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.

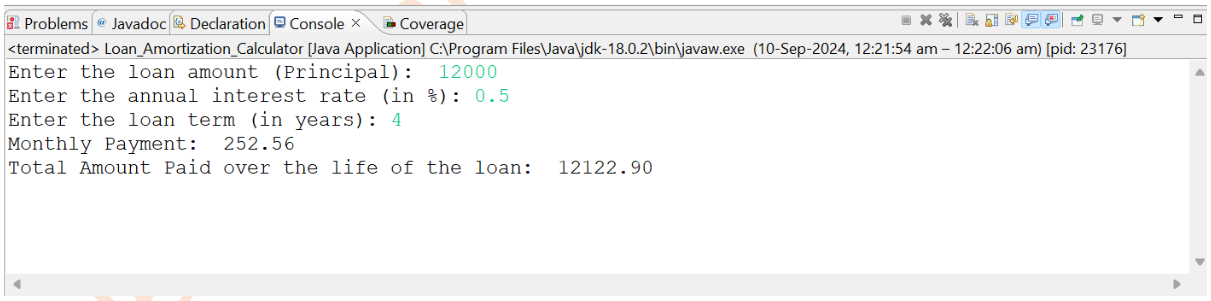
1. Loan Amortization Calculator

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 in.Cdac.LoanAmortizationCalculator;

import java.util.Scanner;

class Loan_Calculator{
      private double principal;
    private double annualInterestRate;
    private int loanTerm;


    public void acceptRecord() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the loan amount (Principal):  ");
        principal = sc.nextDouble();
        System.out.print("Enter the annual interest rate (in %): ");
        annualInterestRate = sc.nextDouble();
        System.out.print("Enter the loan term (in years): ");
        loanTerm = sc.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 = calculateMonthlyPayment();
        double totalPayment = monthlyPayment * loanTerm * 12;

        System.out.printf("Monthly Payment:  %.2f\n", monthlyPayment);
        System.out.printf("Total Amount Paid over the life of the loan: 
%.2f\n", totalPayment);
    }
}
public class Loan_Amortization_Calculator {
       public static void main(String[] args) {
              Loan_Calculator cal = new Loan_Calculator();

              cal.acceptRecord();
              cal.printRecord();
          }
}
```



```
<terminated> Loan_Amortization_Calculator [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe  (10-Sep-2024, 12:21:54 am – 12:22:06 am) [pid: 23176]
Enter the loan amount (Principal):  12000
Enter the annual interest rate (in %): 0.5
Enter the loan term (in years): 4
Monthly Payment:  252.56
Total Amount Paid over the life of the loan:  12122.90
```

2. Compound Interest Calculator for Investment

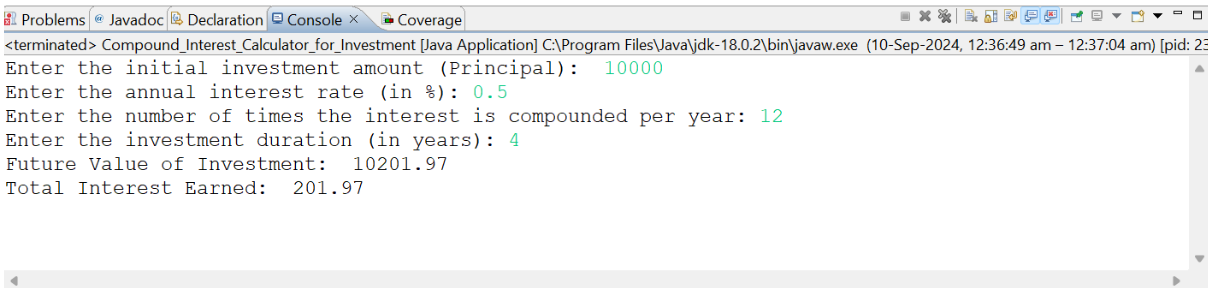
Develop a system to compute the future value of an investment with compound interest. The system should:

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 (₹).

ASSIGNMENT NO.3

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

```
package in.Cdac.CompoundInterestCalculatorforInvestment;

import java.util.Scanner;

class Compound_Interest_Calculator {
	private double principal;
	private double annualInterestRate;
	private int numberOfCompounds;
	private int investmentDuration;


	public void acceptRecord() {
		Scanner sc = new Scanner(System.in);
		System.out.print("Enter the initial investment amount 
(Principal):  ");
		principal = sc.nextDouble();
		System.out.print("Enter the annual interest rate (in %): ");
		annualInterestRate = sc.nextDouble();
		System.out.print("Enter the number of times the interest is 
compounded per year: ");
		numberOfCompounds = sc.nextInt();
		System.out.print("Enter the investment duration (in years): 
");
		investmentDuration = sc.nextInt();
	}


	public double calculateFutureValue() {
		double rate = annualInterestRate / 100;

		double futureValue = principal * Math.pow(1 + rate / 
numberOfCompounds, numberOfCompounds * investmentDuration);

		return futureValue;
	}
	public void printRecord() {
		double futureValue = calculateFutureValue();
		double totalInterest = futureValue - principal;
		System.out.printf("Future Value of Investment:  %.2f\n", 
futureValue);
		System.out.printf("Total Interest Earned:  %.2f\n", 
totalInterest);
	}
}

public class Compound_Interest_Calculator_for_Investment {
	public static void main(String[] args) {
	  Compound_Interest_Calculator cal = new 
Compound_Interest_Calculator();
	  cal.acceptRecord();
	  cal.printRecord();
	}
}
```



```
<terminated> Compound_Interest_Calculator_for_Investment [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe (10-Sep-2024, 12:36:49 am – 12:37:04 am) [pid: 23
Enter the initial investment amount (Principal):  10000
Enter the annual interest rate (in %): 0.5
Enter the number of times the interest is compounded per year: 12
Enter the investment duration (in years): 4
Future Value of Investment:  10201.97
Total Interest Earned:  201.97
```

3. BMI (Body Mass Index) Tracker

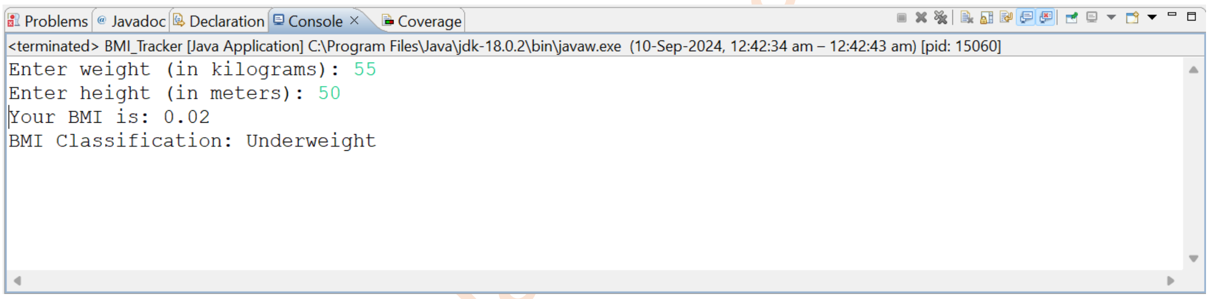
Create a system to calculate and classify Body Mass Index (BMI). The system should:

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 in.Cdac.BMITracker;

import java.util.Scanner;

class BMI {
	private double weight;
	private double height;

	public void acceptRecord() {
		Scanner sc = new Scanner(System.in);
		System.out.print("Enter weight (in kilograms): ");
		weight = sc.nextDouble();
		System.out.print("Enter height (in meters): ");
		height = sc.nextDouble();
	}

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

	public String classifyBMI(double bmi) {
		if (bmi < 18.5) {
			return "Underweight";
		} else if (bmi >= 18.5 && bmi < 24.9) {
			return "Normal weight";
		} else if (bmi >= 25 && bmi < 29.9) {
			return "Overweight";
```

```
                } else {
                        return "Obese";
                }
        }

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

                System.out.printf("Your BMI is: %.2f\n", bmi);
                System.out.println("BMI Classification: " + classification);
        }
}

public class BMI_Tracker {
        public static void main(String[] args) {
                BMI t1 = new BMI();
                t1.acceptRecord();
                t1.printRecord();
        }
}
```


Problems Javadoc Declaration Console Coverage

```
<terminated> BMI_Tracker [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe  (10-Sep-2024, 12:42:34 am – 12:42:43 am) [pid: 15060]
Enter weight (in kilograms): 55
Enter height (in meters): 50
Your BMI is: 0.02
BMI Classification: Underweight
```

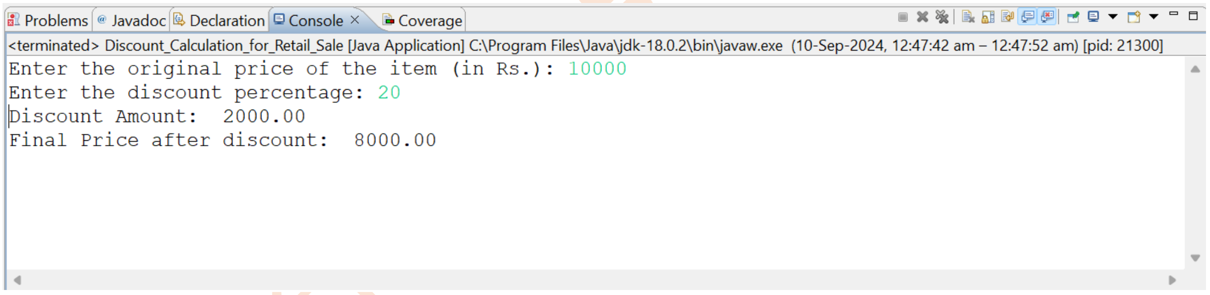
4. Discount Calculation for Retail Sales

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 in.Cdac.DiscountCalculationforRetailSales;

import java.util.Scanner;

class Discount_Calculation{
      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 (in 
Rs.): ");
        originalPrice = sc.nextDouble();
        System.out.print("Enter the discount percentage: ");
        discountRate = sc.nextDouble();
    }
    public double calculateDiscount() {
        return originalPrice * (discountRate / 100);
    }
    public void printRecord() {
        double discountAmount = calculateDiscount();
        double finalPrice = originalPrice - discountAmount;
        System.out.printf("Discount Amount:  %.2f\n", discountAmount);
        System.out.printf("Final Price after discount:  %.2f\n", 
finalPrice);
    }
}
public class Discount_Calculation_for_Retail_Sale {
       public static void main(String[] args) {
              Discount_Calculation cal = new Discount_Calculation();
              cal.acceptRecord();
              cal.printRecord();
          }
}
```



```
<terminated> Discount_Calculation_for_Retail_Sale [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe  (10-Sep-2024, 12:47:42 am – 12:47:52 am) [pid: 21300]
Enter the original price of the item (in Rs.): 10000
Enter the discount percentage: 20
Discount Amount:  2000.00
Final Price after discount:  8000.00
```

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 (₹).

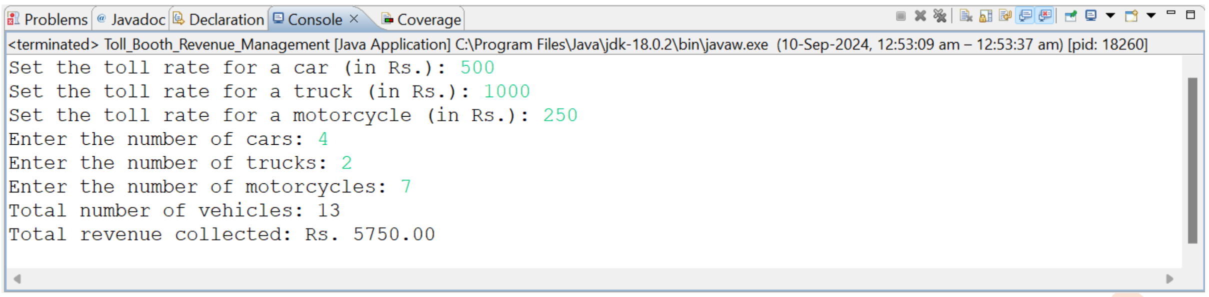
- **Toll Rate Examples:**
  - Car: ₹50.00
  - Truck: ₹100.00
  - Motorcycle: ₹30.00

ASSIGNMENT NO.3

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

```
package in.Cdac.TollBoothRevenueManagement;

import java.util.Scanner;

class Toll_Booth_Revenue{
      private double carRate;
    private double truckRate;
    private double motorcycleRate;
    private int numberOfCars;
    private int numberOfTrucks;
    private int numberOfMotorcycles;
    public void acceptRecord() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of cars: ");
        numberOfCars = sc.nextInt();
        System.out.print("Enter the number of trucks: ");
        numberOfTrucks = sc.nextInt();
        System.out.print("Enter the number of motorcycles: ");
        numberOfMotorcycles = sc.nextInt();
    }
    public void setTollRates() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Set the toll rate for a car (in Rs.): ");
        carRate = sc.nextDouble();
        System.out.print("Set the toll rate for a truck (in Rs.): ");
        truckRate = sc.nextDouble();
        System.out.print("Set the toll rate for a motorcycle (in Rs.):
");
        motorcycleRate = sc.nextDouble();
    }
    public double calculateRevenue() {
        double totalRevenue = (numberOfCars * carRate) +
(numberOfTrucks * truckRate) + (numberOfMotorcycles * motorcycleRate);
        return totalRevenue;
    }
    public void printRecord() {
        int totalVehicles = numberOfCars + numberOfTrucks +
numberOfMotorcycles;
        double totalRevenue = calculateRevenue();
        System.out.println("Total number of vehicles: " +
totalVehicles);
        System.out.printf("Total revenue collected: Rs. %.2f\n",
totalRevenue);
    }
}
public class Toll_Booth_Revenue_Management {
      public static void main(String[] args) {
        Toll_Booth_Revenue r1 = new Toll_Booth_Revenue();
        r1.setTollRates();
        r1.acceptRecord();
        r1.printRecord();
    }
}
```

ASSIGNMENT NO.3



```
<terminated> Toll_Booth_Revenue_Management [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe  (10-Sep-2024, 12:53:09 am – 12:53:37 am) [pid: 18260]
Set the toll rate for a car (in Rs.): 500
Set the toll rate for a truck (in Rs.): 1000
Set the toll rate for a motorcycle (in Rs.): 250
Enter the number of cars: 4
Enter the number of trucks: 2
Enter the number of motorcycles: 7
Total number of vehicles: 13
Total revenue collected: Rs. 5750.00
```