OOPJ ASS3 (SN)

Sunday, September 8, 2024 8:58 PM

Note:

- The assignment is designed to practice class, fields, and methods only.
- Create a separate project for each question.
- Do not use getter/setter methods or constructors for these assignments.
- Define two classes: one class to implement the logic and another class to test it.

1. Loan Amortization Calculator

Implement a system to calculate and display the monthly payments for a mortgage loan. The system should:

1. Accept the principal amount (loan amount), annual interest rate, and loan term (in years) from the user.
2. Calculate the monthly payment using the standard mortgage formula:
   - **Monthly Payment Calculation:**
     - `monthlyPayment = principal * (monthlyInterestRate * (1 + monthlyInterestRate)^(numberOfMonths)) / ((1 + monthlyInterestRate)^(numberOfMonths) - 1)`
     - Where `monthlyInterestRate = annualInterestRate / 12 / 100` and `numberOfMonths = loanTerm * 12`
     - Note: Here ^ means power and to find it you can use Math.pow( ) method
3. Display the monthly payment and the total amount paid over the life of the loan, in Indian Rupees (₹).

Define class LoanAmortizationCalculator with methods acceptRecord, calculateMonthlyPayment & printRecord and test the functionality in main method.

```java
package org.example2;
import java.util.Scanner;

public class Loan {
	private double principle;
	private double interest;
	private int time;

	public void acceptRecord () {
		Scanner sc = new Scanner (System.in);

		System.out.print ("Enter principle : ");
		this.principle = sc.nextDouble();
		System.out.print ("Enter interest : ");
		this.interest = sc.nextDouble();
		System.out.print ("Enter time : ");
		this.time = sc.nextInt();
	}


	public double monthlyPayment () {
		double monthlyInterestRate = interest / 12 / 100;
		int numberOfMonths = time * 12;

	  double numerator = monthlyInterestRate * Math.pow(1 + monthlyInterestRate, numberOfMonths);
	  double denominator = Math.pow(1 + monthlyInterestRate, numberOfMonths) - 1;
	  return principle * (numerator / denominator);
	}

	public void printRecord() {
	  double monthlyPayment = monthlyPayment();
	  double totalAmountPaid = monthlyPayment * time * 12;
	  System.out.print ("Monthly Payment is : " + monthlyPayment+ " / "+"Total amount paid is : " + totalAmountPaid );
	}


     public static void main(String[] args) {
		Loan ln = new Loan ();

		ln.acceptRecord();
		ln.printRecord();

	}
}
```

```
Problems  Javadoc  Declaration  Console ×
<terminated> Loan [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v20240426-1
Enter principle : 100000
Enter interest :        12
Enter time :    10
Monthly Payment is : 1434.7094840258733 / Total amount paid is : 172165.1380831048
```

2. Compound Interest Calculator for Investment

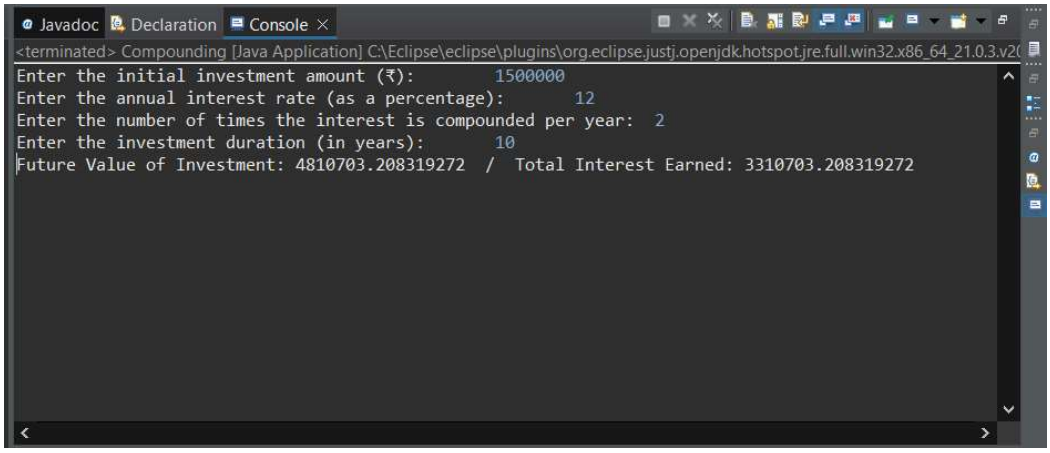
Develop a system to compute the future value of an investment with compound interest. The system should:

1. Accept the initial investment amount, annual interest rate, number of times the interest is compounded per year, and investment duration (in years) from the user.
2. Calculate the future value of the investment using the formula:
   - **Future Value Calculation:**
     - `futureValue = principal * (1 + annualInterestRate / numberOfCompounds)^(numberOfCompounds * years)`
   - **Total Interest Earned:** `totalInterest = futureValue - principal`
3. Display the future value and the total interest earned, in Indian Rupees (₹).

Define class CompoundInterestCalculator with methods acceptRecord , calculateFutureValue, printRecord and test the functionality in main method.

```java
package org.example2;

import java.util.Scanner;

public class Compounding {
    private double investment;
    private double interest;
    private int n;
    private int time;


    public void acceptRecord() {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the initial investment amount (₹): ");
        this.investment = sc.nextDouble();
        System.out.print("Enter the annual interest rate (as a percentage): ");
        this.interest = sc.nextDouble();
        System.out.print("Enter the number of times the interest is compounded per year: ");
        this.n = sc.nextInt();
        System.out.print("Enter the investment duration (in years): ");
        this.time = sc.nextInt();
    }


    public double calculateFutureValue() {
        double ratePerPeriod = interest / 100 / n;
        int totalPeriods = n * time;
        return investment * Math.pow(1 + ratePerPeriod, totalPeriods);
    }


    public double calculateTotalInterest() {
        return calculateFutureValue() - investment;
    }


    public void printRecord() {
        double futureValue = calculateFutureValue();
        double totalInterest = calculateTotalInterest();

        System.out.print("Future Value of Investment: "+ futureValue +"  /  "+"Total Interest Earned: "+ totalInterest);
    }

    public static void main(String[] args) {
        // Create an instance of the Compounding class
        Compounding cp = new Compounding();

        // Accept user input
        cp.acceptRecord();
```

```
        // Print the results
        cp.printRecord();
    }
}
```



3. BMI (Body Mass Index) Tracker

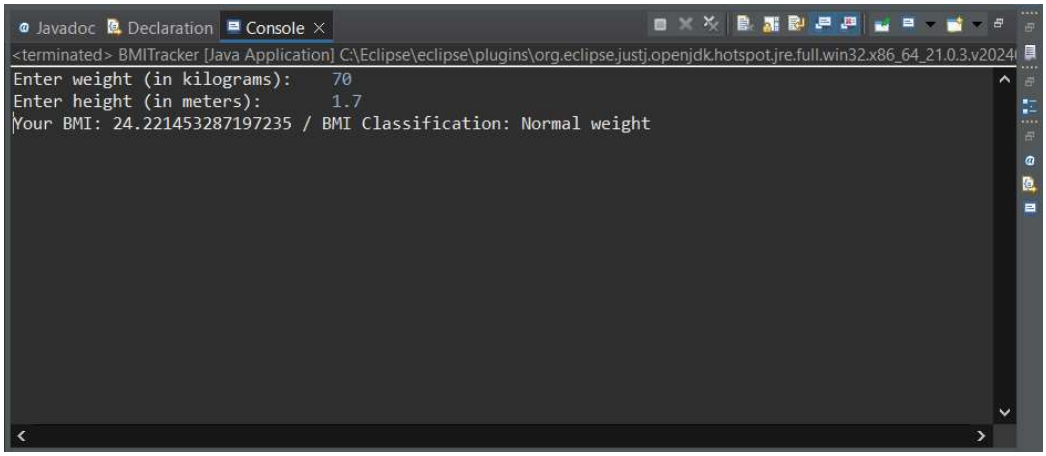
Create a system to calculate and classify Body Mass Index (BMI). The system should:

1. Accept weight (in kilograms) and height (in meters) from the user.
2. Calculate the BMI using the formula:
   - **BMI Calculation:** `BMI = weight / (height * height)`
3. Classify the BMI into one of the following categories:
   - Underweight: BMI < 18.5
   - Normal weight: 18.5 ≤ BMI < 24.9
   - Overweight: 25 ≤ BMI < 29.9
   - Obese: BMI ≥ 30
4. Display the BMI value and its classification.

Define class BMITracker with methods acceptRecord, calculateBMI, classifyBMI & printRecord and test the functionality in main method.

```
package org.example2;

import java.util.Scanner;

public class BMITracker {
    private double weight;
    private double height;


    public void acceptRecord() {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter weight (in kilograms): ");
        this.weight = sc.nextDouble();
        System.out.print("Enter height (in meters): ");
        this.height = sc.nextDouble();
    }


    public double calculateBMI() {
        return weight / (height * height);
    }


    public String classifyBMI(double bmi) {
        if (bmi < 18.5) {
            return "Underweight";
        } else if (bmi < 24.9) {
            return "Normal weight";
        } else if (bmi < 29.9) {
            return "Overweight";
        } else {
            return "Obese";
        }
    }


    public void printRecord() {
        double bmi = calculateBMI();
        String classification = classifyBMI(bmi);

        System.out.print("Your BMI: "+bmi+ " / "+"BMI Classification: " + classification);
```

```
    }

    public static void main(String[] args) {
        BMITracker bmiTracker = new BMITracker();


        bmiTracker.acceptRecord();


        bmiTracker.printRecord();
    }
}
```



4. Discount Calculation for Retail Sales

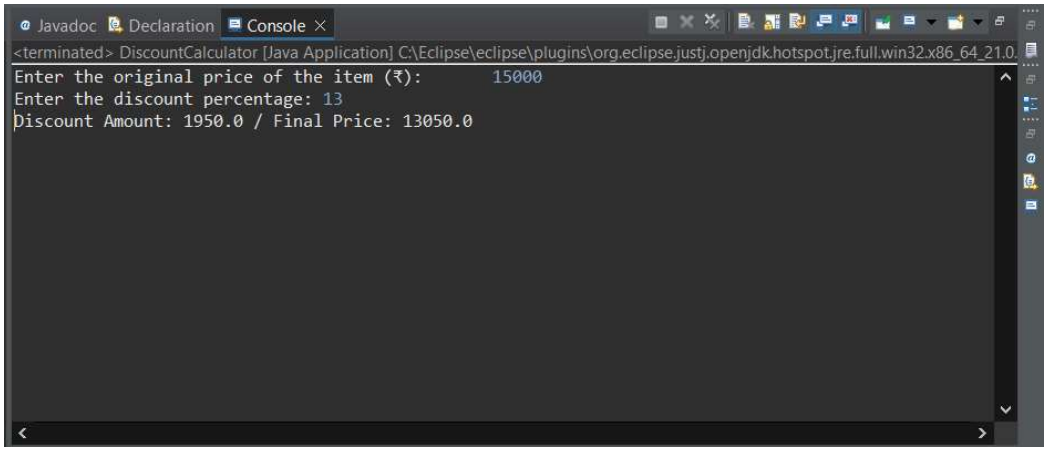
Design a system to calculate the final price of an item after applying a discount. The system should:

1. Accept the original price of an item and the discount percentage from the user.
2. Calculate the discount amount and the final price using the following formulas:
   - **Discount Amount Calculation:** `discountAmount = originalPrice * (discountRate / 100)`
   - **Final Price Calculation:** `finalPrice = originalPrice - discountAmount`
3. Display the discount amount and the final price of the item, in Indian Rupees (₹).

Define class DiscountCalculator with methods acceptRecord, calculateDiscount & printRecord and test the functionality in main method.

```
package org.example2;

import java.util.Scanner;

public class DiscountCalculator {
    private double originalPrice;
    private double discountRate;


    public void acceptRecord() {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the original price of the item (₹): ");
        this.originalPrice = sc.nextDouble();
        System.out.print("Enter the discount percentage: ");
        this.discountRate = sc.nextDouble();
    }


    public double calculateDiscountAmount() {
        return originalPrice * (discountRate / 100);
    }


    public double calculateFinalPrice() {
        double discountAmount = calculateDiscountAmount();
        return originalPrice - discountAmount;
    }


    public void printRecord() {
        double discountAmount = calculateDiscountAmount();
        double finalPrice = calculateFinalPrice();

        System.out.print("Discount Amount: "+ discountAmount +" / "+"Final Price: "+ finalPrice);

    }

    public static void main(String[] args) {
        DiscountCalculator dc = new DiscountCalculator();
```

```
        dc.acceptRecord();


        dc.printRecord();
    }
}
```



5. Toll Booth Revenue Management

Develop a system to simulate a toll booth for collecting revenue. The system should:

1. Allow the user to set toll rates for different vehicle types: Car, Truck, and Motorcycle.
2. Accept the number of vehicles of each type passing through the toll booth.
3. Calculate the total revenue based on the toll rates and number of vehicles.
4. Display the total number of vehicles and the total revenue collected, in Indian Rupees (₹).

- **Toll Rate Examples:**
    - Car: ₹50.00
    - Truck: ₹100.00
    - Motorcycle: ₹30.00

Define class TollBoothRevenueManager with methods acceptRecord, setTollRates, calculateRevenue & printRecord and test the functionality in main method.

```
package org.example2;

import java.util.Scanner;

public class TollBoothRevenueManager {
    private double carRate;
    private double truckRate;
    private double motorcycleRate;
    private int numCars;
    private int numTrucks;
    private int numMotorcycles;


    public void setTollRates() {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the toll rate for Car (₹): ");
        this.carRate = sc.nextDouble();
        System.out.print("Enter the toll rate for Truck (₹): ");
        this.truckRate = sc.nextDouble();
        System.out.print("Enter the toll rate for Motorcycle (₹): ");
        this.motorcycleRate = sc.nextDouble();
    }


    public void acceptRecord() {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the number of Cars: ");
        this.numCars = sc.nextInt();
        System.out.print("Enter the number of Trucks: ");
        this.numTrucks = sc.nextInt();
        System.out.print("Enter the number of Motorcycles: ");
        this.numMotorcycles = sc.nextInt();
    }


    public double calculateRevenue() {
        double totalRevenue = (numCars * carRate) + (numTrucks * truckRate) + (numMotorcycles * motorcycleRate);
        return totalRevenue;
    }
```

```java
    public void printRecord() {
        double totalRevenue = calculateRevenue();
        int totalVehicles = numCars + numTrucks + numMotorcycles;

        System.out.printf("Total Number of Vehicles: %d\n", totalVehicles);
        System.out.printf("Total Revenue Collected: ₹%.2f\n", totalRevenue);
    }

    public static void main(String[] args) {
        TollBoothRevenueManager manager = new TollBoothRevenueManager();


        manager.setTollRates();


        manager.acceptRecord();


        manager.printRecord();
    }
}
```