1. Loan Amortization Calculator

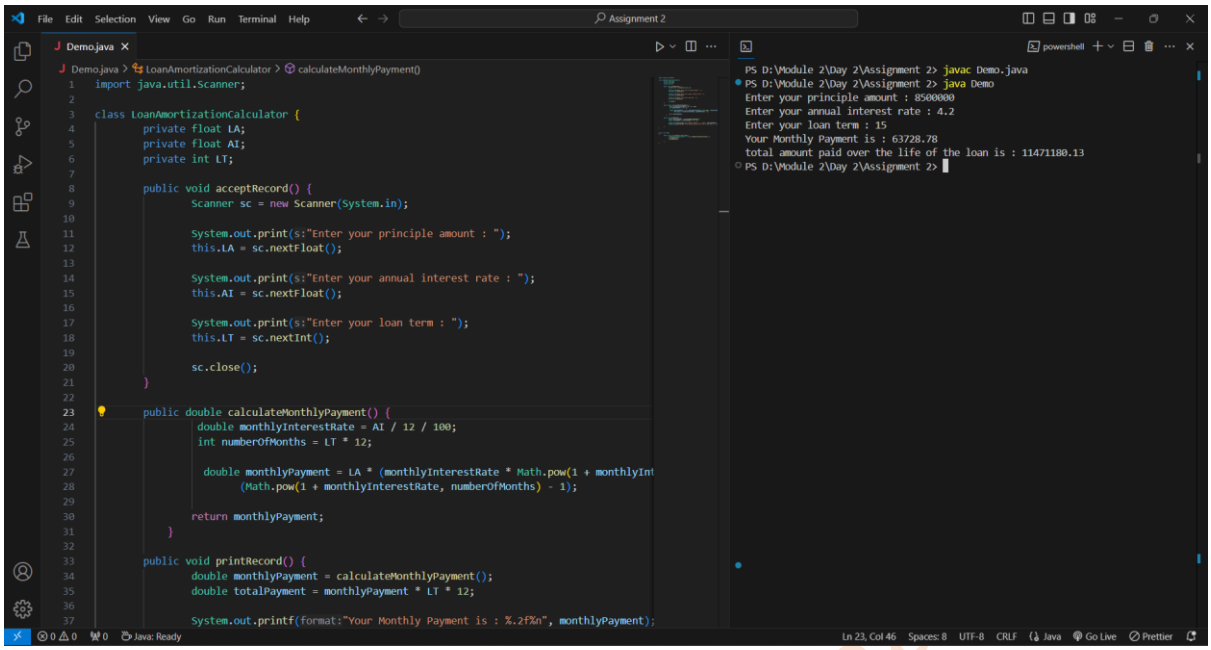
Implement a system to calculate and display the monthly payments for a mortgage loan. The system should:

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.

```
import java.util.Scanner;

class LoanAmortizationCalculator {
    private float LA;
    private float AI;
    private int LT;

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

        System.out.print("Enter your principle amount : ");
        this.LA = sc.nextFloat();

        System.out.print("Enter your annual interest rate : ");
        this.AI = sc.nextFloat();

        System.out.print("Enter your loan term : ");
        this.LT = sc.nextInt();

        sc.close();
    }

    public double calculateMonthlyPayment() {
```

```
            double monthlyInterestRate = AI / 12 / 100;
            int numberOfMonths = LT * 12;

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

            return monthlyPayment;
        }

    public void printRecord() {
            double monthlyPayment = calculateMonthlyPayment();
            double totalPayment = monthlyPayment * LT * 12;

            System.out.printf("Your Monthly Payment is : %.2f%n", monthlyPayment);
            System.out.printf("total amount paid over the life of the loan is : %.2f%n",
totalPayment);

    }
}


public class Demo
{
    public static void main(String[] args) {
            LoanAmortizationCalculator  l = new LoanAmortizationCalculator ();
            l.acceptRecord();
            l.printRecord();

    }
}
```



2. Compound Interest Calculator for Investment

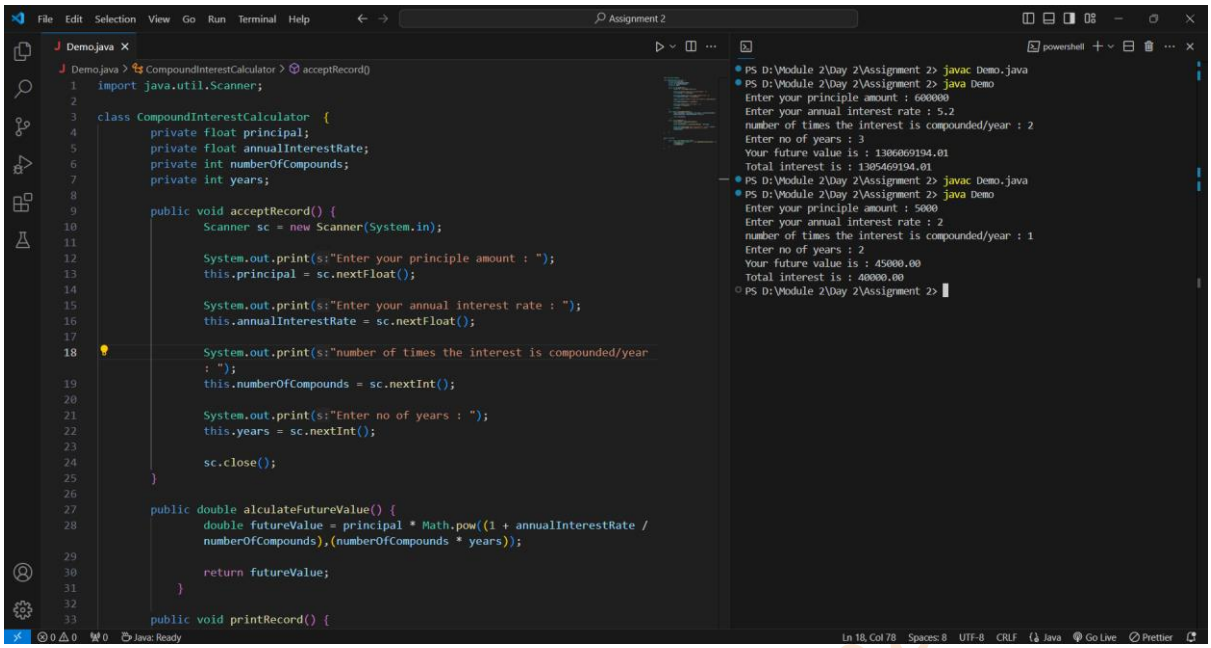
Develop a system to compute the future value of an investment with compound interest. The system should:

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.

```
import java.util.Scanner;

class CompoundInterestCalculator  {
        private float principal;
        private float annualInterestRate;
        private int numberOfCompounds;
        private int years;

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

```
            System.out.print("Enter your principle amount : ");
            this.principal = sc.nextFloat();

            System.out.print("Enter your annual interest rate : ");
            this.annualInterestRate = sc.nextFloat();

            System.out.print("number of times the interest is compounded/year : ");
            this.numberOfCompounds = sc.nextInt();

            System.out.print("Enter no of years : ");
            this.years = sc.nextInt();

            sc.close();
        }

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

            return futureValue;
          }

        public void printRecord() {
            double future = alculateFutureValue();
            // float futureValue;
            double totalInterest = alculateFutureValue() - principal;

            System.out.printf("Your future value is : %.2f%n", future);
            System.out.printf("Total interest is : %.2f%n", totalInterest);

        }
}


public class Demo
{
        public static void main(String[] args) {
            CompoundInterestCalculator   l = new CompoundInterestCalculator  ();
            l.acceptRecord();
            l.printRecord();

        }
}
```



3. BMI (Body Mass Index) Tracker

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

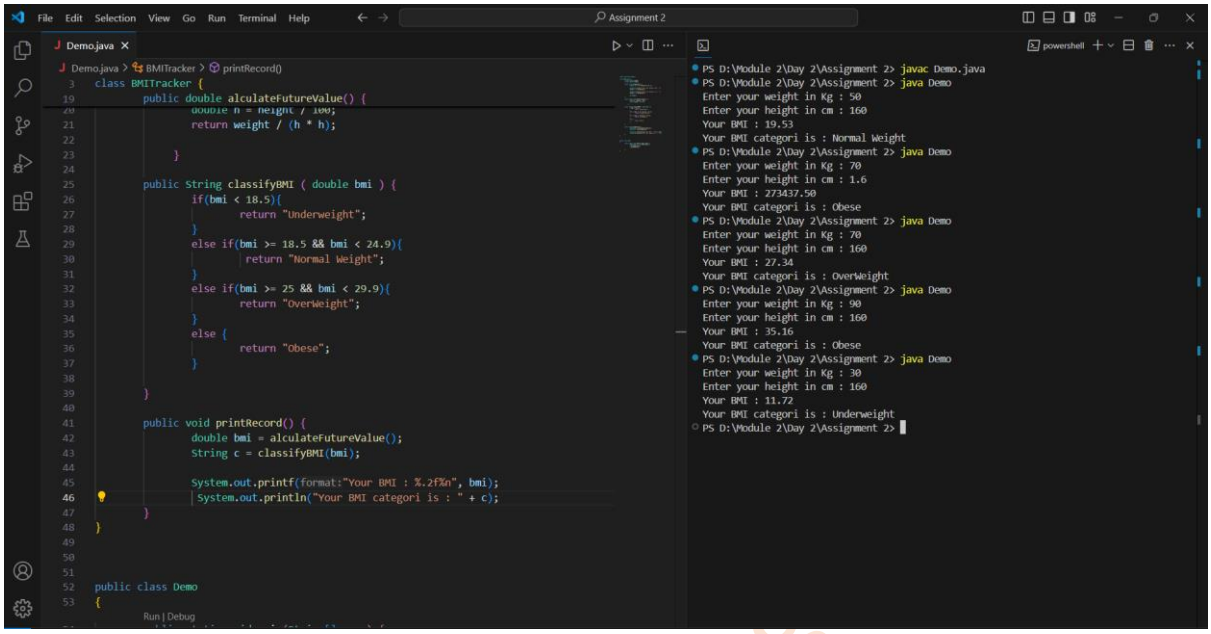
1. Accept weight (in kilograms) and height (in meters) from the user.
2. Calculate the BMI using the formula:
   - **BMI Calculation:** `BMI = weight / (height * height)`
3. Classify the BMI into one of the following categories:
   - Underweight: BMI < 18.5
   - Normal weight: 18.5 ≤ BMI < 24.9
   -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

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

```
import java.util.Scanner;

class BMITracker {
    private double weight;
    private double height;

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

        System.out.print("Enter your weight in Kg : ");
        weight = sc.nextDouble();

        System.out.print("Enter your height in cm : ");
```

```
        height = sc.nextDouble();

        sc.close();
    }

    public double alculateFutureValue() {
        double h = height / 100;
        return weight / (h * h);

      }

    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 = alculateFutureValue();
        String c = classifyBMI(bmi);

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


public class Demo
{
    public static void main(String[] args) {
        BMITracker l = new BMITracker();
        l.acceptRecord();
        l.printRecord();

    }
```

}



4. Discount Calculation for Retail Sales

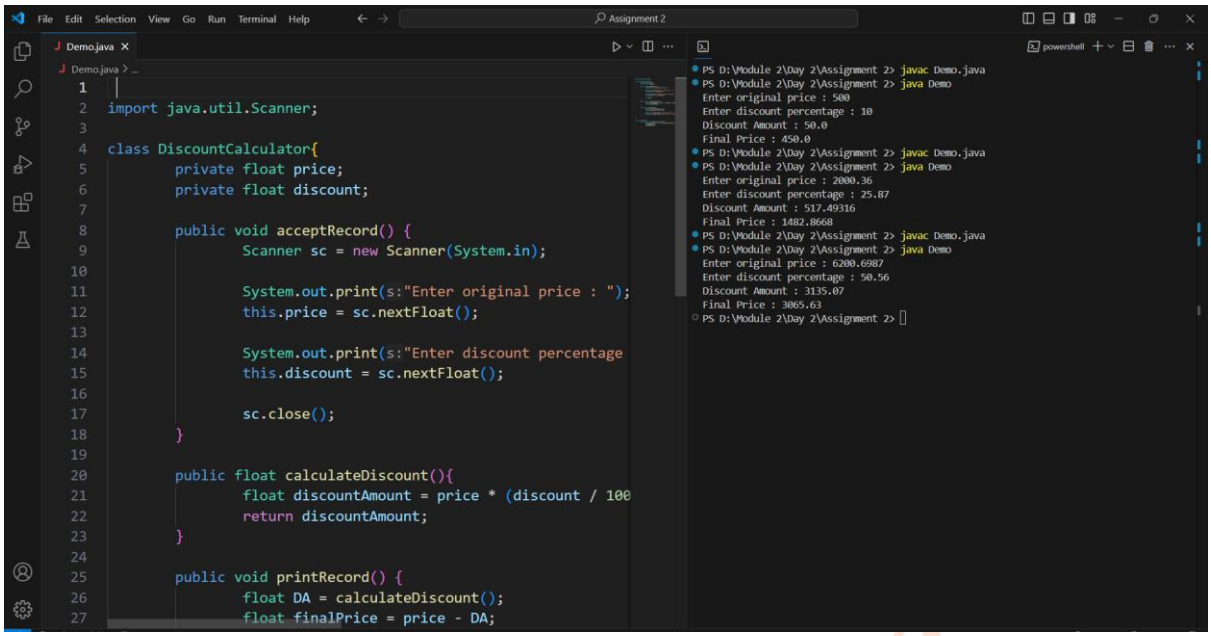
Design a system to calculate the final price of an item after applying a discount. The system should:

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.

```
import java.util.Scanner;

class DiscountCalculator{
    private float price;
    private float discount;

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

        System.out.print("Enter original price : ");
```

```
        this.price = sc.nextFloat();

        System.out.print("Enter discount percentage : ");
        this.discount = sc.nextFloat();

        sc.close();
    }

    public float calculateDiscount(){
        float discountAmount = price * (discount / 100);
        return discountAmount;
    }

    public void printRecord() {
        float DA = calculateDiscount();
        float finalPrice = price - DA;

        System.out.printf("Discount Amount : %.2f%n", DA);
        System.out.printf("Final Price : %.2f%n", finalPrice);
    }

}

public class Demo {
    public static void main(String[] args) {
        DiscountCalculator D = new DiscountCalculator();
        D.acceptRecord();
        D.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.

```
import java.util.Scanner;

class TollBoothRevenueManager {
    private float cartoll ;
    private float trucktoll ;
    private float motorcycletoll ;

    private int carcount;
    private int truckcount;
    private int motorcyclecount;
```

```
    public void acceptRecord(int  carcount, int truckcount, int motorcyclecount){
        this.truckcount = truckcount;
        this.carcount = carcount;
        this.motorcyclecount = motorcyclecount;
    }

    public void setTollRates(float cartoll, float trucktoll, float motorcycletoll) {
        this.trucktoll = trucktoll;
        this.cartoll = cartoll;
        this.motorcycletoll = motorcycletoll;
    }

    public float calculateRevenue() {
        return (cartoll * carcount) + (trucktoll * truckcount) + (motorcycletoll *
motorcyclecount);
    }

    public void printRecord() {
        float r = calculateRevenue();
        int c = carcount + truckcount + motorcyclecount;

        System.out.printf("Total no. of vehicles : %d%n", c);
        System.out.printf("Total revenue collected : %.2f%n", r );
    }
}

public class Demo{
    public static void main(String[] args) {
        TollBoothRevenueManager t = new TollBoothRevenueManager();

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter car toll : ");
        float cartoll = sc.nextFloat();

        System.out.print("Enter truck toll : ");
        float trucktoll = sc.nextFloat();

        System.out.print("Enter motorcycle toll : ");
        float motorcycletoll = sc.nextFloat();
        t.setTollRates(cartoll, trucktoll, motorcycletoll);

        System.out.print("Enter car count : ");
        int carcount = sc.nextInt();

        System.out.print("Enter truck count : ");
```

```
        int truckcount = sc.nextInt();

        System.out.print("Enter motorcycle count : ");
        int motorcyclecount = sc.nextInt();

        t.acceptRecord(carcount, truckcount, motorcyclecount);
        t.printRecord();

        sc.close();
    }
}
```