ASSIGNMENT NO.4

Note:

- The assignment is designed to practice constructor, getter/setter and toString method.
- Create a separate project for each question and create separate file for each class.
- Try to test the functionality by using menu-driven program.

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 (₹).

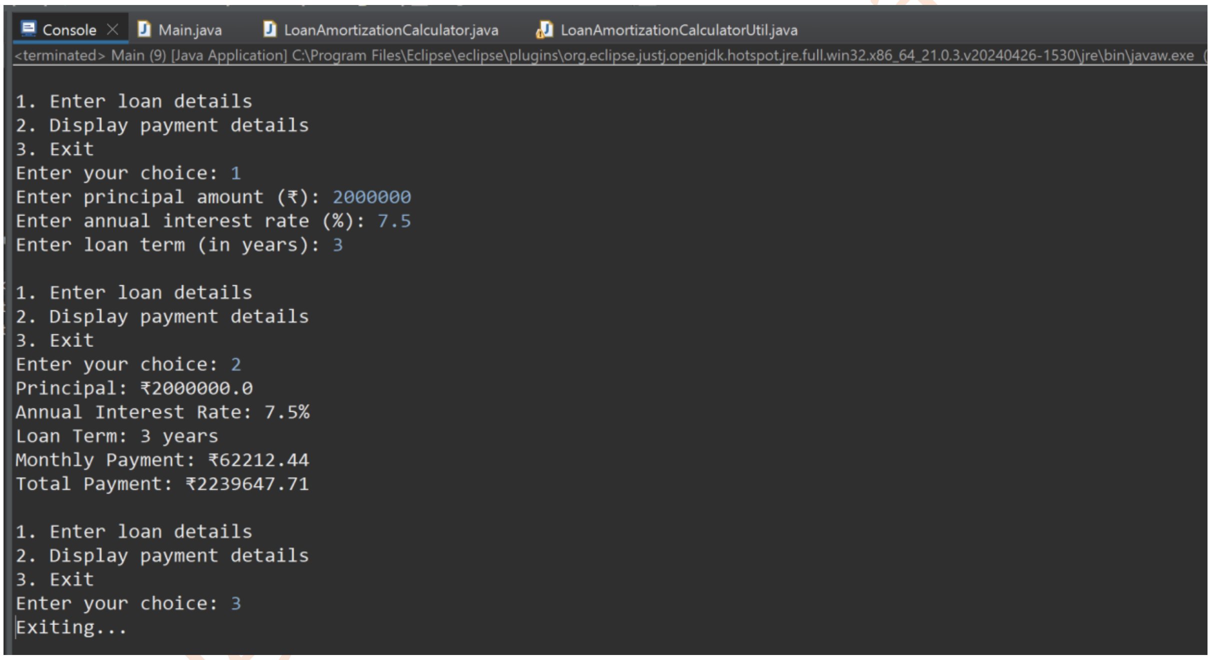
Define the class `LoanAmortizationCalculator` with fields, an appropriate constructor, getter and setter methods, a `toString` method and business logic methods. Define the class `LoanAmortizationCalculatorUtil` with methods `acceptRecord`, `printRecord`, and `menuList`. Define the class `Program` with a `main` method and test the functionality of the utility class.

Program class:
```
package org.example.loancalc;
public class Program {
    public static void main(String[] args) {
        LoanAmortizationCalculatorUtil util = new LoanAmortizationCalculatorUtil();
        util.menuList();
    }
}
```

LoanAmortizationCalculator class:
```
package org.example.loancalc;
public class LoanAmortizationCalculator {

    private double principal;
    private double annualInterestRate;
    private int loanTerm; // in years

    // Constructor
    public LoanAmortizationCalculator(double principal, double annualInterestRate, int
loanTerm) {
        this.principal = principal;
        this.annualInterestRate = annualInterestRate;
        this.loanTerm = loanTerm;
    }

    // Getters and Setters
    public double getPrincipal() {
        return principal;
    }

    public void setPrincipal(double principal) {
        this.principal = principal;
    }

    public double getAnnualInterestRate() {
        return annualInterestRate;
    }

    public void setAnnualInterestRate(double annualInterestRate) {
        this.annualInterestRate = annualInterestRate;
    }

    public int getLoanTerm() {
        return loanTerm;
    }

    public void setLoanTerm(int loanTerm) {
        this.loanTerm = loanTerm;
    }

    // Method to calculate monthly payment
    public 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);
    }

    // Method to calculate total payment over the loan period
    public double calculateTotalPayment() {
        return calculateMonthlyPayment() * loanTerm * 12;
    }

    // toString method to display loan details
    @Override
    public String toString() {
        return "Principal: ₹" + principal + "\nAnnual Interest Rate: " + annualInterestRate +
"%\nLoan Term: " + loanTerm + " years";
    }
}
```

Util class:

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

public class LoanAmortizationCalculatorUtil {
    private LoanAmortizationCalculator calculator;

    // Method to accept user input
    public void acceptRecord() {
        Scanner scanner = new Scanner(System.in);

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

        System.out.print("Enter annual interest rate (%): ");
        double interestRate = scanner.nextDouble();

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

        calculator = new LoanAmortizationCalculator(principal, interestRate, loanTerm);
    }

    // Method to display the monthly payment and total payment
    public void printRecord() {
        System.out.println(calculator.toString());
        System.out.printf("Monthly Payment: ₹%.2f%n",
calculator.calculateMonthlyPayment());
        System.out.printf("Total Payment: ₹%.2f%n", calculator.calculateTotalPayment());
    }

    // Method to show menu options
    public void menuList() {
        Scanner scanner = new Scanner(System.in);
        int choice;
        do {
            System.out.println("\n1. Enter loan details");
            System.out.println("2. Display payment details");
            System.out.println("3. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    acceptRecord();
                    break;
                case 2:
                    if (calculator != null) {
                        printRecord();
                    } else {
                        System.out.println("No loan details available. Please enter loan details first.");
                    }
                    break;
                case 3:
                    System.out.println("Exiting...");
                    break;
                default:
                    System.out.println("Invalid choice, try again.");
            }
        } while (choice != 3);
    }
}
```



```
Console  Main.java  LoanAmortizationCalculator.java  LoanAmortizationCalculatorUtil.java
<terminated> Main (9) [Java Application] C:\Program Files\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v20240426-1530\jre\bin\javaw.exe

1. Enter loan details
2. Display payment details
3. Exit
Enter your choice: 1
Enter principal amount (₹): 2000000
Enter annual interest rate (%): 7.5
Enter loan term (in years): 3

1. Enter loan details
2. Display payment details
3. Exit
Enter your choice: 2
Principal: ₹2000000.0
Annual Interest Rate: 7.5%
Loan Term: 3 years
Monthly Payment: ₹62212.44
Total Payment: ₹2239647.71

1. Enter loan details
2. Display payment details
3. Exit
Enter your choice: 3
Exiting...
```

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 the class `CompoundInterestCalculator` with fields, an appropriate constructor, getter and setter methods, a `toString` method and business logic methods. Define the class `CompoundInterestCalculatorUtil` with methods `acceptRecord`, `printRecord`, and `menuList`. Define the class `Program` with a `main` method to test the functionality of the utility class.

Program class:

```
package org.example.CompoundInterestCalc;

import java.util.Scanner;

public class Program {

    public static void main(String[] args) {
        CompoundInterestCalculatorUtil util = new
CompoundInterestCalculatorUtil();
        Scanner scanner = new Scanner(System.in);
        int choice;

        do {
            util.menuList();
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    util.acceptRecord();
                    break;
                case 2:
                    util.printRecord();
                    break;
                case 3:
                    System.out.println("Exiting...");
                    break;
                default:
                    System.out.println("Invalid choice! Please try again.");
            }
        } while (choice != 3);

        scanner.close();
    }
}
```

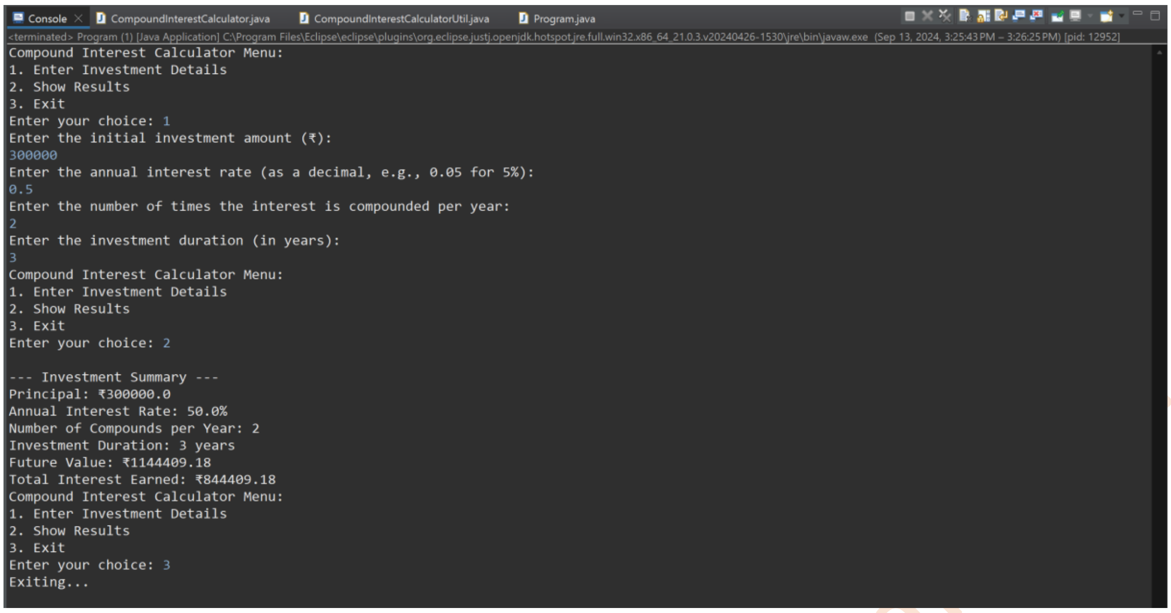
CompoundInterestCalculator class:

```
package org.example.CompoundInterestCalc;

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

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

    // Getter and Setter methods
    public double getPrincipal() {
        return principal;
    }

    public void setPrincipal(double principal) {
        this.principal = principal;
    }

    public double getAnnualInterestRate() {
        return annualInterestRate;
    }

    public void setAnnualInterestRate(double annualInterestRate) {
        this.annualInterestRate = annualInterestRate;
    }

    public int getNumberOfCompounds() {
        return numberOfCompounds;
    }

    public void setNumberOfCompounds(int numberOfCompounds) {
        this.numberOfCompounds = numberOfCompounds;
    }

    public int getYears() {
        return years;
    }
    public void setYears(int years) {
        this.years = years;
    }

    // Business logic method to calculate the future value
    public double calculateFutureValue() {
        return principal * Math.pow(1 + (annualInterestRate / numberOfCompounds),
numberOfCompounds * years);
    }
    public double calculateTotalInterest() {
        return calculateFutureValue() - principal;
    }

    @Override
    public String toString() {
        return "Principal: ₹" + principal +
                "\nAnnual Interest Rate: " + (annualInterestRate * 100) + "%" +
                "\nNumber of Compounds per Year: " + numberOfCompounds +
                "\nInvestment Duration: " + years + " years";
    }
}
```

CompoundInterestCalculatorUtil class:

```
package org.example.CompoundInterestCalc;

import java.util.Scanner;

public class CompoundInterestCalculatorUtil {

    private Scanner scanner = new Scanner(System.in);
    private CompoundInterestCalculator calculator;

    // Method to accept user input
    public void acceptRecord() {
        System.out.println("Enter the initial investment amount (₹): ");
        double principal = scanner.nextDouble();

        System.out.println("Enter the annual interest rate (as a decimal, e.g., 0.05 for
5%): ");
        double annualInterestRate = scanner.nextDouble();

        System.out.println("Enter the number of times the interest is compounded per
year: ");
```

```
        int numberOfCompounds = scanner.nextInt();

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

        calculator = new CompoundInterestCalculator(principal, annualInterestRate,
numberOfCompounds, years);
    }

    // Method to print the future value and interest earned
    public void printRecord() {
        double futureValue = calculator.calculateFutureValue();
        double totalInterest = calculator.calculateTotalInterest();

        System.out.println("\n--- Investment Summary ---");
        System.out.println(calculator);
        System.out.printf("Future Value: ₹%.2f\n", futureValue);
        System.out.printf("Total Interest Earned: ₹%.2f\n", totalInterest);
    }

    // Method to display a menu
    public void menuList() {
        System.out.println("Compound Interest Calculator Menu:");
        System.out.println("1. Enter Investment Details");
        System.out.println("2. Show Results");
        System.out.println("3. Exit");
    }
}
```


Console
CompoundInterestCalculator.java
CompoundInterestCalculatorUtil.java
Program.java

```
<terminated> Program (1) [Java Application] C:\Program Files\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v20240426-1530\jre\bin\javaw.exe (Sep 13, 2024, 3:25:43 PM – 3:26:25 PM) [pid: 12952]
Compound Interest Calculator Menu:
1. Enter Investment Details
2. Show Results
3. Exit
Enter your choice: 1
Enter the initial investment amount (₹):
300000
Enter the annual interest rate (as a decimal, e.g., 0.05 for 5%):
0.5
Enter the number of times the interest is compounded per year:
2
Enter the investment duration (in years):
3
Compound Interest Calculator Menu:
1. Enter Investment Details
2. Show Results
3. Exit
Enter your choice: 2

--- Investment Summary ---
Principal: ₹300000.0
Annual Interest Rate: 50.0%
Number of Compounds per Year: 2
Investment Duration: 3 years
Future Value: ₹1144409.18
Total Interest Earned: ₹844409.18
Compound Interest Calculator Menu:
1. Enter Investment Details
2. Show Results
3. Exit
Enter your choice: 3
Exiting...
```

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 the class `BMITracker` with fields, an appropriate constructor, getter and setter methods, a `toString` method, and business logic methods. Define the class `BMITrackerUtil` with methods `acceptRecord`, `printRecord`, and `menuList`. Define the class `Program` with a `main` method to test the functionality of the utility class.

Program class:

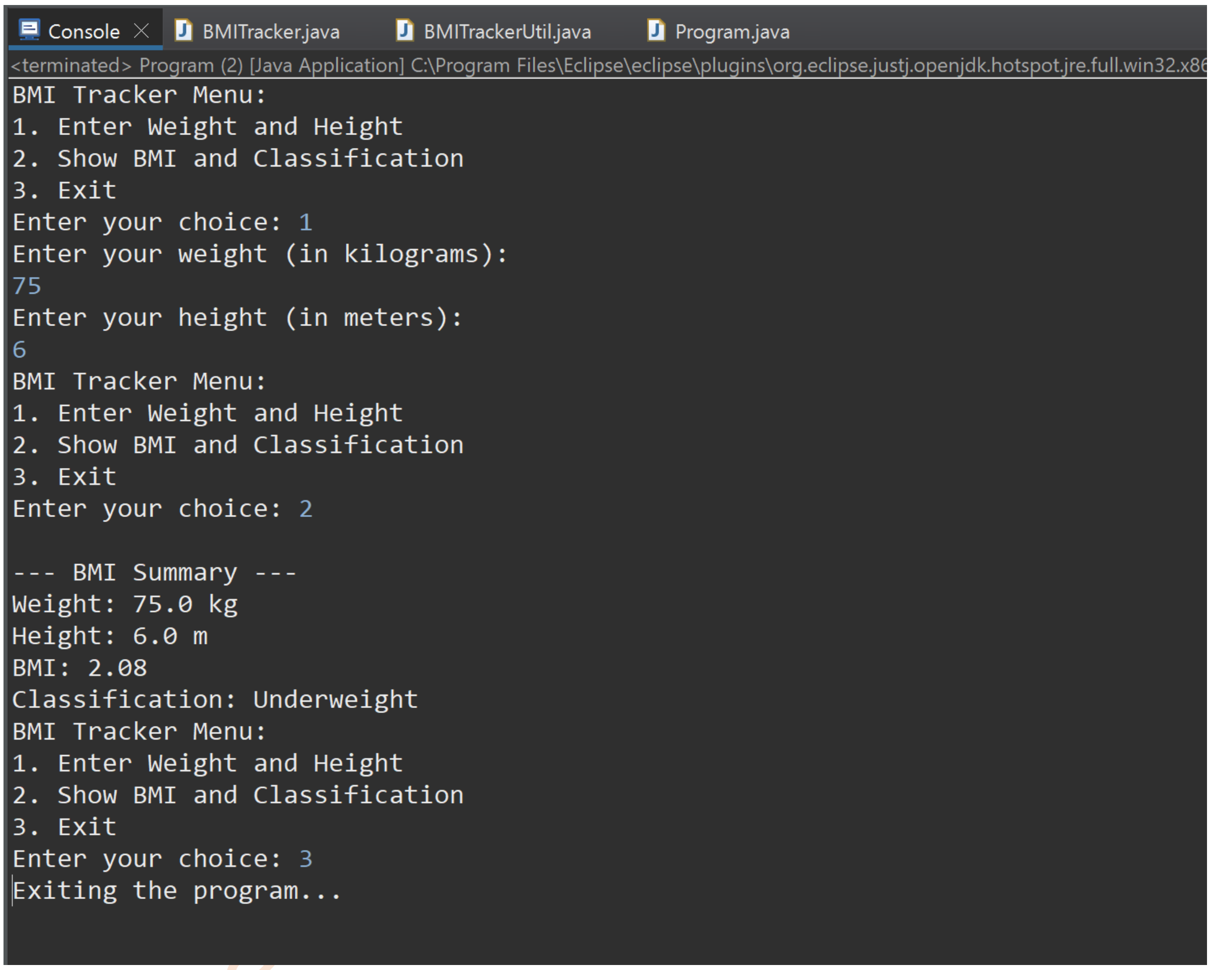
```
package org.example.bmi;
import java.util.Scanner;

public class Program {

    public static void main(String[] args) {
        BMITrackerUtil util = new BMITrackerUtil();
        Scanner scanner = new Scanner(System.in);
        int choice;

        do {
            util.menuList();
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    util.acceptRecord();
                    break;
                case 2:
                    util.printRecord();
                    break;
                case 3:
                    System.out.println("Exiting the program...");
                    break;
                default:
                    System.out.println("Invalid choice! Please try again.");
            }
        } while (choice != 3);

        scanner.close();
    }
}
```

BMITracker class:

```
package org.example.bmi;

public class BMITracker {
    private double weight;
    private double height;

    public BMITracker(double weight, double height) {
        this.weight = weight;
        this.height = height;
    }
    // Getter and Setter methods
    public double getWeight() {
        return weight;
    }

    public void setWeight(double weight) {
        this.weight = weight;
    }

    public double getHeight() {
        return height;
    }

    public void setHeight(double height) {
        this.height = height;
    }
    public double calculateBMI() {
        return weight / (height * height);
    }

    public String classifyBMI() {
        double bmi = calculateBMI();
        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";
        }
    }

    @Override
    public String toString() {
        return "Weight: " + weight + " kg\nHeight: " + height + " m";
    }
}
```

BMITrackerUtil class:

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

public class BMITrackerUtil {

    private Scanner scanner = new Scanner(System.in);
    private BMITracker tracker;
    // Method to accept user input
    public void acceptRecord() {
        System.out.println("Enter your weight (in kilograms): ");
        double weight = scanner.nextDouble();

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

        tracker = new BMITracker(weight, height);
    }
    // Method to print the BMI value and classification
    public void printRecord() {
        double bmi = tracker.calculateBMI();
        String classification = tracker.classifyBMI();

        System.out.println("\n--- BMI Summary ---");
        System.out.println(tracker);
        System.out.printf("BMI: %.2f\n", bmi);
        System.out.println("Classification: " + classification);
    }
    // Method to display a menu
    public void menuList() {
        System.out.println("BMI Tracker Menu:");
        System.out.println("1. Enter Weight and Height");
        System.out.println("2. Show BMI and Classification");
        System.out.println("3. Exit");
    }
}
```



```
<terminated> Program (2) [Java Application] C:\Program Files\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86
BMI Tracker Menu:
1. Enter Weight and Height
2. Show BMI and Classification
3. Exit
Enter your choice: 1
Enter your weight (in kilograms):
75
Enter your height (in meters):
6
BMI Tracker Menu:
1. Enter Weight and Height
2. Show BMI and Classification
3. Exit
Enter your choice: 2

--- BMI Summary ---
Weight: 75.0 kg
Height: 6.0 m
BMI: 2.08
Classification: Underweight
BMI Tracker Menu:
1. Enter Weight and Height
2. Show BMI and Classification
3. Exit
Enter your choice: 3
Exiting the program...
```

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 the class `DiscountCalculator` with fields, an appropriate constructor, getter and setter methods, a `toString` method, and business logic methods. Define the class `DiscountCalculatorUtil` with methods `acceptRecord`, `printRecord`, and `menuList`. Define the class `Program` with a `main` method to test the functionality of the utility class.
Code:

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

public class Program {

    public static void main(String[] args) {
        DiscountCalculatorUtil util = new DiscountCalculatorUtil();
        Scanner scanner = new Scanner(System.in);
        int choice;

        do {
            util.menuList();
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    util.acceptRecord();
                    break;
                case 2:
                    util.printRecord();
                    break;
                case 3:
                    System.out.println("Exiting the program...");
                    break;
                default:
                    System.out.println("Invalid choice! Please try again.");
            }
        } while (choice != 3);

        scanner.close();
    }
}
```

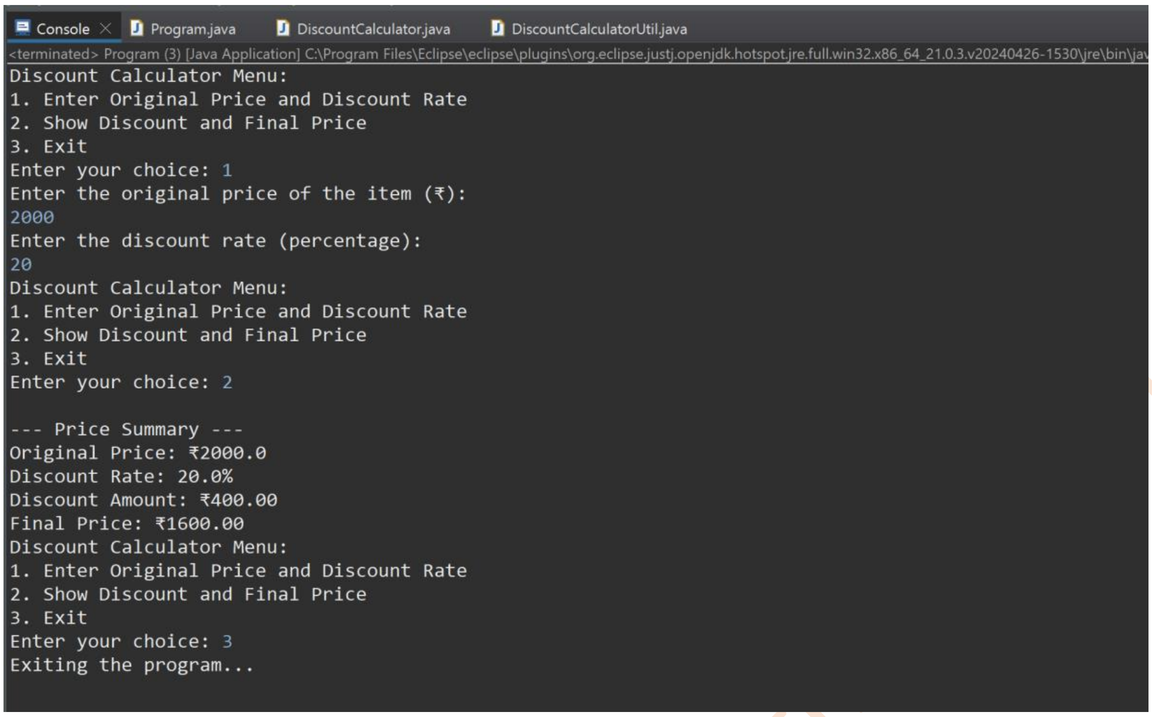
DiscountCalculator class:

```
package org.example.retails;

public class DiscountCalculator {
    private double originalPrice;
    private double discountRate;

    public DiscountCalculator(double originalPrice, double discountRate) {
        this.originalPrice = originalPrice;
        this.discountRate = discountRate;
    }

    // Getter and Setter methods
    public double getOriginalPrice() {
        return originalPrice;
    }

    public void setOriginalPrice(double originalPrice) {
        this.originalPrice = originalPrice;
    }

    public double getDiscountRate() {
        return discountRate;
    }

    public void setDiscountRate(double discountRate) {
        this.discountRate = discountRate;
    }

    public double calculateDiscountAmount() {
        return originalPrice * (discountRate / 100);
    }

    public double calculateFinalPrice() {
        return originalPrice - calculateDiscountAmount();
    }

    @Override
    public String toString() {
        return "Original Price: ₹" + originalPrice +
               "\nDiscount Rate: " + discountRate + "%";
    }
}
```

DiscountCalculatorUtilclass:

```
package org.example.retails;
```

```
import java.util.Scanner;

public class DiscountCalculatorUtil {

    private Scanner scanner = new Scanner(System.in);
    private DiscountCalculator calculator;

    // Method to accept user input
    public void acceptRecord() {
        System.out.println("Enter the original price of the item (₹): ");
        double originalPrice = scanner.nextDouble();

        System.out.println("Enter the discount rate (percentage): ");
        double discountRate = scanner.nextDouble();

        calculator = new DiscountCalculator(originalPrice, discountRate);
    }

    // Method to print the discount amount and final price
    public void printRecord() {
        double discountAmount = calculator.calculateDiscountAmount();
        double finalPrice = calculator.calculateFinalPrice();

        System.out.println("\n--- Price Summary ---");
        System.out.println(calculator);
        System.out.printf("Discount Amount: ₹%.2f\n", discountAmount);
        System.out.printf("Final Price: ₹%.2f\n", finalPrice);
    }

    // Method to display a menu
    public void menuList() {
        System.out.println("Discount Calculator Menu:");
        System.out.println("1. Enter Original Price and Discount Rate");
        System.out.println("2. Show Discount and Final Price");
        System.out.println("3. Exit");
    }
}
```



Console | Program.java | DiscountCalculator.java | DiscountCalculatorUtil.java

```
<terminated> Program (3) [Java Application] C:\Program Files\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v20240426-1530\jre\bin\jav
Discount Calculator Menu:
1. Enter Original Price and Discount Rate
2. Show Discount and Final Price
3. Exit
Enter your choice: 1
Enter the original price of the item (₹):
2000
Enter the discount rate (percentage):
20
Discount Calculator Menu:
1. Enter Original Price and Discount Rate
2. Show Discount and Final Price
3. Exit
Enter your choice: 2

--- Price Summary ---
Original Price: ₹2000.0
Discount Rate: 20.0%
Discount Amount: ₹400.00
Final Price: ₹1600.00
Discount Calculator Menu:
1. Enter Original Price and Discount Rate
2. Show Discount and Final Price
3. Exit
Enter your choice: 3
Exiting the program...
```

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 the class `TollBoothRevenueManager` with fields, an appropriate constructor, getter and setter methods, a `toString` method, and business logic methods. Define the class `TollBoothRevenueManagerUtil` with methods `acceptRecord`, `printRecord`, and `menuList`. Define the class `Program` with a `main` method to test the functionality of the utility class.

Class Program:
package org.example.tollboothmanagement;
import java.util.Scanner;

public class Program {

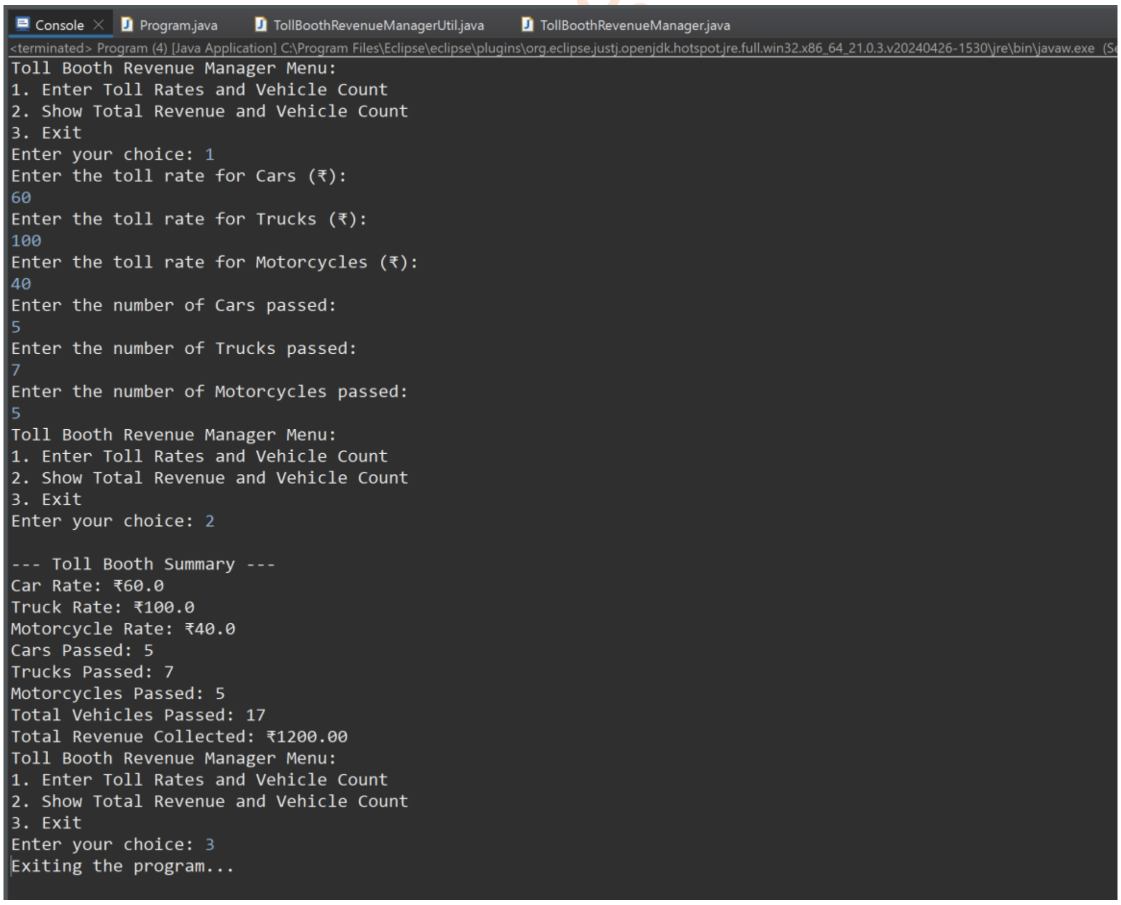
```
    public static void main(String[] args) {
        TollBoothRevenueManagerUtil util = new TollBoothRevenueManagerUtil();
        Scanner scanner = new Scanner(System.in);
        int choice;

        do {
            util.menuList();
            System.out.print("Enter your choice: ");
            choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    util.acceptRecord();
                    break;
                case 2:
                    util.printRecord();
                    break;
                case 3:
                    System.out.println("Exiting the program...");
                    break;
                default:
                    System.out.println("Invalid choice! Please try again.");
            }
        } while (choice != 3);

        scanner.close();
    }
}
```

TollBoothRevenueManagerUtil:

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

public class TollBoothRevenueManagerUtil {

    private Scanner scanner = new Scanner(System.in);
    private TollBoothRevenueManager manager;

    // Method to accept user input for toll rates and vehicle counts
    public void acceptRecord() {
        System.out.println("Enter the toll rate for Cars (₹): ");
        double carRate = scanner.nextDouble();

        System.out.println("Enter the toll rate for Trucks (₹): ");
        double truckRate = scanner.nextDouble();

        System.out.println("Enter the toll rate for Motorcycles (₹): ");
        double motorcycleRate = scanner.nextDouble();
        manager = new TollBoothRevenueManager(carRate, truckRate, motorcycleRate);

        System.out.println("Enter the number of Cars passed: ");
        int carCount = scanner.nextInt();
        manager.setCarCount(carCount);

        System.out.println("Enter the number of Trucks passed: ");
        int truckCount = scanner.nextInt();
        manager.setTruckCount(truckCount);

        System.out.println("Enter the number of Motorcycles passed: ");
        int motorcycleCount = scanner.nextInt();
        manager.setMotorcycleCount(motorcycleCount);
    }

    // Method to print the total revenue and vehicle count
    public void printRecord() {
        double totalRevenue = manager.calculateTotalRevenue();
        int totalVehicles = manager.calculateTotalVehicles();

        System.out.println("\n--- Toll Booth Summary ---");
        System.out.println(manager);
        System.out.printf("Total Vehicles Passed: %d\n", totalVehicles);
        System.out.printf("Total Revenue Collected: ₹%.2f\n", totalRevenue);
    }

    // Method to display a menu
    public void menuList() {
        System.out.println("Toll Booth Revenue Manager Menu:");
        System.out.println("1. Enter Toll Rates and Vehicle Count");
        System.out.println("2. Show Total Revenue and Vehicle Count");
        System.out.println("3. Exit");
    }
}
```

TollBoothRevenueManager:

```
package org.example.tollboothmanagement;

public class TollBoothRevenueManager {
    private double carRate;
    private double truckRate;
    private double motorcycleRate;

    private int carCount;
    private int truckCount;
    private int motorcycleCount;

    public TollBoothRevenueManager(double carRate, double truckRate, double
motorcycleRate) {
        this.carRate = carRate;
        this.truckRate = truckRate;
        this.motorcycleRate = motorcycleRate;
    }

    // Getter and Setter methods
    public double getCarRate() {
        return carRate;
    }

    public void setCarRate(double carRate) {
        this.carRate = carRate;
    }

    public double getTruckRate() {
        return truckRate;
    }

    public void setTruckRate(double truckRate) {
        this.truckRate = truckRate;
    }

    public double getMotorcycleRate() {
        return motorcycleRate;
    }

    public void setMotorcycleRate(double motorcycleRate) {
        this.motorcycleRate = motorcycleRate;
    }

    public int getCarCount() {
        return carCount;
    }

    public void setCarCount(int carCount) {
        this.carCount = carCount;
    }

    public int getTruckCount() {
        return truckCount;
    }

    public void setTruckCount(int truckCount) {
        this.truckCount = truckCount;
    }

    public int getMotorcycleCount() {
        return motorcycleCount;
    }

    public void setMotorcycleCount(int motorcycleCount) {
        this.motorcycleCount = motorcycleCount;
    }

    // Business logic to calculate total revenue
    public double calculateTotalRevenue() {
        return (carCount * carRate) + (truckCount * truckRate) + (motorcycleCount *
motorcycleRate);
    }

    // Business logic to calculate total vehicles
    public int calculateTotalVehicles() {
        return carCount + truckCount + motorcycleCount;
    }

    @Override
    public String toString() {
        return "Car Rate: ₹" + carRate +
                "\nTruck Rate: ₹" + truckRate +
                "\nMotorcycle Rate: ₹" + motorcycleRate +
                "\nCars Passed: " + carCount +
                "\nTrucks Passed: " + truckCount +
                "\nMotorcycles Passed: " + motorcycleCount;
    }
}
```


Console
Program.java
TollBoothRevenueManagerUtil.java
TollBoothRevenueManager.java

```
<terminated> Program (4) [Java Application] C:\Program Files\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v20240426-1530\jre\bin\javaw.exe (Se
Toll Booth Revenue Manager Menu:
1. Enter Toll Rates and Vehicle Count
2. Show Total Revenue and Vehicle Count
3. Exit
Enter your choice: 1
Enter the toll rate for Cars (₹):
60
Enter the toll rate for Trucks (₹):
100
Enter the toll rate for Motorcycles (₹):
40
Enter the number of Cars passed:
5
Enter the number of Trucks passed:
7
Enter the number of Motorcycles passed:
5
Toll Booth Revenue Manager Menu:
1. Enter Toll Rates and Vehicle Count
2. Show Total Revenue and Vehicle Count
3. Exit
Enter your choice: 2

--- Toll Booth Summary ---
Car Rate: ₹60.0
Truck Rate: ₹100.0
Motorcycle Rate: ₹40.0
Cars Passed: 5
Trucks Passed: 7
Motorcycles Passed: 5
Total Vehicles Passed: 17
Total Revenue Collected: ₹1200.00
Toll Booth Revenue Manager Menu:
1. Enter Toll Rates and Vehicle Count
2. Show Total Revenue and Vehicle Count
3. Exit
Enter your choice: 3
Exiting the program...
```