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:

- Accept the principal amount (loan amount), annual interest rate, and loan term (in years) from the user.
- 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
- 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 org.example.demo1;

import java.util.Scanner;

public class LoanAmortizationCalculator {

    private double principal;
    private double annualInterestRate;
    private int loanTerm;

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

        System.out.print("Enter the loan amount (principal) in ₹: ");
        principal = scanner.nextDouble();

        System.out.print("Enter the annual interest rate (in %): ");
        annualInterestRate = scanner.nextDouble();

        System.out.print("Enter the loan term (in years): ");
        loanTerm = scanner.nextInt();
```

```
    }

    public double calculateMonthlyPayment() {

        double monthlyInterestRate = annualInterestRate /
12 / 100;

        int numberOfMonths = loanTerm * 12;

        double monthlyPayment = principal *
(monthlyInterestRate * Math.pow(1 + monthlyInterestRate,
numberOfMonths))
                / (Math.pow(1 + monthlyInterestRate,
numberOfMonths) - 1);

        return monthlyPayment;
    }

    public void printRecord(double monthlyPayment) {

        double totalAmountPaid = monthlyPayment *
loanTerm * 12;

        System.out.printf("Your monthly payment is: ₹%.2f
\n", monthlyPayment);
```

```
        System.out.printf("Total amount paid over the loan
term: ₹%.2f\n", totalAmountPaid);
    }

    public static void main(String[] args) {
        LoanAmortizationCalculator calculator = new
LoanAmortizationCalculator();

        calculator.acceptRecord();

        double monthlyPayment =
calculator.calculateMonthlyPayment();

        calculator.printRecord(monthlyPayment);
    }
}
```

2. Compound Interest Calculator for Investment

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

- 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.
- 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
- 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 org.example.demo1;


import java.util.Scanner;


public class CompoundInterestCalculator {



   private double principal;

   private double annualInterestRate;

   private int numberOfCompounds;

   private int years;



   public void acceptRecord() {
```

```
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the initial investment amount (principal)
in ₹: ");
        principal = scanner.nextDouble();


        System.out.print("Enter the annual interest rate (in %): ");
        annualInterestRate = scanner.nextDouble();


        System.out.print("Enter the number of times the interest is
compounded per year: ");
        numberOfCompounds = scanner.nextInt();


        System.out.print("Enter the investment duration (in years): ");
        years = scanner.nextInt();
    }


    public double calculateFutureValue() {


        double interestRateDecimal = annualInterestRate / 100;
```

```
    double futureValue = principal * Math.pow(1 +
(interestRateDecimal / numberOfCompounds),
                                numberOfCompounds * years);

    return futureValue;
  }

  public void printRecord(double futureValue) {

    double totalInterest = futureValue - principal;

    System.out.printf("Future value of the investment: ₹%.2f\n",
futureValue);
    System.out.printf("Total interest earned: ₹%.2f\n", totalInterest);
  }

  public static void main(String[] args) {
    CompoundInterestCalculator calculator = new
CompoundInterestCalculator();  // Create an object of the class
```

```
        calculator.acceptRecord();



        double futureValue = calculator.calculateFutureValue();



        calculator.printRecord(futureValue);
    }
}
```

3. BMI (Body Mass Index) Tracker

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

- Accept weight (in kilograms) and height (in meters) from the user.
- Calculate the BMI using the formula:
    - **BMI Calculation:** BMI = weight / (height * height)
- 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

- 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.example.demo1;

import java.util.Scanner;

public class BMITracker {

	private double weight;
	private double height;

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

		System.out.print("Enter your weight (in kg): ");
		weight = scanner.nextDouble();

		System.out.print("Enter your height (in meters): ");
		height = scanner.nextDouble();
	}

	public double calculateBMI() {

		double bmi = weight / (height * height);
		return bmi;
	}

	public String classifyBMI(double bmi) {
		String classification;
```

```
            if (bmi < 18.5) {
                classification = "Underweight";
            } else if (bmi >= 18.5 && bmi < 24.9) {
                classification = "Normal weight";
            } else if (bmi >= 25 && bmi < 29.9) {
                classification = "Overweight";
            } else {
                classification = "Obese";
            }

            return classification;
        }

        public void printRecord(double bmi, String classification) {

            System.out.println("BMI Classification: " + classification);
        }

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

            tracker.acceptRecord();

            double bmi = tracker.calculateBMI();

            String classification = tracker.classifyBMI(bmi);

            tracker.printRecord(bmi, classification);
        }
}
```

4. Discount Calculation for Retail Sales

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

- Accept the original price of an item and the discount percentage

from the user.

- 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
- 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.example.demo1;
import java.util.Scanner;

public class DiscountCalculator {


  private double originalPrice;
  private double discountRate;


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

    System.out.print("Enter the original price of the item (in ₹): ");
    originalPrice = scanner.nextDouble();


    System.out.print("Enter the discount percentage: ");
    discountRate = scanner.nextDouble();
  }
```

```
    public double[] calculateDiscount() {

        double discountAmount = originalPrice * (discountRate / 100);


        double finalPrice = originalPrice - discountAmount;

        return new double[]{discountAmount, finalPrice};  // Return both
values as an array
    }


    public void printRecord(double discountAmount, double finalPrice) {

        System.out.printf("Discount Amount: ₹%.2f\n", discountAmount);
        System.out.printf("Final Price after discount: ₹%.2f\n", finalPrice);
    }


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


        calculator.acceptRecord();


        double[] results = calculator.calculateDiscount();


        calculator.printRecord(results[0], results[1]);
    }
}
```

5. Toll Booth Revenue Management

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

- Allow the user to set toll rates for different vehicle types: Car, Truck, and Motorcycle.
- Accept the number of vehicles of each type passing through the toll booth.
- Calculate the total revenue based on the toll rates and number of vehicles.
- 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.example.demo3;
import java.util.Scanner;

public class TollBoothRevenueManager {

   private double carTollRate;
   private double truckTollRate;
   private double motorcycleTollRate;

   private int numCars;
   private int numTrucks;
```

```
private int numMotorcycles;


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


   System.out.print("Enter the toll rate for Car in ₹: ");
   carTollRate = scanner.nextDouble();

   System.out.print("Enter the toll rate for Truck in ₹: ");
   truckTollRate = scanner.nextDouble();

   System.out.print("Enter the toll rate for Motorcycle in ₹ : ");
   motorcycleTollRate = scanner.nextDouble();
}


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


   System.out.print("Enter the number of Cars: ");
   numCars = scanner.nextInt();


   System.out.print("Enter the number of Trucks: ");
   numTrucks = scanner.nextInt();


   System.out.print("Enter the number of Motorcycles: ");
   numMotorcycles = scanner.nextInt();
}


public double calculateRevenue() {

   double carRevenue = numCars * carTollRate;
```

```
        double truckRevenue = numTrucks * truckTollRate;
        double motorcycleRevenue = numMotorcycles *
motorcycleTollRate;


        double totalRevenue = carRevenue + truckRevenue +
motorcycleRevenue;

        return totalRevenue;
    }


    public void printRecord(double totalRevenue) {

        int totalVehicles = numCars + numTrucks + numMotorcycles;


        System.out.println("Total number of vehicles: " + totalVehicles);
        System.out.printf("Total revenue collected: ₹%.2f\n",
totalRevenue);
    }


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


        manager.setTollRates();


        manager.acceptRecord();


        double totalRevenue = manager.calculateRevenue();
```