 km/h
Fuel Capacity: 1.00 liters
Travel Distance: 10.00 km

Answer

-

Status : Skipped ***Marks : 0/10***

4. Problem Statement

Ram is designing a program to calculate the Body Mass Index (BMI). Your task is to assist him by following the given specifications.

Create a base class BMIcalculator with a method calculateBMI() to compute BMI using the formula weight / (height * height).

Extend the class with a subclass CustomBMIcalculator that overrides the method calculateBMI() to calculate BMI based on custom criteria, assigning categories such as "Underweight," "Normal Weight," "Overweight," or "Obese."

BMI < 18.5, category = "Underweight"BMI >= 18.5 & < 24.9, category =

"Normal Weight"BMI >= 25 & < 29.9, category = "Overweight"else category = "Obese"

Implement user input for weight and height and display both the standard and custom BMI calculations.

Input Format

The first line of input consists of a double value, representing the weight in kgs.

The second line consists of a double value, representing the height in meters.

Output Format

The first line of output prints: "Standard BMI Calculation:"

The second line of output prints: "BMI: " followed by the calculated BMI value (to two decimal places).

The third line of output prints: "Custom BMI Calculation:"

The fourth line of output prints: "Category: " followed by the BMI category.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 69.7
2.6
Output: Standard BMI Calculation:
BMI: 10.31
Custom BMI Calculation:
Category: Underweight

Answer

```
import java.util.Scanner;

// You are using Java
// You are using Java

/**
 * Base class to calculate Body Mass Index (BMI).
```

```
 */
class BMICalculator {

    /**
     * Computes BMI using the formula: weight / (height * height).
     *
     * @param weight The weight in kilograms (kg).
     * @param height The height in meters (m).
     * @return The calculated BMI as a double.
     */
    public double calculateBMI(double weight, double height) {
        return weight / (height * height);
    }
}

/**
 * Subclass that extends BMICalculator to add custom category logic.
 * It overrides the calculateBMI method to print the BMI category.
 */
class CustomBMICalculator extends BMICalculator {

    /**
     * Overrides the base method to calculate BMI and then print the
     * corresponding category based on the calculated value.
     *
     * @param weight The weight in kilograms (kg).
     * @param height The height in meters (m).
     * @return The calculated BMI as a double (same as the parent class method).
     */
    @Override
    public double calculateBMI(double weight, double height) {
        // Call the superclass method to get the standard BMI value
        double bmi = super.calculateBMI(weight, height);

        // Determine the category based on the BMI value and print it
        if (bmi < 18.5) {
            System.out.println("Category: Underweight");
        } else if (bmi <= 24.9) {
            System.out.println("Category: Normal Weight");
        } else if (bmi <= 29.9) {
            System.out.println("Category: Overweight");
        } else {
```

```
            System.out.println("Category: Obese");
        }

        // Return the BMI value as required by the overridden method's signature
        return bmi;
    }
}

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

        double weight = scanner.nextDouble();
        double height = scanner.nextDouble();

        BMIcalculator bmiCalculator = new BMIcalculator(weight, height);
        System.out.println("Standard BMI Calculation:");
        bmiCalculator.displayBMI();

        CustomBMIcalculator customBMIcalculator = new
CustomBMIcalculator(weight, height);
        System.out.println("Custom BMI Calculation:");
        customBMIcalculator.displayCustomBMI();

        scanner.close();
    }
}
```

Status : Wrong ***Marks : 0/10***