**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.

**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.

Ans: **1. CompoundInterestCalculator Class**

**package** main;

**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() {

**double** ratePerPeriod = annualInterestRate / 100 / numberOfCompounds;

**double** futureValue = principal \* Math.*pow*(1 + ratePerPeriod, numberOfCompounds \* years);

**return** futureValue;

}

**public** **double** calculateTotalInterest() {

**return** calculateFutureValue() - principal;

}

@Override

**public** String toString() {

**return** "CompoundInterestCalculator{" +

"principal=" + principal +

", annualInterestRate=" + annualInterestRate +

", numberOfCompounds=" + numberOfCompounds +

", years=" + years +

'}';

}

}

**2. CompoundInterestCalculatorUtil Class**

**package** main;

**import** java.util.Scanner;

**class** CompoundInterestCalculatorUtil {

**private** CompoundInterestCalculator calculator;

**public** **void** acceptRecord() {

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.print("Enter the initial investment amount (₹): ");

**double** principal = scanner.nextDouble();

System.***out***.print("Enter the annual interest rate (in %): ");

**double** annualInterestRate = scanner.nextDouble();

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

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

System.***out***.print("Enter the investment duration (in years): ");

**int** years = scanner.nextInt();

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

}

**public** **void** printRecord() {

**if** (calculator != **null**) {

**double** futureValue = calculator.calculateFutureValue();

**double** totalInterest = calculator.calculateTotalInterest();

System.***out***.printf("Future Value: ₹%.2f%n", futureValue);

System.***out***.printf("Total Interest Earned: ₹%.2f%n", totalInterest);

} **else** {

System.***out***.println("No investment record available. Please enter the investment details first.");

}

}

**public** **void** menuList() {

Scanner scanner = **new** Scanner(System.***in***);

**int** choice;

**do** {

System.***out***.println("\n--- Compound Interest Calculator Menu ---");

System.***out***.println("1. Enter Investment Details");

System.***out***.println("2. Display Future Value and Total Interest Earned");

System.***out***.println("0. Exit");

System.***out***.print("Enter your choice: ");

choice = scanner.nextInt();

**switch** (choice) {

**case** 1:

acceptRecord();

**break**;

**case** 2:

printRecord();

**break**;

**case** 0:

System.***out***.println("Exiting...");

**break**;

**default**:

System.***out***.println("Invalid choice. Please try again.");

}

} **while** (choice != 0);

}

}

**3. Program Class**

**package** main;

**public** **class** Program {

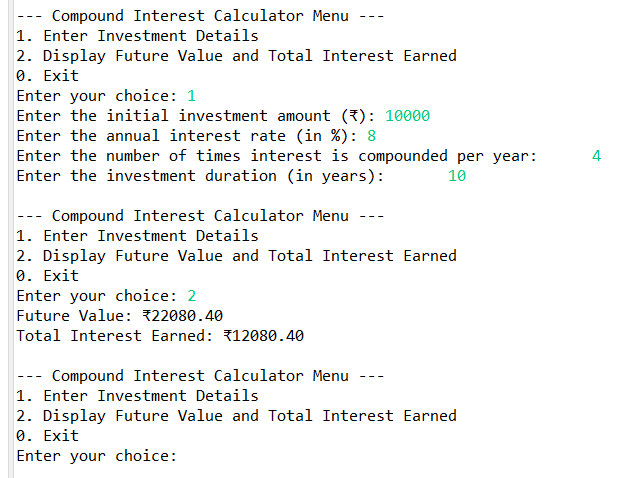
**public** **static** **void** main(String[] args) {

CompoundInterestCalculatorUtil util = **new** CompoundInterestCalculatorUtil();

util.menuList();

}

}



**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.

**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.

**1. LoanAmortizationCalculator Class**

**package** main;

**public** **class** LoanAmortizationCalculator {

**private** **double** principal;

**private** **double** annualInterestRate;

**private** **int** loanTerm;

**public** LoanAmortizationCalculator(**double** principal, **double** annualInterestRate, **int** loanTerm) {

**this**.principal = principal;

**this**.annualInterestRate = annualInterestRate;

**this**.loanTerm = loanTerm;

}

**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;

}

**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);

}

**public** **double** calculateTotalPayment() {

**return** calculateMonthlyPayment() \* loanTerm \* 12;

}

@Override

**public** String toString() {

**return** "LoanAmortizationCalculator{" +

"principal=" + principal +

", annualInterestRate=" + annualInterestRate +

", loanTerm=" + loanTerm +

'}';

}

}

**2. LoanAmortizationCalculatorUtil Class**

**package** main;

**import** java.util.Scanner;

**public** **class** LoanAmortizationCalculatorUtil {

**private** LoanAmortizationCalculator calculator;

**public** **void** acceptRecord() {

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.print("Enter the principal amount (loan amount in ₹): ");

**double** principal = scanner.nextDouble();

System.***out***.print("Enter the annual interest rate (in %): ");

**double** annualInterestRate = scanner.nextDouble();

System.***out***.print("Enter the loan term (in years): ");

**int** loanTerm = scanner.nextInt();

calculator = **new** LoanAmortizationCalculator(principal, annualInterestRate, loanTerm);

}

**public** **void** printRecord() {

**if** (calculator != **null**) {

**double** monthlyPayment = calculator.calculateMonthlyPayment();

**double** totalPayment = calculator.calculateTotalPayment();

System.***out***.printf("Monthly Payment: ₹%.2f%n", monthlyPayment);

System.***out***.printf("Total Amount Paid over the Life of the Loan: ₹%.2f%n", totalPayment);

} **else** {

System.***out***.println("No loan record available. Please enter the loan details first.");

}

}

**public** **void** menuList() {

Scanner scanner = **new** Scanner(System.***in***);

**int** choice;

**do** {

System.***out***.println("\n--- Loan Amortization Calculator Menu ---");

System.***out***.println("1. Enter Loan Details");

System.***out***.println("2. Display Monthly Payment and Total Payment");

System.***out***.println("0. Exit");

System.***out***.print("Enter your choice: ");

choice = scanner.nextInt();

**switch** (choice) {

**case** 1:

acceptRecord();

**break**;

**case** 2:

printRecord();

**break**;

**case** 0:

System.***out***.println("Exiting...");

**break**;

**default**:

System.***out***.println("Invalid choice. Please try again.");

}

} **while** (choice != 0);

}

}

**3. Program Class**

**package** main;

**public** **class** Program {

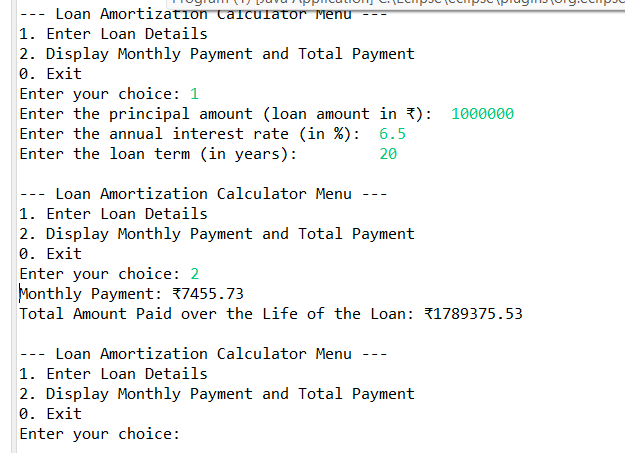
**public** **static** **void** main(String[] args) {

LoanAmortizationCalculatorUtil util = **new** LoanAmortizationCalculatorUtil();

util.menuList();

}

}



**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.

**1. BMITracker Class**

**package main;**

**public class BMITracker {**

**private double weight;**

**private double height;**

**public BMITracker(double weight, double height) {**

**this.weight = weight;**

**this.height = height;**

**}**

**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 "BMITracker{" +**

**"weight=" + weight +**

**", height=" + height +**

**'}';**

**}**

**}**

**2. BMITrackerUtil Class**

**package** main;

**import** java.util.Scanner;

**public** **class** BMITrackerUtil {

**private** BMITracker bmiTracker;

**public** **void** acceptRecord() {

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.print("Enter your weight (in kilograms): ");

**double** weight = scanner.nextDouble();

System.***out***.print("Enter your height (in meters): ");

**double** height = scanner.nextDouble();

bmiTracker = **new** BMITracker(weight, height);

}

**public** **void** printRecord() {

**if** (bmiTracker != **null**) {

**double** bmi = bmiTracker.calculateBMI();

String classification = bmiTracker.classifyBMI();

System.***out***.printf("Your BMI: %.2f%n", bmi);

System.***out***.println("Classification: " + classification);

} **else** {

System.***out***.println("No BMI record available. Please enter your weight and height first.");

}

}

**public** **void** menuList() {

Scanner scanner = **new** Scanner(System.***in***);

**int** choice;

**do** {

System.***out***.println("\n--- BMI Tracker Menu ---");

System.***out***.println("1. Enter Weight and Height");

System.***out***.println("2. Display BMI and Classification");

System.***out***.println("0. Exit");

System.***out***.print("Enter your choice: ");

choice = scanner.nextInt();

**switch** (choice) {

**case** 1:

acceptRecord();

**break**;

**case** 2:

printRecord();

**break**;

**case** 0:

System.***out***.println("Exiting...");

**break**;

**default**:

System.***out***.println("Invalid choice. Please try again.");

}

} **while** (choice != 0);

}

}

**3. Program Class**

**package** main;

**public** **class** Program {

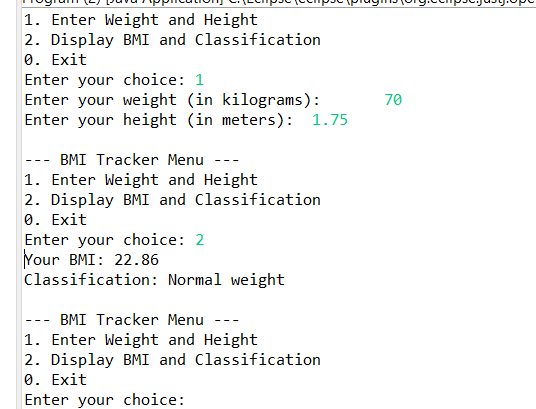
**public** **static** **void** main(String[] args) {

BMITrackerUtil util = **new** BMITrackerUtil();

util.menuList();

}

}



**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.

1. **DiscountCalculator Class**

**package main;**

**public class DiscountCalculator {**

**private double originalPrice;**

**private double discountRate;**

**public DiscountCalculator(double originalPrice, double discountRate) {**

**this.originalPrice = originalPrice;**

**this.discountRate = discountRate;**

**}**

**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 "DiscountCalculator{" +**

**"originalPrice=" + originalPrice +**

**", discountRate=" + discountRate +**

**'}';**

**}**

**}**

1. **DiscountCalculatorUtil Class**

**package main;**

**import java.util.Scanner;**

**public class DiscountCalculatorUtil {**

**private DiscountCalculator discountCalculator;**

**public void acceptRecord() {**

**Scanner scanner = new Scanner(System.*in*);**

**System.*out*.print("Enter the original price (in Indian Rupees): ");**

**double originalPrice = scanner.nextDouble();**

**System.*out*.print("Enter the discount percentage: ");**

**double discountRate = scanner.nextDouble();**

**discountCalculator = new DiscountCalculator(originalPrice, discountRate);**

**}**

**public void printRecord() {**

**if (discountCalculator != null) {**

**double discountAmount = discountCalculator.calculateDiscountAmount();**

**double finalPrice = discountCalculator.calculateFinalPrice();**

**System.*out*.printf("Discount Amount: ₹%.2f%n", discountAmount);**

**System.*out*.printf("Final Price: ₹%.2f%n", finalPrice);**

**} else {**

**System.*out*.println("No record available. Please enter the original price and discount rate first.");**

**}**

**}**

**public void menuList() {**

**Scanner scanner = new Scanner(System.*in*);**

**int choice;**

**do {**

**System.*out*.println("\n--- Discount Calculator Menu ---");**

**System.*out*.println("1. Enter Original Price and Discount Rate");**

**System.*out*.println("2. Display Discount Amount and Final Price");**

**System.*out*.println("0. Exit");**

**System.*out*.print("Enter your choice: ");**

**choice = scanner.nextInt();**

**switch (choice) {**

**case 1:**

**acceptRecord();**

**break;**

**case 2:**

**printRecord();**

**break;**

**case 0:**

**System.*out*.println("Exiting...");**

**break;**

**default:**

**System.*out*.println("Invalid choice. Please try again.");**

**}**

**} while (choice != 0);**

**}**

**}**

1. **Program Class**

**package main;**

**public class Program {**

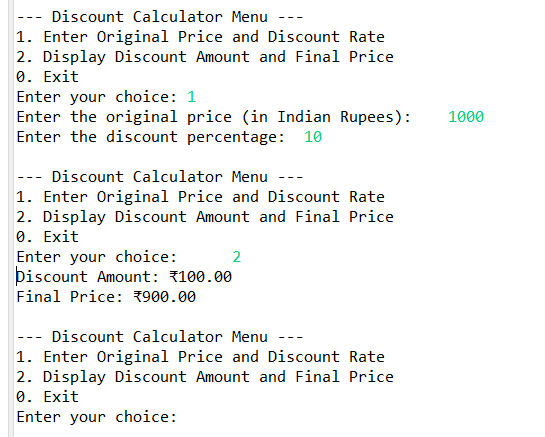
**public static void main(String[] args) {**

**DiscountCalculatorUtil util = new DiscountCalculatorUtil();**

**util.menuList();**

**}**

**}**



**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.

1. **TollBoothRevenueManager Class**

**package main;**

**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;**

**}**

**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;**

**}**

**public double calculateTotalRevenue() {**

**return (carCount \* carRate) + (truckCount \* truckRate) + (motorcycleCount \* motorcycleRate);**

**}**

**public int calculateTotalVehicles() {**

**return carCount + truckCount + motorcycleCount;**

**}**

**@Override**

**public String toString() {**

**return "TollBoothRevenueManager{" +**

**"carRate=" + carRate +**

**", truckRate=" + truckRate +**

**", motorcycleRate=" + motorcycleRate +**

**", carCount=" + carCount +**

**", truckCount=" + truckCount +**

**", motorcycleCount=" + motorcycleCount +**

**'}';**

**}**

**}**

1. **TollBoothrevenueManagerUtil Class**

**package main;**

**import java.util.Scanner;**

**public class TollBoothrevenueManagerUtil {**

**private TollBoothRevenueManager manager;**

**public void acceptRecord() {**

**Scanner scanner = new Scanner(System.*in*);**

**System.*out*.print("Enter toll rate for Car: ₹");**

**double carRate = scanner.nextDouble();**

**System.*out*.print("Enter toll rate for Truck: ₹");**

**double truckRate = scanner.nextDouble();**

**System.*out*.print("Enter toll rate for Motorcycle: ₹");**

**double motorcycleRate = scanner.nextDouble();**

**manager = new TollBoothRevenueManager(carRate, truckRate, motorcycleRate);**

**System.*out*.print("Enter number of Cars: ");**

**int carCount = scanner.nextInt();**

**manager.setCarCount(carCount);**

**System.*out*.print("Enter number of Trucks: ");**

**int truckCount = scanner.nextInt();**

**manager.setTruckCount(truckCount);**

**System.*out*.print("Enter number of Motorcycles: ");**

**int motorcycleCount = scanner.nextInt();**

**manager.setMotorcycleCount(motorcycleCount);**

**}**

**public void printRecord() {**

**if (manager != null) {**

**int totalVehicles = manager.calculateTotalVehicles();**

**double totalRevenue = manager.calculateTotalRevenue();**

**System.*out*.printf("Total Vehicles: %d%n", totalVehicles);**

**System.*out*.printf("Total Revenue: ₹%.2f%n", totalRevenue);**

**} else {**

**System.*out*.println("No record available. Please enter toll rates and vehicle counts first.");**

**}**

**}**

**public void menuList() {**

**Scanner scanner = new Scanner(System.*in*);**

**int choice;**

**do {**

**System.*out*.println("\n--- Toll Booth Revenue Manager Menu ---");**

**System.*out*.println("1. Set Toll Rates and Vehicle Counts");**

**System.*out*.println("2. Display Total Vehicles and Total Revenue");**

**System.*out*.println("0. Exit");**

**System.*out*.print("Enter your choice: ");**

**choice = scanner.nextInt();**

**switch (choice) {**

**case 1:**

**acceptRecord();**

**break;**

**case 2:**

**printRecord();**

**break;**

**case 0:**

**System.*out*.println("Exiting...");**

**break;**

**default:**

**System.*out*.println("Invalid choice. Please try again.");**

**}**

**} while (choice != 0);**

**}**

**}**

1. **Program Class**

**package main;**

**public class Program {**

**public static void main(String[] args) {**

**TollBoothrevenueManagerUtil util = new TollBoothrevenueManagerUtil();**

**util.menuList();**

**}**

**}**

