**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.

Code:

```
package org.example.loancalculator;

public class Main {

	public static void main(String[] args) {
		LoanAmortizationCalculator p1 = new LoanAmortizationCalculator();
		p1.acceptRecord();
		p1.calculateMonthlyPayment();
		p1.printRecord();
	}

}
```

# ASSIGNMENT NO.3

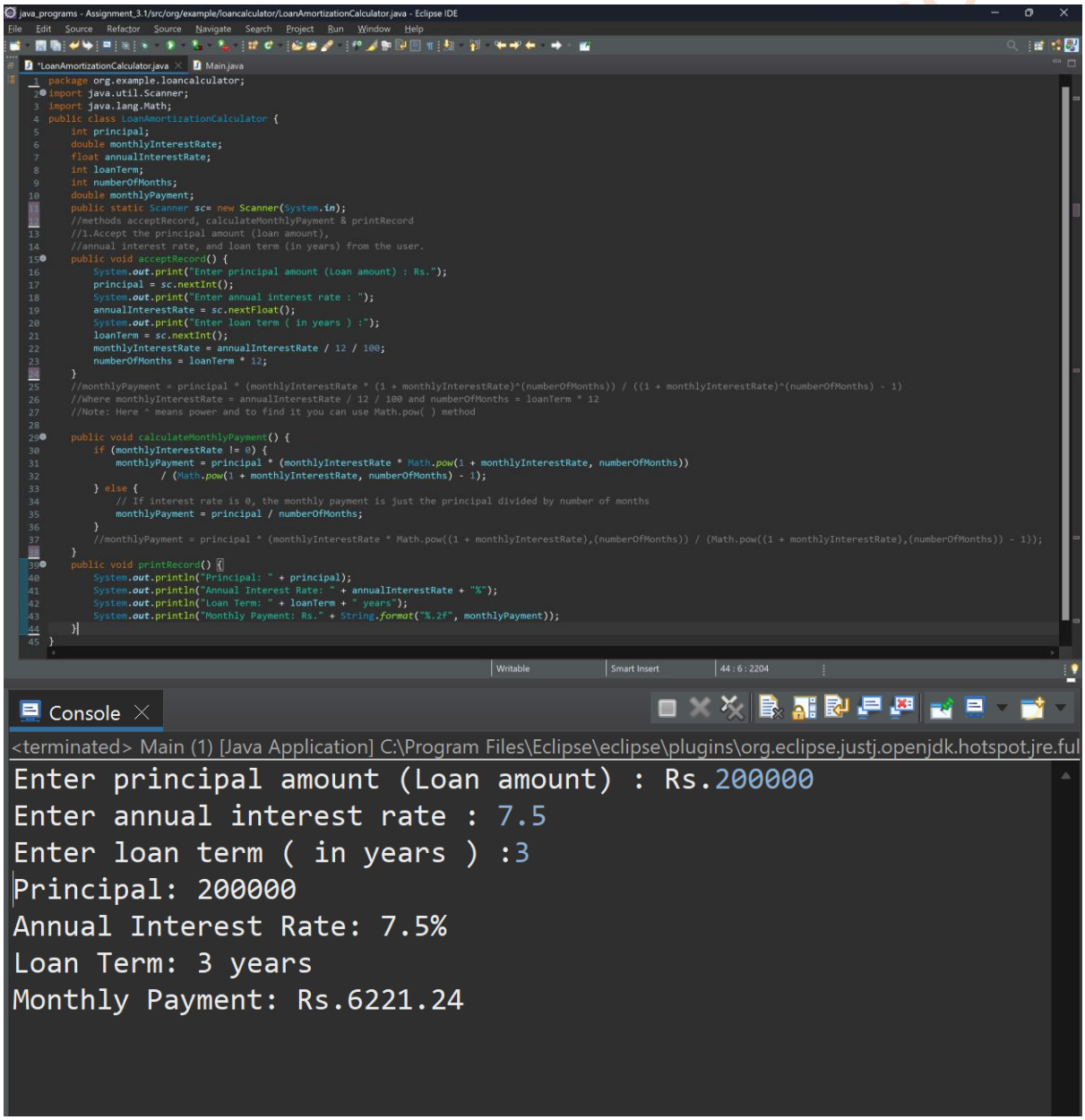
LoanAmortizationCalculator.java  Main.java

```java
package org.example.loancalculator;

public class Main {

    public static void main(String[] args) {
        LoanAmortizationCalculator p1 = new LoanAmortizationCalculator();
        p1.acceptRecord();
        p1.calculateMonthlyPayment();
        p1.printRecord();
    }

}
```

**Class LoanAmortizationCalculator :**

```java
package org.example.loancalculator;
import java.util.Scanner;
import java.lang.Math;
public class LoanAmortizationCalculator {
        int principal;
        double monthlyInterestRate;
        float annualInterestRate;
        int loanTerm;
        int numberOfMonths;
        double monthlyPayment;
        public static Scanner sc= new Scanner(System.in);
        //methods acceptRecord, calculateMonthlyPayment & printRecord
        //1.Accept the principal amount (loan amount),
        //annual interest rate, and loan term (in years) from the user.
        public void acceptRecord() {
                System.out.print("Enter principal amount (Loan amount) : Rs.");
                principal = sc.nextInt();
                System.out.print("Enter annual interest rate : ");
                annualInterestRate = sc.nextFloat();
                System.out.print("Enter loan term ( in years ) :");
                loanTerm = sc.nextInt();
                monthlyInterestRate = annualInterestRate / 12 / 100;
                numberOfMonths = loanTerm * 12;
        }
        //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

        public void calculateMonthlyPayment() {
                if (monthlyInterestRate != 0) {
       monthlyPayment = principal * (monthlyInterestRate * Math.pow(1 + monthlyInterestRate,
numberOfMonths))
            / (Math.pow(1 + monthlyInterestRate, numberOfMonths) - 1);
    } else {
       // If interest rate is 0, the monthly payment is just the principal divided by number of months
       monthlyPayment = principal / numberOfMonths;
```

```
    }
                //monthlyPayment = principal * (monthlyInterestRate * Math.pow((1 +
monthlyInterestRate),(numberOfMonths)) / (Math.pow((1 +
monthlyInterestRate),(numberOfMonths)) - 1));
        }
        public void printRecord() {
                System.out.println("Principal: " + principal);
    System.out.println("Annual Interest Rate: " + annualInterestRate + "%");
    System.out.println("Loan Term: " + loanTerm + " years");
    System.out.println("Monthly Payment: Rs." + String.format("%.2f", monthlyPayment));
        }
}
```



## 2. Compound Interest Calculator for Investment

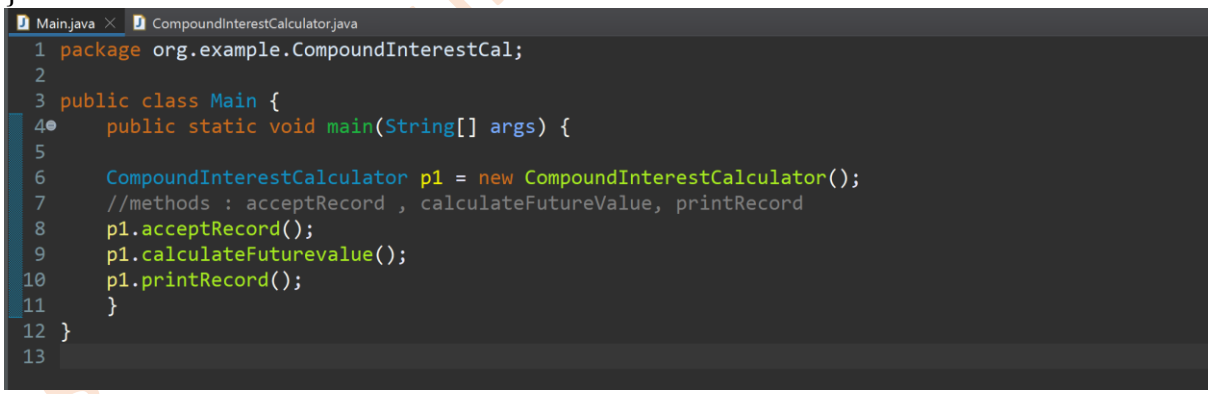
Develop a system to compute the future value of an investment with compound interest. The system should:

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.

Code:

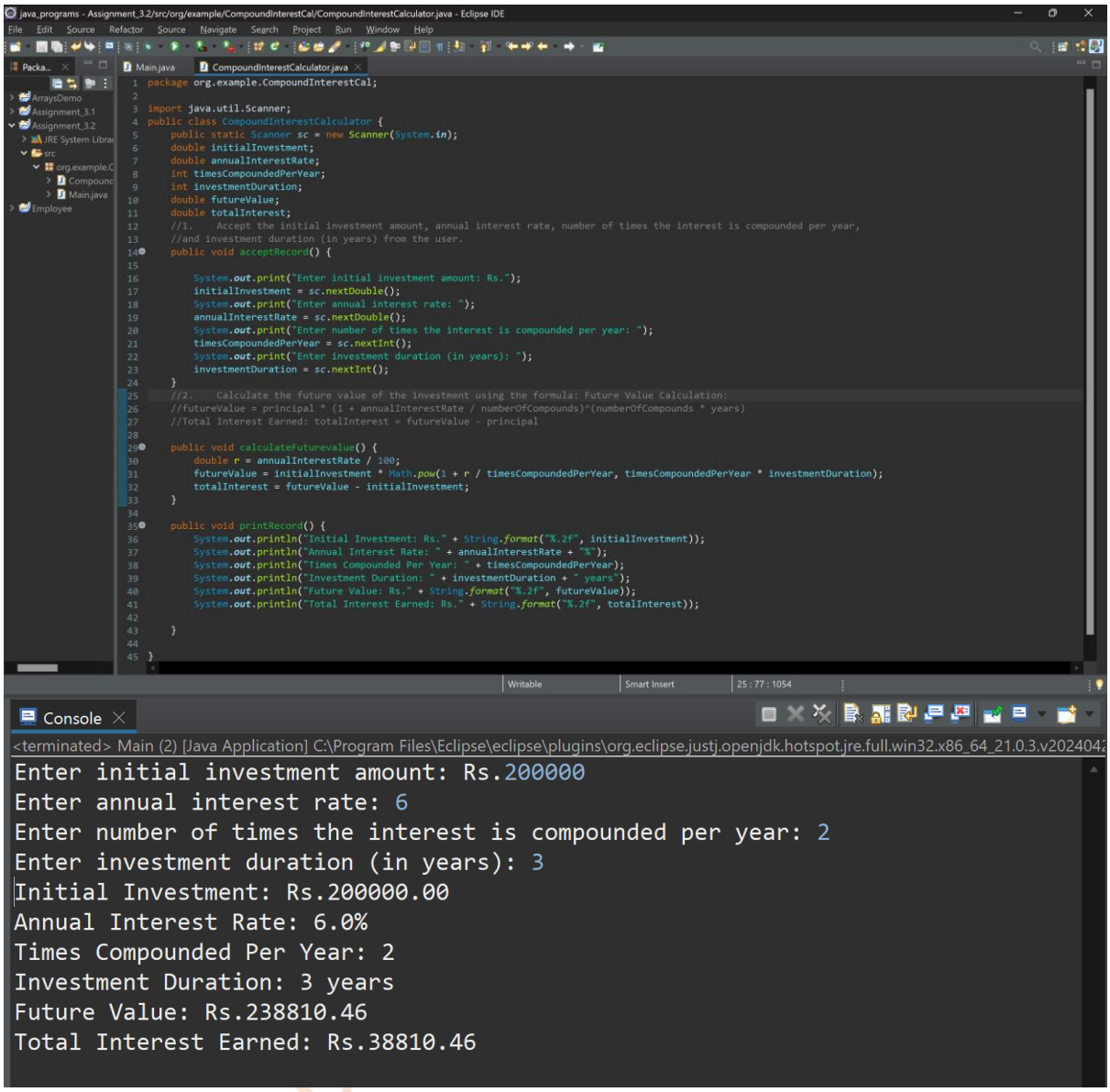
```
package org.example.CompoundInterestCal;

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

        CompoundInterestCalculator p1 = new CompoundInterestCalculator();
        //methods : acceptRecord , calculateFutureValue, printRecord
        p1.acceptRecord();
        p1.calculateFuturevalue();
        p1.printRecord();
        }
}
```




Class CompoundInterestCalculator:

```
package org.example.CompoundInterestCal;

import java.util.Scanner;
public class CompoundInterestCalculator {
        public static Scanner sc = new Scanner(System.in);
        double initialInvestment;
   double annualInterestRate;
```

```
    int timesCompoundedPerYear;
    int investmentDuration;
    double futureValue;
    double totalInterest;


        //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.
        public void acceptRecord() {

                System.out.print("Enter initial investment amount: Rs.");
                initialInvestment = sc.nextDouble();
                System.out.print("Enter annual interest rate: ");
                annualInterestRate = sc.nextDouble();
                System.out.print("Enter number of times the interest is compounded per year:
");
                timesCompoundedPerYear = sc.nextInt();
                System.out.print("Enter investment duration (in years): ");
                investmentDuration = sc.nextInt();
        }
        //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

        public void calculateFuturevalue() {
                double r = annualInterestRate / 100;
      futureValue = initialInvestment * Math.pow(1 + r / timesCompoundedPerYear,
timesCompoundedPerYear * investmentDuration);
      totalInterest = futureValue - initialInvestment;
        }

        public void printRecord() {
                System.out.println("Initial Investment: Rs." + String.format("%.2f",
initialInvestment));
      System.out.println("Annual Interest Rate: " + annualInterestRate + "%");
      System.out.println("Times Compounded Per Year: " + timesCompoundedPerYear);
      System.out.println("Investment Duration: " + investmentDuration + " years");
      System.out.println("Future Value: Rs." + String.format("%.2f", futureValue));
      System.out.println("Total Interest Earned: Rs." + String.format("%.2f", totalInterest));

        }

}
```




## 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.

# ASSIGNMENT NO.3

Code:

```
package org.example.BMItracker;

public class Main {
	public static void main(String[] args) {
		BMITracker b = new BMITracker();
//		methods acceptRecord, calculateBMI, classifyBMI & printRecord
		b.acceptRecord();
		b.calculateBMI();
		b.classifyBMI();
		b.printRecord();
	}
}
```

Main.java BMITracker.java

```
package org.example.BMItracker;

public class Main {
    public static void main(String[] args) {
        BMITracker b = new BMITracker();
//      methods acceptRecord, calculateBMI, classifyBMI & printRecord
        b.acceptRecord();
        b.calculateBMI();
        b.classifyBMI();
        b.printRecord();
    }
}
```

```
package org.example.BMItracker;
import java.util.Scanner;
public class BMITracker {
	public static Scanner sc = new Scanner(System.in);
	int weight;
	float height;
	float bmi;
	String bmiClass;
	//methods acceptRecord, calculateBMI, classifyBMI & printRecord
//	1.	Accept weight (in kilograms) and height (in meters) from the user.
	public void acceptRecord() {
		System.out.print("Enter weight (in kg): ");
		weight = sc.nextInt();
		System.out.print("Enter height (in meters): ");
		height = sc.nextFloat();
	}
//	2.	Calculate the BMI using the formula: BMI Calculation: BMI = weight / (height
* height)
	public void calculateBMI() {
		bmi = weight / ( height * height );
	}
//	3.	Classify the BMI into one of the following categories:
		//	o	Under weight: BMI < 18.5
		//	o	Normal weight: 18.5 ≤ BMI < 24.9
```

```
        //      o       Overweight: 25 ≤ BMI < 29.9
        //      o       Obese: BMI ≥ 30
    public void classifyBMI() {
        if ( bmi < 18.5 ) {
            bmiClass = "Underweight";
        }else if( bmi <= 24.9 && bmi >= 18.5 ) {
            bmiClass = "Normal weight";
        }else if( bmi <= 29.9 && bmi >= 25 ) {
            bmiClass = "Overweight";
        }else {
            bmiClass = "Obese";
        }
    }
//  4.  Display the BMI value and its classification.
    public void printRecord() {

        System.out.println("Weight: "+ weight + "kg");
        System.out.println("Height: "+ height + "m");
        System.out.println("BMI: "+ String.format("%.2f",bmi));
        System.out.println("BMI classification: " + bmiClass);
    }
}
```

Main.java  BMITracker.java

```
package org.example.BMITracker;
import java.util.Scanner;
public class BMITracker {
    public static Scanner sc = new Scanner(System.in);
    int weight;
    float height;
    float bmi;
    String bmiClass;
    //methods acceptRecord, calculateBMI, classifyBMI & printRecord
//  1.  Accept weight (in kilograms) and height (in meters) from the user.
    public void acceptRecord() {
        System.out.print("Enter weight (in kg): ");
        weight = sc.nextInt();
        System.out.print("Enter height (in meters): ");
        height = sc.nextFloat();
    }
//  2.  Calculate the BMI using the formula: BMI Calculation: BMI = weight / (height * height)
    public void calculateBMI() {
        bmi = weight / ( height * height );
    }
//  3.  Classify the BMI into one of the following categories:
        //  o   Under weight: BMI < 18.5
        //  o   Normal weight: 18.5 ≤ BMI < 24.9
        //  o   Overweight: 25 ≤ BMI < 29.9
        //  o   Obese: BMI ≥ 30
    public void classifyBMI() {
        if ( bmi < 18.5 ) {
            bmiClass = "Underweight";
        }else if( bmi <= 24.9 && bmi >= 18.5 ) {
            bmiClass = "Normal weight";
        }else if( bmi <= 29.9 && bmi >= 25 ) {
            bmiClass = "Overweight";
        }else {
            bmiClass = "Obese";
        }
    }
//  4.  Display the BMI value and its classification.
    public void printRecord() {

        System.out.println("Weight: "+ weight + "kg");
        System.out.println("Height: "+ height + "m");
        System.out.println("BMI: "+ String.format("%.2f",bmi));
        System.out.println("BMI classification: " + bmiClass);
    }
}
```

Writable  Smart Insert  26 : 32 : 928

```
Console
<terminated> Main (3) [Java Application] C:\Program Files\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jr
Enter weight (in kg): 70
Enter height (in meters): 6.5
Weight: 70kg
Height: 6.5m
BMI: 1.66
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.

Code:

```
package org.example.retailsales;

public class Main {
	public static void main(String[] args) {
		DiscountCalculator d = new DiscountCalculator();
		//methods acceptRecord, calculateDiscount & printRecord
		d.acceptRecord();
		d.calculateDiscount();
		d.printRecord();
	}
}
```

Class:

```
package org.example.retailsales;
import java.util.Scanner;
public class DiscountCalculator {
	public static Scanner sc = new Scanner(System.in);
	int originalPrice;
```

```
        int discountRate;
        float discountAmount;
        float finalPrice;
        public void acceptRecord() {
                // TODO Auto-generated method stub
                System.out.print("Enter original price of an item: ");
                originalPrice = sc.nextInt();
                System.out.print("Enter discount percentage: ");
                discountRate = sc.nextInt();

        }       public void calculateDiscount() {
                // TODO Auto-generated method stub
                discountAmount = originalPrice * (discountRate / 100.0f);
                finalPrice = originalPrice - discountAmount;
        }
        public void printRecord() {
                // TODO Auto-generated method stub
                System.out.printf("Discount amount: ₹%.2f%n", discountAmount);
                System.out.printf("Final price: ₹%.2f%n", finalPrice);
        }
}
```

Main.java | DiscountCalculator.java

```
1  package org.example.retailsales;
2
3  import java.util.Scanner;
4
5  public class DiscountCalculator {
6
7      public static Scanner sc = new Scanner(System.in);
8      int originalPrice;
9      int discountRate;
10     float discountAmount;
11     float finalPrice;
12 //  1.  Accept the original price of an item and the discount percentage from the user.
13 //  2.  Calculate the discount amount and the final price using the following formulas:
14 //      o   Discount Amount Calculation: discountAmount = originalPrice * (discountRate / 100)
15 //      o   Final Price Calculation: finalPrice = originalPrice - discountAmount
16 //  3.  Display the discount amount and the final price of the item, in Indian Rupees (₹).
17
18     public void acceptRecord() {
19         // TODO Auto-generated method stub
20         System.out.print("Enter original price of an item: ");
21         originalPrice = sc.nextInt();
22         System.out.print("Enter discount percentage: ");
23         discountRate = sc.nextInt();
24
25     }
26
27     public void calculateDiscount() {
28         // TODO Auto-generated method stub
29         discountAmount = originalPrice * (discountRate / 100.0f);
30         finalPrice = originalPrice - discountAmount;
31     }
32
33     public void printRecord() {
34         // TODO Auto-generated method stub
35         System.out.printf("Discount amount: ₹%.2f%n", discountAmount);
36         System.out.printf("Final price: ₹%.2f%n", finalPrice);
37
38     }
39
40 }
```

```java
package org.example.retailsales;

public class Main {
    public static void main(String[] args) {
        DiscountCalculator d = new DiscountCalculator();
        //methods acceptRecord, calculateDiscount & printRecord
        d.acceptRecord();
        d.calculateDiscount();
        d.printRecord();
    }
}
```

```
Enter original price of an item: 1000
Enter discount percentage: 20
Discount amount: ₹200.00
Final price: ₹800.00
```

## 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.

Code:

```java
package org.example.tollboothrevenue;

public class Main {
        public static void main(String[] args) {
                TollBoothRevenueManager t = new TollBoothRevenueManager();
                //methods acceptRecord, setTollRates, calculateRevenue & printRecord
                t.setTollRates();
                t.acceptRecord();
                t.calculateRevenue();
                t.printRecord();
        }
```

}

```java
package org.example.tollboothrevenue;

public class Main {
    public static void main(String[] args) {
        TollBoothRevenueManager t = new TollBoothRevenueManager();
        //methods acceptRecord, setTollRates, calculateRevenue & printRecord
        t.setTollRates();
        t.acceptRecord();
        t.calculateRevenue();
        t.printRecord();
    }
}
```

```java
package org.example.tollboothrevenue;
import java.util.Scanner;

public class TollBoothRevenueManager {
        public static Scanner sc = new Scanner(System.in);
        float carTollRate;
        float truckTollRate;
        float motorcycleTollRate;
   private int numCars;
   private int numTrucks;
   private int numMotorcycles;
   private float totalRevenue;
   private int totalVehicles;

//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 (₹).

   public void setTollRates() {
      System.out.print("Enter toll rate for Car (₹): ");
      carTollRate = sc.nextFloat();
      System.out.print("Enter toll rate for Truck (₹): ");
      truckTollRate = sc.nextFloat();
      System.out.print("Enter toll rate for Motorcycle (₹): ");
      motorcycleTollRate = sc.nextFloat();
   }
   public void acceptRecord() {
      System.out.print("Enter the number of cars: ");
      numCars = sc.nextInt();
      System.out.print("Enter the number of trucks: ");
      numTrucks = sc.nextInt();
      System.out.print("Enter the number of motorcycles: ");
      numMotorcycles = sc.nextInt();
   }
   public void calculateRevenue() {
      totalRevenue = (numCars * carTollRate) + (numTrucks * truckTollRate) +
(numMotorcycles * motorcycleTollRate);
      totalVehicles = numCars + numTrucks + numMotorcycles;
   }
   public void printRecord() {
```

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

TollBoothRevenueManager.java | Main.java

```java
package org.example.tollboothrevenue;
import java.util.Scanner;

public class TollBoothRevenueManager {
    public static Scanner sc = new Scanner(System.in);
    float carTollRate;
    float truckTollRate;
    float motorcycleTollRate;
    private int numCars;
    private int numTrucks;
    private int numMotorcycles;
    private float totalRevenue;
    private int totalVehicles;

//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 (₹).

    public void setTollRates() {
        System.out.print("Enter toll rate for Car (₹): ");
        carTollRate = sc.nextFloat();
        System.out.print("Enter toll rate for Truck (₹): ");
        truckTollRate = sc.nextFloat();
        System.out.print("Enter toll rate for Motorcycle (₹): ");
        motorcycleTollRate = sc.nextFloat();
    }
    public void acceptRecord() {
        System.out.print("Enter the number of cars: ");
        numCars = sc.nextInt();
        System.out.print("Enter the number of trucks: ");
        numTrucks = sc.nextInt();
        System.out.print("Enter the number of motorcycles: ");
        numMotorcycles = sc.nextInt();
    }
    public void calculateRevenue() {
        totalRevenue = (numCars * carTollRate) + (numTrucks * truckTollRate) + (numMotorcycles * motorcycleTollRate);
        totalVehicles = numCars + numTrucks + numMotorcycles;
    }
    public void printRecord() {
        System.out.printf("Total number of vehicles: %d%n", totalVehicles);
        System.out.printf("Total revenue collected: ₹%.2f%n", totalRevenue);
    }
}
```

Console

```
<terminated> Main (5) [Java Application] C:\Program Files\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.ful
Enter toll rate for Car (₹): 50
Enter toll rate for Truck (₹): 100
Enter toll rate for Motorcycle (₹): 30
Enter the number of cars: 5
Enter the number of trucks: 3
Enter the number of motorcycles: 8
Total number of vehicles: 16
Total revenue collected: ₹790.00
```