Rajalakshmi Engineering College

Name: varsha s
Email: 241501237@rajalakshmi.edu.in
Roll no:
Phone: 9342191041
Branch: REC
Department: AI & ML - Section 1
Batch: 2028
Degree: B.E - AI & ML

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 : 40

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

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

// You are using Java

class Employee {
    String name;
    int employeeID;

    // Constructor for the base class
    Employee(String name, int employeeID) {
        this.name = name;
        this.employeeID = employeeID;
    }
}
```

```
class Manager extends Employee {
    double salary; // Monthly salary

    // Constructor for the derived class
    Manager(String name, int employeeID, double salary) {
        super(name, employeeID); // Call to base class constructor
        this.salary = salary;
    }

    // Method to calculate annual salary (12 months)
    double calculateAnnualSalary() {
        return salary * 12;
    }
}

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

        String name = scanner.nextLine();
        int employeeID = scanner.nextInt();
        double salary = scanner.nextDouble();

        Manager manager = new Manager(name, employeeID, salary);

        System.out.println("Name: " + manager.name);
        System.out.println("Annual Salary: Rs. " +
df.format(manager.calculateAnnualSalary()));

        scanner.close();
    }
}
```

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

2. 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

```
import java.util.Scanner;



class Vehicle {
   double speed; // Speed in km/h
   double fuelCapacity; // Fuel capacity in liters

   // Constructor for the base class
   Vehicle(double speed, double fuelCapacity) {
      this.speed = speed;
      this.fuelCapacity = fuelCapacity;
   }
}

class Car extends Vehicle {
   // Constructor for the derived class
   Car(double speed, double fuelCapacity) {
      super(speed, fuelCapacity);
   }

   // Method to calculate travel distance
   double calculateTravelDistance() {
      // Formula: Travel Distance = Speed * Fuel Capacity [cite: 283]
      return speed * fuelCapacity;
   }
}
public class Main {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);

      double speed = scanner.nextDouble();
      double fuelCapacity = scanner.nextDouble();

      Car car = new Car(speed, fuelCapacity);

      System.out.println("Speed: " + String.format("%.2f", car.speed) + " km/h");
      System.out.println("Fuel Capacity: " + String.format("%.2f", car.fuelCapacity)
+ " liters");
```

```
        System.out.println("Travel Distance: " + String.format("%.2f",
car.calculateTravelDistance()) + " km");

        scanner.close();
    }
}
```

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

3. 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;

class VelocityCalculator {
    // Overloaded method 1: Accepts integer distance and time [cite: 270]
    static int calculateVelocity(int distance, int time) {
        return distance / time; // Velocity = distance / time [cite: 271]
    }

    // Overloaded method 2: Accepts double distance and time [cite: 270]
    static double calculateVelocity(double distance, double time) {
        return distance / time; // Velocity = distance / time [cite: 271]
    }
}

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 : Correct **Marks : 10/10**

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;


class BMIcalculator {
    double weight; // in kgs
    double height; // in meters

    BMIcalculator(double weight, double height) {
        this.weight = weight;
        this.height = height;
    }
```

```
    // Standard BMI calculation method
    double calculateBMI() {
        return weight / (height * height); // Formula: weight / (height * height) [cite:
125]
    }

    void displayBMI() {
        System.out.printf("BMI: %.2f\n", calculateBMI()); // Prints BMI to two
decimal places [cite: 137]
    }
}

class CustomBMIcalculator extends BMIcalculator {
    CustomBMIcalculator(double weight, double height) {
        super(weight, height);
    }

    // The problem statement implies the custom calculation is the category
assignment,
    // not a change in the BMI formula itself. We override the method but call
super()
    // to maintain the standard calculation for the custom category logic.
    @Override
    double calculateBMI() {
        return super.calculateBMI();
    }

    String getCategory() {
        double bmi = calculateBMI(); // Get the standard BMI

        if (bmi < 18.5) {
            return "Underweight"; // BMI < 18.5 [cite: 127]
        } else if (bmi >= 18.5 && bmi < 24.9) {
            return "Normal Weight"; // BMI >= 18.5 & < 24.9 [cite: 128]
        } else if (bmi >= 25 && bmi < 29.9) {
            return "Overweight"; // BMI >= 25 & < 29.9 [cite: 129]
        } else {
            return "Obese"; // Else (>= 29.9) [cite: 130]
        }
    }
```

```
    void displayCustomBMI() {
        System.out.println("Category: " + getCategory()); // Prints the category [cite:
139]
    }
}


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