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 q1;

import java.util.Scanner;



class Calculator {

  double principal;

  double annualInterestRate;

  int loanTerm;



  void acceptRecord() {

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

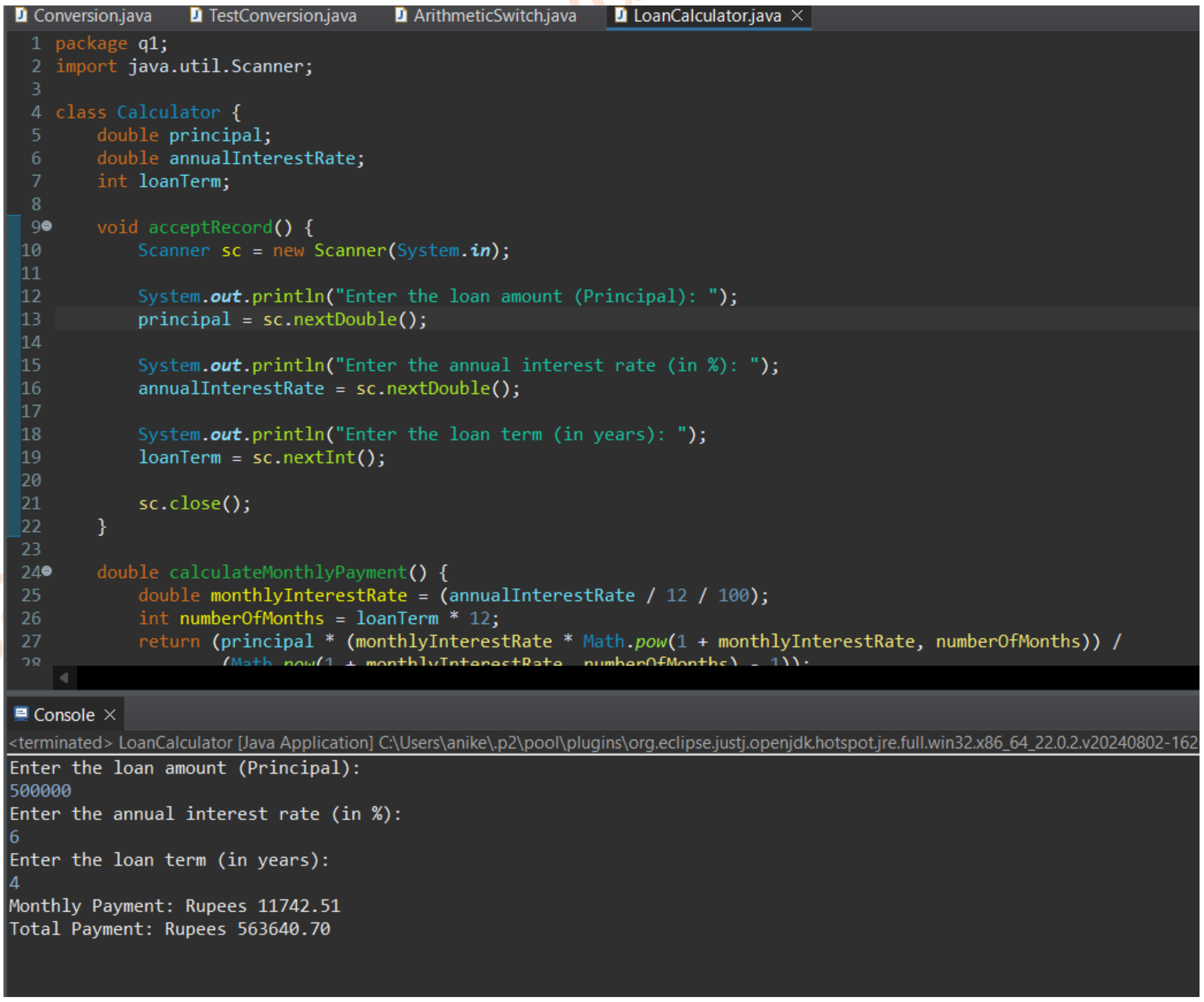
sandeepkulange@gmail.com

```
        System.out.println("Enter the loan amount (Principal): ");
        principal = sc.nextDouble();

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

        System.out.println("Enter the loan term (in years): ");
        loanTerm = sc.nextInt();

        sc.close();
    }

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

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

        System.out.printf("Monthly Payment: Rupees %.2f\n", monthlyPayment);
```

sandeepkulange@gmail.com

```
        System.out.printf("Total Payment: Rupees %.2f\n", totalPayment);
    }
}


public class LoanCalculator {
    public static void main(String[] args) {
        Calculator calculator = new Calculator();
        calculator.acceptRecord();
        calculator.printRecord();
    }
}
```

@gmail.com



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

class Calculator {
    double principal;
    double annualInterestRate;
    int loanTerm;

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

        System.out.println("Enter the loan amount (Principal): ");
        principal = sc.nextDouble();

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

        System.out.println("Enter the loan term (in years): ");
        loanTerm = sc.nextInt();

        sc.close();
    }

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

```
<terminated> LoanCalculator [Java Application] C:\Users\anike\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.2.v20240802-162
Enter the loan amount (Principal):
500000
Enter the annual interest rate (in %):
6
Enter the loan term (in years):
4
Monthly Payment: Rupees 11742.51
Total Payment: Rupees 563640.70
```

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 (₹).

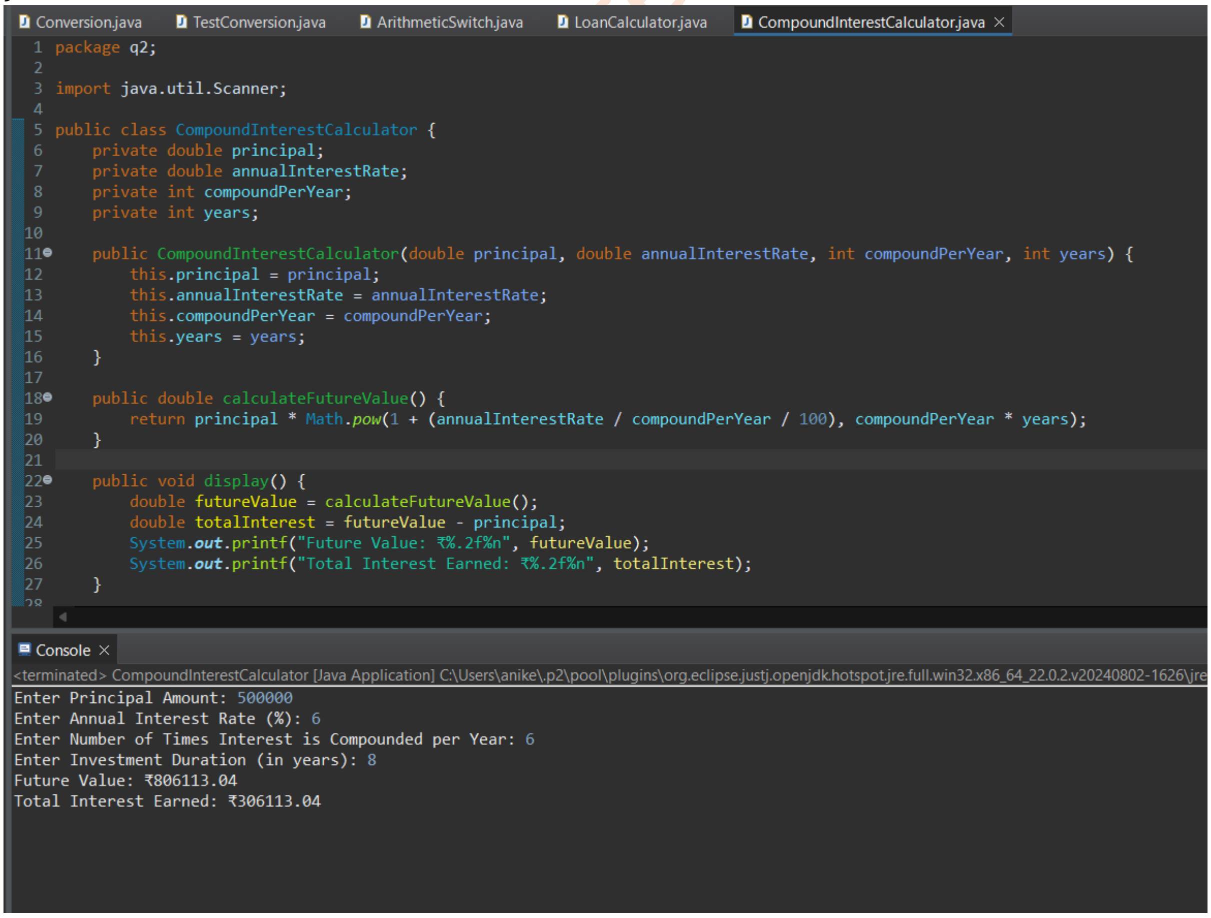
Define class CompoundInterestCalculator with methods acceptRecord , calculateFutureValue, printRecord and test the functionality in main method.

```
        package q2;

import java.util.Scanner;

public class CompoundInterestCalculator {
    private double principal;
    private double annualInterestRate;
    private int compoundPerYear;
    private int years;

    public CompoundInterestCalculator(double principal, double annualInterestRate, int
compoundPerYear, int years) {
        this.principal = principal;
        this.annualInterestRate = annualInterestRate;
        this.compoundPerYear = compoundPerYear;
        this.years = years;
    }

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

    public void display() {
        double futureValue = calculateFutureValue();
        double totalInterest = futureValue - principal;
        System.out.printf("Future Value: ₹%.2f%n", futureValue);
        System.out.printf("Total Interest Earned: ₹%.2f%n", totalInterest);
    }
```

[illegible]kulange@gmail.com

```
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    System.out.print("Enter Principal Amount: ");
    double principal = sc.nextDouble();

    System.out.print("Enter Annual Interest Rate (%): ");
    double annualInterestRate = sc.nextDouble();

    System.out.print("Enter Number of Times Interest is Compounded per Year: ");
    int compoundPerYear = sc.nextInt();

    System.out.print("Enter Investment Duration (in years): ");
    int years = sc.nextInt();

    CompoundInterestCalculator calculator = new CompoundInterestCalculator(principal,
annualInterestRate, compoundPerYear, years);
    calculator.display();

    sc.close();
  }
}
```

e@gmail.com



```
package q2;

import java.util.Scanner;

public class CompoundInterestCalculator {
    private double principal;
    private double annualInterestRate;
    private int compoundPerYear;
    private int years;

    public CompoundInterestCalculator(double principal, double annualInterestRate, int compoundPerYear, int years) {
        this.principal = principal;
        this.annualInterestRate = annualInterestRate;
        this.compoundPerYear = compoundPerYear;
        this.years = years;
    }

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

    public void display() {
        double futureValue = calculateFutureValue();
        double totalInterest = futureValue - principal;
        System.out.printf("Future Value: ₹%.2f%n", futureValue);
        System.out.printf("Total Interest Earned: ₹%.2f%n", totalInterest);
    }
```

```
<terminated> CompoundInterestCalculator [Java Application] C:\Users\anike\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.2.v20240802-1626\jre
Enter Principal Amount: 500000
Enter Annual Interest Rate (%): 6
Enter Number of Times Interest is Compounded per Year: 6
Enter Investment Duration (in years): 8
Future Value: ₹806113.04
Total Interest Earned: ₹306113.04
```

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.

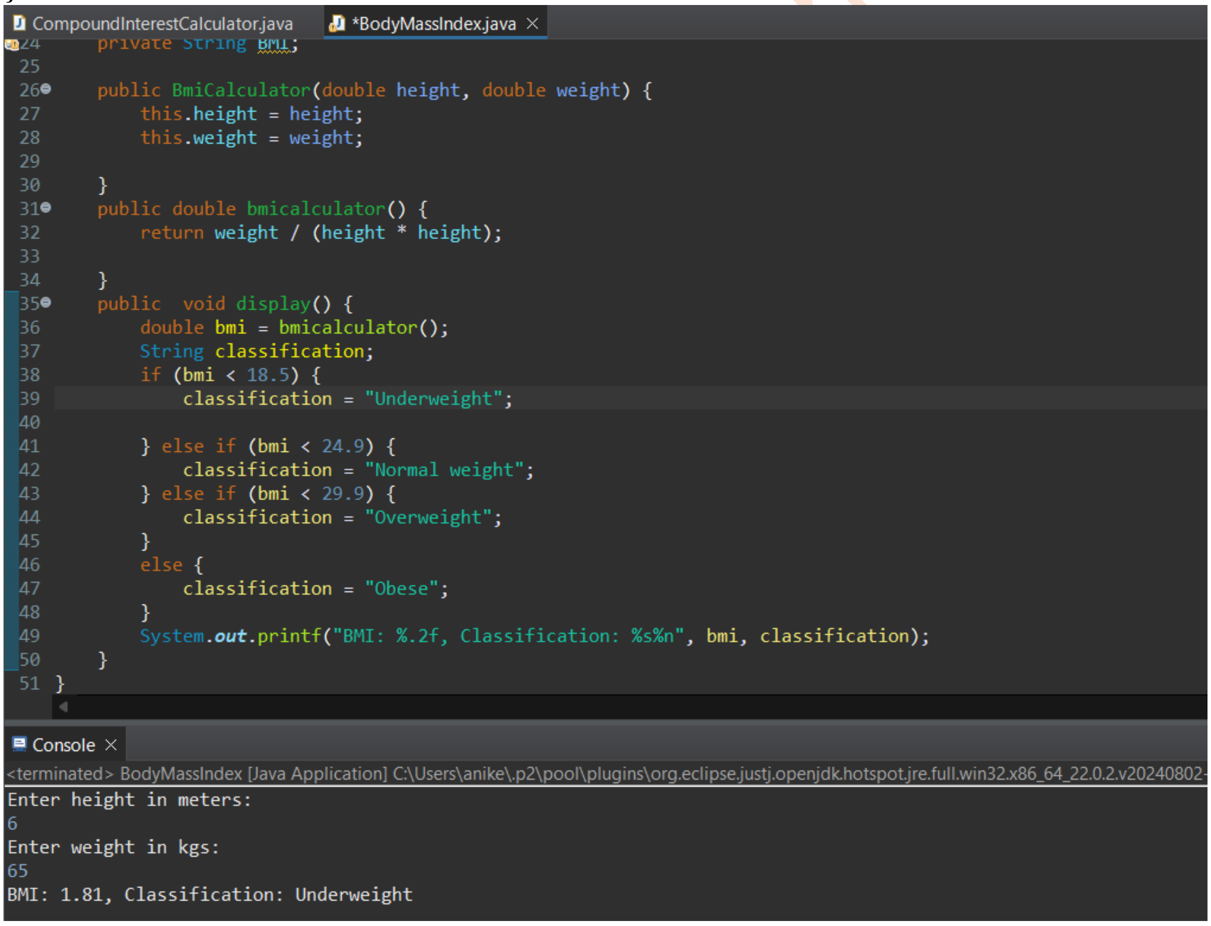
Define class BMITracker with methods acceptRecord, calculateBMI, classifyBMI & printRecord and test the functionality in main method.

```
        package q3;

import java.util.Scanner;

public class BodyMassIndex {
        public static void main(String[] args) {
                Scanner sc = new Scanner(System.in);

                System.out.println("Enter height in meters: ");
                double height = sc.nextDouble();

                System.out.println("Enter weight in kgs: ");
                double weight = sc.nextDouble();

                BmiCalculator calculator = new BmiCalculator(height , weight);
                calculator.display();

        }
}

class BmiCalculator{
        private double weight;
        private double height;
        private String BMI;

        public BmiCalculator(double height, double weight) {
                this.height = height;
                this.weight = weight;

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

sandeepkulange@gmail.com

```
        }
        public  void display() {
                double bmi = bmicalculator();
                String classification;
                if (bmi < 18.5) {
                        classification = "Underweight";

                } else if (bmi < 24.9) {
                        classification = "Normal weight";
                } else if (bmi < 29.9) {
                        classification = "Overweight";
                }
                else {
                        classification = "Obese";
                }
                System.out.printf("BMI: %.2f, Classification: %s%n", bmi, classification);
        }
}
```

mail.com



```
    private String BMI;

    public BmiCalculator(double height, double weight) {
        this.height = height;
        this.weight = weight;

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

    }
    public  void display() {
        double bmi = bmicalculator();
        String classification;
        if (bmi < 18.5) {
            classification = "Underweight";

        } else if (bmi < 24.9) {
            classification = "Normal weight";
        } else if (bmi < 29.9) {
            classification = "Overweight";
        }
        else {
            classification = "Obese";
        }
        System.out.printf("BMI: %.2f, Classification: %s%n", bmi, classification);
    }
}
```

```
<terminated> BodyMassIndex [Java Application] C:\Users\anike\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.2.v20240802-
Enter height in meters:
6
Enter weight in kgs:
65
BMI: 1.81, Classification: Underweight
```

4. Discount Calculation for Retail Sales

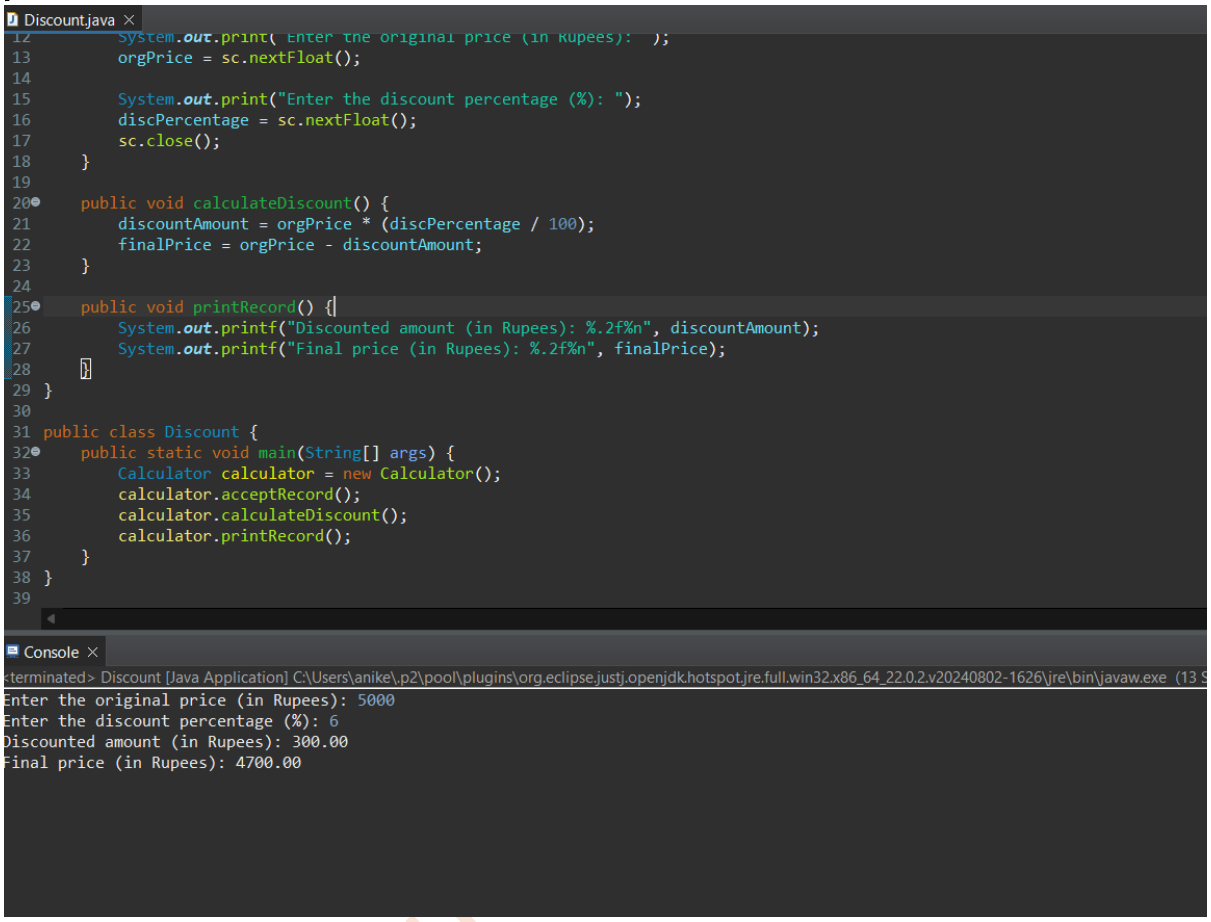
Design a system to calculate the final price of an item after applying a discount. The system should:

ASSIGNMENT NO.3

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 calc;
import java.util.Scanner;

class Calculator {
    private float orgPrice;
    private float discPercentage;
    private float discountAmount;
    private float finalPrice;

    public void acceptRecord() {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the original price (in Rupees): ");
        orgPrice = sc.nextFloat();

        System.out.print("Enter the discount percentage (%): ");
        discPercentage = sc.nextFloat();
        sc.close();
    }

    public void calculateDiscount() {
        discountAmount = orgPrice * (discPercentage / 100);
        finalPrice = orgPrice - discountAmount;
    }

    public void printRecord() {
        System.out.printf("Discounted amount (in Rupees): %.2f%n", discountAmount);
        System.out.printf("Final price (in Rupees): %.2f%n", finalPrice);
    }
}

public class Discount {
    public static void main(String[] args) {
        Calculator calculator = new Calculator();
        calculator.acceptRecord();
        calculator.calculateDiscount();
        calculator.printRecord();
    }
}
```



Discount.java

```
        System.out.print("Enter the original price (in Rupees): ");
        orgPrice = sc.nextFloat();

        System.out.print("Enter the discount percentage (%): ");
        discPercentage = sc.nextFloat();
        sc.close();
    }

    public void calculateDiscount() {
        discountAmount = orgPrice * (discPercentage / 100);
        finalPrice = orgPrice - discountAmount;
    }

    public void printRecord() {
        System.out.printf("Discounted amount (in Rupees): %.2f%n", discountAmount);
        System.out.printf("Final price (in Rupees): %.2f%n", finalPrice);
    }
}

public class Discount {
    public static void main(String[] args) {
        Calculator calculator = new Calculator();
        calculator.acceptRecord();
        calculator.calculateDiscount();
        calculator.printRecord();
    }
}
```

Console

```
<terminated> Discount [Java Application] C:\Users\anike\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.2.v20240802-1626\jre\bin\javaw.exe  (13 S
Enter the original price (in Rupees): 5000
Enter the discount percentage (%): 6
Discounted amount (in Rupees): 300.00
Final price (in Rupees): 4700.00
```

sandeepku

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 (₹).

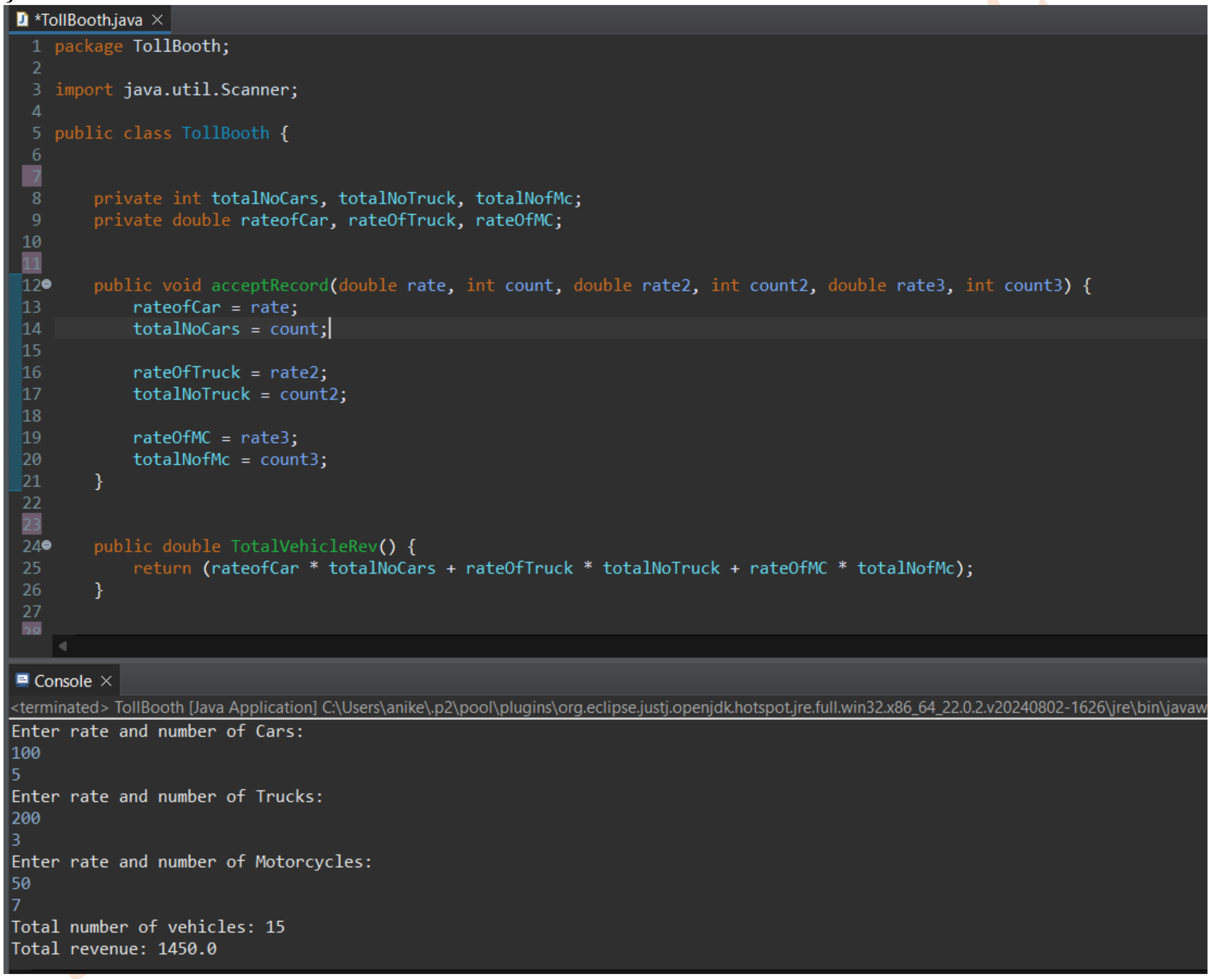
- **Toll Rate Examples:**
  - Car: ₹50.00
  - Truck: ₹100.00
  - Motorcycle: ₹30.00

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

```
package TollBooth;

import java.util.Scanner;

public class TollBooth {


    private int totalNoCars, totalNoTruck, totalNofMc;
    private double rateofCar, rateOfTruck, rateOfMC;


    public void acceptRecord(double rate, int count, double rate2, int count2, double rate3, int
count3) {
        rateofCar = rate;
        totalNoCars = count;

        rateOfTruck = rate2;
        totalNoTruck = count2;

        rateOfMC = rate3;
        totalNofMc = count3;
    }


    public double TotalVehicleRev() {
        return (rateofCar * totalNoCars + rateOfTruck * totalNoTruck + rateOfMC *
totalNofMc);
    }


    public int totalNoOfVehicle() {
        return (totalNoCars + totalNoTruck + totalNofMc);
    }

    public void printRecord() {
        System.out.println("Total number of vehicles: " + totalNoOfVehicle());
        System.out.println("Total revenue: " + TotalVehicleRev());
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        TollBooth tollBooth = new TollBooth();

        // Input data
        System.out.println("Enter rate and number of Cars: ");
        double carRate = sc.nextDouble();
        int carCount = sc.nextInt();

        System.out.println("Enter rate and number of Trucks: ");
        double truckRate = sc.nextDouble();
```

sandeepkulange@gmail.com

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

        System.out.println("Enter rate and number of Motorcycles: ");
        double mcRate = sc.nextDouble();
        int mcCount = sc.nextInt();


        tollBooth.acceptRecord(carRate, carCount, truckRate, truckCount, mcRate, mcCount);

        tollBooth.printRecord();

        sc.close();  // Close the scanner
    }
}
```



*TollBooth.java

```
package TollBooth;

import java.util.Scanner;

public class TollBooth {


    private int totalNoCars, totalNoTruck, totalNofMc;
    private double rateofCar, rateOfTruck, rateOfMC;


    public void acceptRecord(double rate, int count, double rate2, int count2, double rate3, int count3) {
        rateofCar = rate;
        totalNoCars = count;

        rateOfTruck = rate2;
        totalNoTruck = count2;

        rateOfMC = rate3;
        totalNofMc = count3;
    }


    public double TotalVehicleRev() {
        return (rateofCar * totalNoCars + rateOfTruck * totalNoTruck + rateOfMC * totalNofMc);
    }

```

Console

```
<terminated> TollBooth [Java Application] C:\Users\anike\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_22.0.2.v20240802-1626\jre\bin\javaw
Enter rate and number of Cars:
100
5
Enter rate and number of Trucks:
200
3
Enter rate and number of Motorcycles:
50
7
Total number of vehicles: 15
Total revenue: 1450.0
```