# OOPJ ASS4 Sol (SN)

Thursday, September 12, 2024 1:26 PM

**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 (₹).

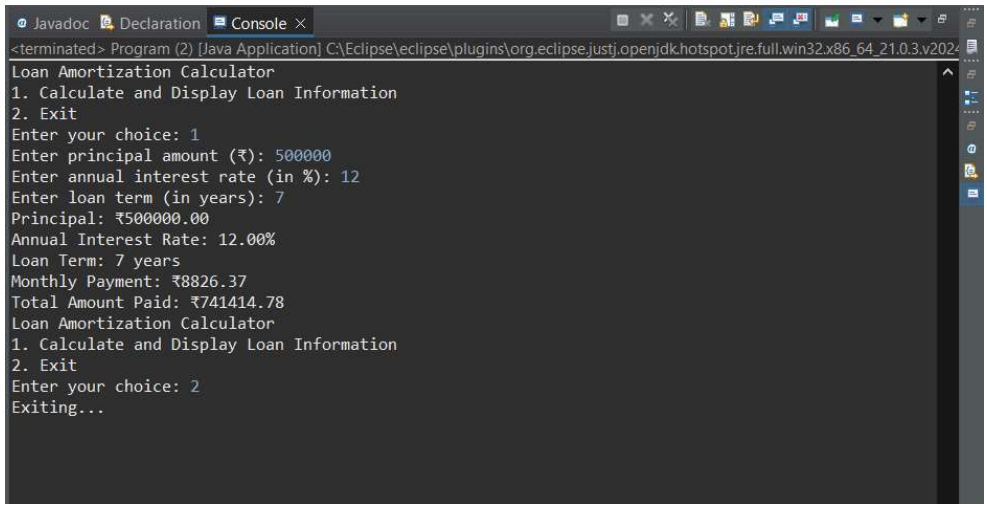
Define the class `LoanAmortizationCalculator` with fields, an appropriate constructor, getter and setter methods, a `toString` method and business logic methods. Define the class `LoanAmortizationCalculatorUtil` with methods `acceptRecord`, `printRecord`, and `menuList`. Define the class `Program` with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

```java
package org.example.LoanAmortizationCalculator;

public class LoanAmortizationCalculator {
    // Fields
    private double principal;
    private double annualInterestRate;
    private int loanTerm;

    // 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 amount paid
    public double calculateTotalAmountPaid() {
        return calculateMonthlyPayment() * loanTerm * 12;
    }

    // toString method
    @Override
    public String toString() {
        return String.format("Principal: ₹%.2f\nAnnual Interest Rate: %.2f%%\nLoan Term: %d years\nMonthly Payment: ₹%.2f\nTotal Amount Paid: ₹%.2f",
                principal, annualInterestRate, loanTerm, calculateMonthlyPayment(), calculateTotalAmountPaid());
    }
}
```

```java
package org.example.LoanAmortizationCalculatorUtil;
import org.example.LoanAmortizationCalculator.LoanAmortizationCalculator;
import java.util.Scanner;
```

```
public class LoanAmortizationCalculatorUtil {

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

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

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

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

        return new LoanAmortizationCalculator(principal, annualInterestRate, loanTerm);
    }

    // Method to print record
    public static void printRecord(LoanAmortizationCalculator calculator) {
        System.out.println(calculator);
    }

    // Method to display menu
    public static void menuList() {
        System.out.println("Loan Amortization Calculator");
        System.out.println("1. Calculate and Display Loan Information");
        System.out.println("2. Exit");
    }
}
```

```
package org.example.Program;
import java.util.Scanner;
import org.example.LoanAmortizationCalculatorUtil.LoanAmortizationCalculatorUtil;
import org.example.LoanAmortizationCalculator.LoanAmortizationCalculator;

public class Program {

    public static void main(String[] args) {


        Scanner scanner = new Scanner(System.in);
        int choice;
        LoanAmortizationCalculator calculator = null;

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

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




**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.

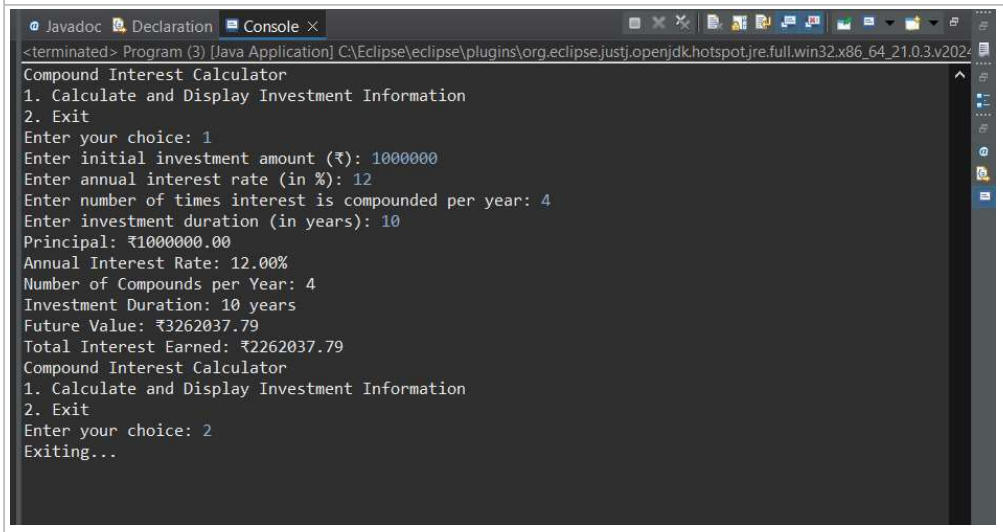
```
package org.example.domain;

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;
          }


          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;
          }
```

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


        public double calculateTotalInterest() {
            return calculateFutureValue() - principal;
        }


        @Override
        public String toString() {
            return String.format("Principal: ₹%.2f\nAnnual Interest Rate: %.2f%%\nNumber of Compounds per Year: %d\nInvestment Duration: %d years
\nFuture Value: ₹%.2f\nTotal Interest Earned: ₹%.2f",
                    principal, annualInterestRate, numberOfCompounds, years, calculateFutureValue(), calculateTotalInterest());
        }
    }
```

```
package org.example.util;
import java.util.Scanner;
import org.example.domain.*;

public class CompoundInterestCalculatorUtil {


    public static void menuList() {
        System.out.println("Compound Interest Calculator");
        System.out.println("1. Calculate and Display Investment Information");
        System.out.println("2. Exit");
    }


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

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

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

        System.out.print("Enter number of times interest is compounded per year: ");
        int numberOfCompounds = sc.nextInt();

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

        return new CompoundInterestCalculator(principal, annualInterestRate, numberOfCompounds, years);
    }


    public static void printRecord(CompoundInterestCalculator calculator) {
        System.out.println(calculator);
    }
}
```

```
package org.example.main;

import java.util.Scanner;
import org.example.domain.*;
import org.example.util.*;

public class Program {

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

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

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

```
        } finally {
            scanner.close();
        }
    }
}
```




### 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.

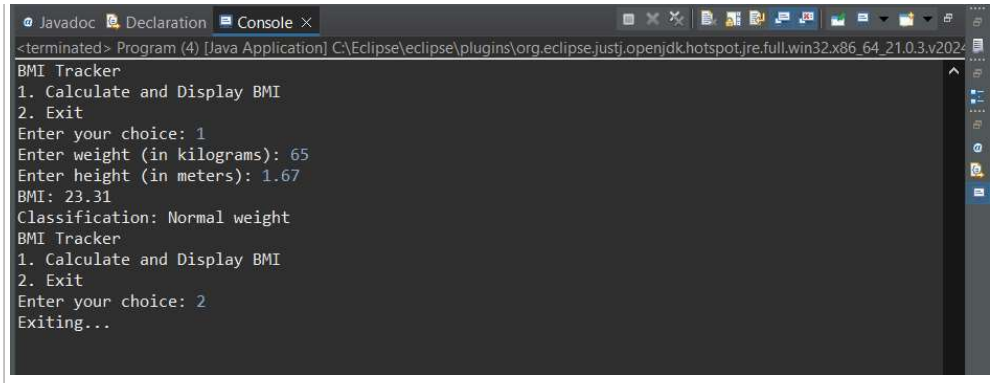
Define the class `BMITracker` with fields, an appropriate constructor, getter and setter methods, a `toString` method, and business logic methods. Define the class `BMITrackerUtil` with methods `acceptRecord`, `printRecord`, and `menuList`. Define the class `Program` with a `main` method to test the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

```
package org.example.domain;

public class BMITracker {

          private double weight;
          private double height;

          // Constructor
          public BMITracker(double weight, double height) {
              this.weight = weight;
              this.height = height;
          }

          // Getters and Setters
          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;
          }

          // Method to calculate BMI
          public double calculateBMI() {
              return weight / (height * height);
          }

          // Method to classify BMI
          public String classifyBMI() {
              double bmi = calculateBMI();
              if (bmi < 18.5) {
                  return "Underweight";
```

```
            } else if (bmi < 24.9) {
                return "Normal weight";
            } else if (bmi < 29.9) {
                return "Overweight";
            } else {
                return "Obese";
            }
        }

        // toString method to display BMI and classification
        @Override
        public String toString() {
            double bmi = calculateBMI();
            return String.format("BMI: %.2f\nClassification: %s", bmi, classifyBMI());
        }
    }
```

```
package org.example.util;
import java.util.Scanner;
import org.example.domain.*;

public class BMITrackerUtil {

    // Method to display the menu
    public static void menuList() {
        System.out.println("BMI Tracker");
        System.out.println("1. Calculate and Display BMI");
        System.out.println("2. Exit");
    }

    // Method to accept user input and create a BMITracker object
    public static BMITracker acceptRecord() {
        Scanner sc_= new Scanner(System.in);

        System.out.print("Enter weight (in kilograms): ");
        double weight = sc.nextDouble();

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

        return new BMITracker(weight, height);
    }

    // Method to print the BMI record
    public static void printRecord(BMITracker tracker) {
        System.out.println(tracker);
    }
}
```

```
package org.example.main;
import java.util.Scanner;
import org.example.domain.BMITracker;
import org.example.util.BMITrackerUtil;

public class Program {

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

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

                switch (choice) {
                    case 1:
                        tracker = BMITrackerUtil.acceptRecord();
                        BMITrackerUtil.printRecord(tracker);
                        break;
                    case 2:
                        System.out.println("Exiting...");
                        break;
                    default:
                        System.out.println("Invalid choice. Please try again.");
                }
            } while (choice != 2);
        } finally {
            scanner.close();
        }
    }
}
```




### 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.

```
package org.example.domain;

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

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

    // Getters and Setters
    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;
    }

    // Method to calculate the discount amount
    public double calculateDiscountAmount() {
        return originalPrice * (discountRate / 100);
    }

    // Method to calculate the final price after discount
    public double calculateFinalPrice() {
        return originalPrice - calculateDiscountAmount();
    }

    // toString method to display the discount amount and final price
    @Override
    public String toString() {
        return String.format("Original Price: ₹%.2f\nDiscount Rate: %.2f%%\nDiscount Amount: ₹%.2f\nFinal Price: ₹%.2f",
                originalPrice, discountRate, calculateDiscountAmount(), calculateFinalPrice());
    }
}
```

```
package org.example.util;
import org.example.domain.*;

import java.util.Scanner;

public class DiscountCalculatorUtil {

    // Method to display the menu
    public static void menuList() {
        System.out.println("Discount Calculation for Retail Sales");
```

```
        System.out.println("1. Calculate and Display Discount Information");
        System.out.println("2. Exit");
    }

    // Method to accept user input and create a DiscountCalculator object
    public static DiscountCalculator acceptRecord() {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter original price of the item (₹): ");
        double originalPrice = sc.nextDouble();

        System.out.print("Enter discount percentage: ");
        double discountRate = sc.nextDouble();

        return new DiscountCalculator(originalPrice, discountRate);
    }

    // Method to print the discount record
    public static void printRecord(DiscountCalculator calculator) {
        System.out.println(calculator);
    }
}
```

```
package org.example.main;

import java.util.Scanner;
import org.example.domain.*;
import org.example.util.*;

public class Program {

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

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

                switch (choice) {
                    case 1:
                        calculator = DiscountCalculatorUtil.acceptRecord();
                        DiscountCalculatorUtil.printRecord(calculator);
                        break;
                    case 2:
                        System.out.println("Exiting...");
                        break;
                    default:
                        System.out.println("Invalid choice. Please try again.");
                }
            } while (choice != 2);
        } finally {
            scanner.close();
        }
    }
}
```

```
Javadoc  Declaration  Console ×
<terminated> Program (5) [Java Application] C:\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v202
Discount Calculation for Retail Sales
1. Calculate and Display Discount Information
2. Exit
Enter your choice: 1
Enter original price of the item (₹): 1000
Enter discount percentage: 14
Original Price: ₹1000.00
Discount Rate: 14.00%
Discount Amount: ₹140.00
Final Price: ₹860.00
Discount Calculation for Retail Sales
1. Calculate and Display Discount Information
2. Exit
Enter your choice: 2
Exiting...
```

### 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.

```
package org.example.domain;

public class TollBoothRevenueManager {
    private double carRate;
    private double truckRate;
    private double motorcycleRate;
    private int numCars;
    private int numTrucks;
    private int numMotorcycles;

    // Constructor
    public TollBoothRevenueManager(double carRate, double truckRate, double motorcycleRate, int numCars, int numTrucks, int numMotorcycles) {
        this.carRate = carRate;
        this.truckRate = truckRate;
        this.motorcycleRate = motorcycleRate;
        this.numCars = numCars;
        this.numTrucks = numTrucks;
        this.numMotorcycles = numMotorcycles;
    }

    // Getters and Setters
    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 getNumCars() {
        return numCars;
    }

    public void setNumCars(int numCars) {
        this.numCars = numCars;
    }

    public int getNumTrucks() {
        return numTrucks;
    }

    public void setNumTrucks(int numTrucks) {
        this.numTrucks = numTrucks;
    }

    public int getNumMotorcycles() {
        return numMotorcycles;
    }

    public void setNumMotorcycles(int numMotorcycles) {
        this.numMotorcycles = numMotorcycles;
    }

    // Method to calculate the total revenue
    public double calculateTotalRevenue() {
        return (carRate * numCars) + (truckRate * numTrucks) + (motorcycleRate * numMotorcycles);
    }

    // Method to calculate the total number of vehicles
    public int calculateTotalVehicles() {
        return numCars + numTrucks + numMotorcycles;
    }

    // toString method to display the toll booth information
    @Override
    public String toString() {
        return String.format("Car Rate: ₹%.2f\nTruck Rate: ₹%.2f\nMotorcycle Rate: ₹%.2f\nNumber of Cars: %d\nNumber of Trucks: %d\nNumber of Motorcycles:
```

```
%d\nTotal Vehicles: %d\nTotal Revenue: ₹%.2f",
                carRate, truckRate, motorcycleRate, numCars, numTrucks, numMotorcycles, calculateTotalVehicles(), calculateTotalRevenue());
    }
}
```

```
package org.example.util;
import org.example.domain.*;

import java.util.Scanner;

public class TollBoothRevenueManagerUtil {

    // Method to display the menu
    public static void menuList() {
        System.out.println("Toll Booth Revenue Management");
        System.out.println("1. Set Toll Rates and Vehicle Counts");
        System.out.println("2. Display Toll Booth Information");
        System.out.println("3. Exit");
    }

    // Method to accept user input and create a TollBoothRevenueManager object
    public static TollBoothRevenueManager acceptRecord() {
        Scanner sc = new Scanner(System.in);

        // Setting default rates
        double defaultCarRate = 50.00;
        double defaultTruckRate = 100.00;
        double defaultMotorcycleRate = 30.00;

        System.out.println("Enter toll rates (in ₹):");
        System.out.print("Car Rate (default ₹50.00): ");
        double carRate = sc.nextDouble();
        carRate = (carRate <= 0) ? defaultCarRate : carRate;

        System.out.print("Truck Rate (default ₹100.00): ");
        double truckRate = sc.nextDouble();
        truckRate = (truckRate <= 0) ? defaultTruckRate : truckRate;

        System.out.print("Motorcycle Rate (default ₹30.00): ");
        double motorcycleRate = sc.nextDouble();
        motorcycleRate = (motorcycleRate <= 0) ? defaultMotorcycleRate : motorcycleRate;

        System.out.print("Enter number of cars: ");
        int numCars = sc.nextInt();

        System.out.print("Enter number of trucks: ");
        int numTrucks = sc.nextInt();

        System.out.print("Enter number of motorcycles: ");
        int numMotorcycles = sc.nextInt();

        return new TollBoothRevenueManager(carRate, truckRate, motorcycleRate, numCars, numTrucks, numMotorcycles);
    }

    // Method to print the toll booth record
    public static void printRecord(TollBoothRevenueManager manager) {
        System.out.println(manager);
    }
}
```

```
package org.example.main;
import java.util.Scanner;
import org.example.domain.*;
import org.example.util.*;

public class Program {

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

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

                switch (choice) {
                    case 1:
                        manager = TollBoothRevenueManagerUtil.acceptRecord();
                        break;
                    case 2:
                        if (manager != null) {
                            TollBoothRevenueManagerUtil.printRecord(manager);
                        } else {
                            System.out.println("No toll booth data available. Please enter the data first.");
                        }
                        break;
                    case 3:
```

```
                    System.out.println("Exiting...");
                    break;
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        } while (choice != 3);
    } finally {
        scanner.close();
    }
  }
}
```