OOPJ
Assignment-3

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.

```
package com.practice.q1;
import java.util.Scanner;

class LoanAmortizationCalculator {

    private double principal;
    private double annualInterestRate;
    private double loanTerm;
    private double monthlyInterestRate;
    private double numberOfMonths;
    private double monthlyPayment;
    private double totalAmount;
```

```
	public void acceptRecord() {
		Scanner sc = new Scanner(System.in);
		System.out.print("Enter Principal: ");
		this.principal = sc.nextDouble();
		System.out.print("Enter Interest Rate: ");
		this.annualInterestRate = sc.nextDouble();
		System.out.print("Enter Loan Term: ");
		this.loanTerm = sc.nextDouble();
	}

	public double calculateMonthlyPayment() {
		monthlyInterestRate = annualInterestRate / 12 / 100;
		numberOfMonths = loanTerm * 12;
		monthlyPayment = principal * (monthlyInterestRate * Math.pow((1 + monthlyInterestRate),
(numberOfMonths))) / (Math.pow((1 + monthlyInterestRate), (numberOfMonths)) - 1);
		return monthlyPayment;
	}

	public double calculateTotalAmount() {
		totalAmount = calculateMonthlyPayment() * loanTerm * 12;
		return totalAmount;
	}

	public void printRecord() {
		System.out.printf("Monthly Payment: %.2f%n", monthlyPayment);
		System.out.printf("Total Amount Paid: %.2f%n", totalAmount);
	}

}
public class Program {
	public static void main(String[] args) {

		LoanAmortizationCalculator lac = new LoanAmortizationCalculator();

		lac.acceptRecord();
		lac.calculateMonthlyPayment();
		lac.calculateTotalAmount();
		lac.printRecord();
	}
}
```

Output

```
Enter Principal: 1000000
Enter Interest Rate: 10
Enter Loan Term: 10
Monthly Payment: 13215.07
Total Amount Paid: 1585808.84
```

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.

```
package com.practice.q2;
import java.util.Scanner;

class CompoundInterestCalculator {
	private double principal;
	private double annualInterestRate;
	private double numberOfCompounds;
	private double years;
	private double futureValue;
	private double totalInterest;

	public void acceptRecord() {
		Scanner sc = new Scanner(System.in);
		System.out.print("Enter Principal: ");
		this.principal = sc.nextDouble();
		System.out.print("Enter Interest Rate: ");
		this.annualInterestRate = sc.nextDouble();
		System.out.print("Enter number of compounds: ");
		this.numberOfCompounds = sc.nextDouble();
		System.out.print("Enter duration: ");
		this.years = sc.nextDouble();
	}

	public double calculateFutureValue() {
		futureValue = principal * Math.pow((1 + annualInterestRate / numberOfCompounds / 100),
(numberOfCompounds * years));
		return futureValue;
	}

	public double calculateTotalInterest() {
		totalInterest = futureValue - principal;
		return totalInterest;
	}

	public void printRecord() {
		System.out.printf("Future Value: %.2f%n", futureValue);
		System.out.printf("Total Interest: %.2f%n", totalInterest);
	}
}
public class Program {
```

```
	public static void main(String[] args) {
		CompoundInterestCalculator cic = new CompoundInterestCalculator();

		cic.acceptRecord();
		cic.calculateFutureValue();
		cic.calculateTotalInterest();
		cic.printRecord();
	}
}
```

Output

```
Enter Principal: 100000
Enter Interest Rate: 8
Enter number of compounds: 2
Enter duration: 3
Future Value: 126531.90
Total Interest: 26531.90
```

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 com.example.a3q3;
import java.util.Scanner;
class BMITracker {

	private double weight;
	private double height;
	private double bmi;

	public void acceptRecord() {
		Scanner sc = new Scanner(System.in);
		System.out.print("Enter weight (in kg): ");
```

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

        public double calculateBMI() {
                bmi = weight / Math.pow(height, 2);
                return bmi;
        }

        public String classifyBMI() {
                if(bmi > 30) {
                        return "Obese";
                } else if (bmi > 25) {
                        return "Overweight";
                } else if (bmi > 18.5) {
                        return "Normal weight";
                } else {
                        return "Under weight";
                }
        }

        public void printRecord() {
                System.out.printf("BMI: %.2f%n", this.bmi );
                System.out.printf("Classification: %s", this.classifyBMI());
        }
}
public class Program {
        public static void main(String[] args) {

                BMITracker bmit = new BMITracker();

                bmit.acceptRecord();
                bmit.calculateBMI();
                bmit.printRecord();
        }
}
```

Output

```
Enter weight (in kg): 73
Enter height (in m): 1.75
BMI: 23.84
Classification: Normal weight
```

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 com.example.a3q4;

import java.util.Scanner;

class DiscountCalculator{

	private float originalPrice;
	private float discountRate;
	private float discountAmount;
	private float finalPrice;

	public void acceptRecord() {
		Scanner sc = new Scanner(System.in);
		System.out.print("Enter Original Price: ");
		this.originalPrice = sc.nextFloat();
		System.out.print("Enter dicount percent: ");
		this.discountRate = sc.nextFloat();
	}

	public float calculateDiscount() {
		discountAmount = originalPrice * (discountRate / 100);
		return discountAmount;
	}

	public float calculateFinalPrice() {
		finalPrice = originalPrice - discountAmount;
		return finalPrice;
	}

	public void printRecord() {
		System.out.printf("Discount Amount: %.2f%n", calculateDiscount());
		System.out.printf("Final Amount: %.2f%n", calculateFinalPrice());
	}

}
public class Program {
	public static void main(String[] args) {

		DiscountCalculator dc = new DiscountCalculator();

		dc.acceptRecord();
		dc.printRecord();

	}
}
```

Output

```
Enter Original Price: 1000
Enter dicount percent: 15
Discount Amount: 150.00
Final Amount: 850.00
```

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 com.example.a3q5;

import java.util.Scanner;

class TollBoothRevenueManager {

	private float carTollRate;
	private float truckTollRate;
	private float motorcycleTollRate;
	private int numberOfCar;
	private int numberOfTruck;
	private int numberOfMotorcycle;
	private float totalRevenue;
	private int totalVehicle;

	public void acceptRecord() {

		Scanner sc = new Scanner(System.in);
		System.out.print("Enter number of Cars: ");
		this.numberOfCar = sc.nextInt();
		System.out.print("Enter number of Trucks: ");
		this.numberOfTruck = sc.nextInt();
		System.out.print("Enter number of Motorcycles: ");
		this.numberOfMotorcycle = sc.nextInt();
	}
```

```
	public void setTollRates() {

		Scanner sc = new Scanner(System.in);
		System.out.print("Enter toll rate for Cars: ");
		this.carTollRate = sc.nextFloat();
		System.out.print("Enter toll rate for Trucks: ");
		this.truckTollRate = sc.nextFloat();
		System.out.print("Enter toll rate for Motorcycles: ");
		this.motorcycleTollRate = sc.nextFloat();
	}

	public float calculateRevenue() {
		totalRevenue = (numberOfCar * carTollRate) + (numberOfTruck * truckTollRate) +
(numberOfMotorcycle * motorcycleTollRate);
		return totalRevenue;
	}

	public int calculateTotalVehicle() {
		totalVehicle = numberOfCar + numberOfTruck + numberOfMotorcycle;
		return totalVehicle;
	}

	public void printRecord() {
		System.out.println("Total Vehicle: " + calculateTotalVehicle());
		System.out.printf("Total Revenue: %.2f%n", calculateRevenue());
	}
}
public class Program {
	public static void main(String[] args) {

		TollBoothRevenueManager tbrm = new TollBoothRevenueManager();

		tbrm.acceptRecord();
		tbrm.setTollRates();
		tbrm.calculateRevenue();
		tbrm.printRecord();
	}
}
```

Output

```
Enter number of Cars: 300
Enter number of Trucks: 800
Enter number of Motorcycles: 500
Enter toll rate for Cars: 50
Enter toll rate for Trucks: 100
Enter toll rate for Motorcycles: 30
Total Vehicle: 1600
Total Revenue: 110000.00
```