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.
Sol:

Program: there are 3 class files LoanAmortizationCalculator, class LoanAmortizationCalculatorUtil, Program

1 LoanAmortizationCalculator :

```
package in.rahul.SR;

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



   public LoanAmortizationCalculator() {

   }

   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;
    }

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

    @Override
    public String toString() {
        return String.format("Loan Details:\nPrincipal: ₹%.2f\nAnnual Interest Rate:
%.2f%%\nLoan Term: %d years",
                principal, annualInterestRate, loanTerm);
    }
}
```

2. LoanAmortizationCalculatorUtil:

```
package in.rahul.SR;

import java.util.Scanner;

 public class LoanAmortizationCalculatorUtil {
    private LoanAmortizationCalculator loanAmortizationCalculator;
    Scanner scanner = new Scanner(System.in);
```

```
    // Method to accept loan details from the user
    public void acceptRecord() {
        System.out.print("Enter Loan Principal Amount (in ₹): ");
        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();

        loanAmortizationCalculator = new LoanAmortizationCalculator(principal,
annualInterestRate, loanTerm);
    }

    // Method to display loan details and calculated payments
    public void printRecord() {
            System.out.println(loanAmortizationCalculator);  // Display loan details
            double monthlyPayment = loanAmortizationCalculator.calculateMonthlyPayment();
            double totalPayment = loanAmortizationCalculator.calculateTotalPayment();
            System.out.printf("Monthly Payment: ₹%.2f\n", monthlyPayment);
            System.out.printf("Total Payment Over %d Years: ₹%.2f\n",
loanAmortizationCalculator.getLoanTerm(), totalPayment);

    }

    // Method to display the menu options
    public void menuList() {
        System.out.println("1. Enter Loan Details");
        System.out.println("2. Display Loan Amortization Details");
        System.out.println("3. Exit");
    }
}
```

3. program main method:

```
package in.rahul.SR;

import java.util.Scanner;

public class program {
    public static void main(String[] args) {
        LoanAmortizationCalculatorUtil util = new LoanAmortizationCalculatorUtil();
        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();   // Accept loan details from user
                    break;
                case 2:
                    util.printRecord();    // Display calculated loan details
                    break;
                case 3:
                    System.out.println("Exiting...");
                    break;
```

```
            default:
                System.out.println("Invalid choice! Please select a valid option.");
            }
        } while (choice != 3);  // Repeat menu until user selects "Exit"

        scanner.close();
    }
}
```

Output:

```
1. Enter Loan Details
2. Display Loan Amortization Details
3. Exit
Enter your choice: 1
Enter Loan Principal Amount (in ₹): 1000
Enter Annual Interest Rate (in %): 12
Enter Loan Term (in years): 2
1. Enter Loan Details
2. Display Loan Amortization Details
3. Exit
Enter your choice: 2
Loan Details:
Principal: ₹1000.00
Annual Interest Rate: 12.00%
Loan Term: 2 years
Monthly Payment: ₹47.07
Total Payment Over 2 Years: ₹1129.76
1. Enter Loan Details
2. Display Loan Amortization 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.

Sol:

Program: there are 3 class files CompoundInterestCalculator, CompoundInterestCalculatorUtil, Program.

1. CompoundInterestCalculator

```
package com.Sid.in;

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

    // Default Constructor
```

```
    public CompoundInterestCalculator() {}

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

    // 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 getNumberOfCompounds() {
        return numberOfCompounds;
    }

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

    public int getYears() {
        return years;
    }

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

    // Method to calculate future value
    public double calculateFutureValue() {
        return principal * Math.pow(1 + (annualInterestRate / numberOfCompounds),
numberOfCompounds * years);
    }

    // Method to calculate total interest earned
    public double calculateTotalInterest() {
        return calculateFutureValue() - principal;
    }

    @Override
    public String toString() {
        return String.format("Investment Details:\nPrincipal: ₹%.2f\nAnnual Interest Rate:
%.2f%%\n" +
```

```
                    "Compounds per Year: %d\nInvestment Duration: %d years",
                    principal, annualInterestRate, numberOfCompounds, years);
        }
    }
```

2. CompoundInterestCalculatorUtil

```
package com.Sid.in;

import java.util.Scanner;

class CompoundInterestCalculatorUtil {
        Scanner scanner = new Scanner(System.in);
  private CompoundInterestCalculator compoundInterestCalculator;

  public void acceptRecord() {
    System.out.print("Enter Initial Investment Amount (in ₹): ");
    double principal = scanner.nextDouble();
    System.out.print("Enter Annual Interest Rate (in %): ");
    double annualInterestRate = scanner.nextDouble();
    System.out.print("Enter Number of Compounds per Year: ");
    int numberOfCompounds = scanner.nextInt();
    System.out.print("Enter Investment Duration (in years): ");
    int years = scanner.nextInt();

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

  public void printRecord() {

      System.out.println(compoundInterestCalculator);  // Display investment details
      double futureValue = compoundInterestCalculator.calculateFutureValue();
      double totalInterest = compoundInterestCalculator.calculateTotalInterest();
      System.out.printf("Future Value: ₹%.2f\n", futureValue);
      System.out.printf("Total Interest Earned: ₹%.2f\n", totalInterest);
  }

  // Method to display the menu options
  public void menuList() {
    System.out.println("1. Enter Investment Details");
    System.out.println("2. Display Future Value and Total Interest");
    System.out.println("3. Exit");
  }
}
```

3.program contain main method:

```
package com.Sid.in;

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();   // Accept investment details from user
                    break;
                case 2:
                    util.printRecord();    // Display future value and total interest
                    break;
                case 3:
                    System.out.println("Exiting...");
                    break;
                default:
                    System.out.println("Invalid choice! Please select a valid option.");
            }

        } while (choice != 3);  // Repeat menu until user selects "Exit"
        scanner.close();
    }
}
```

Output:

```
1. Enter Investment Details
2. Display Future Value and Total Interest
3. Exit
Enter your choice: 1
Enter Initial Investment Amount (in ₹): 1000
Enter Annual Interest Rate (in %): 10
Enter Number of Compounds per Year: 3
Enter Investment Duration (in years): 1
1. Enter Investment Details
2. Display Future Value and Total Interest
3. Exit
Enter your choice: 2
Investment Details:
Principal: ₹1000.00
Annual Interest Rate: 10.00%
Compounds per Year: 3
Investment Duration: 1 years
Future Value: ₹81370.37
Total Interest Earned: ₹80370.37
1. Enter Investment Details
2. Display Future Value and Total Interest
3. Exit
```

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.
Sol:

1.BMITracker:

```
package com.in.IR;

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

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

    public double getWeight() {
        return weight;
    }

    public void setWeight(double weight) {
        this.weight = weight;
        calculateBMI();  // Recalculate BMI when weight is updated
```

```
    }

    public double getHeight() {
        return height;
    }

    public void setHeight(double height) {
        this.height = height;
        calculateBMI();  // Recalculate BMI when height is updated
    }

    public double getBMI() {
        return bmi;
    }

    private void calculateBMI() {
        if (height > 0) {
            this.bmi = weight / (height * height);
        }
    }

    public String classifyBMI() {
        if (bmi < 18.5) {
            return "Underweight";
        } else if (bmi < 24.9) {
            return "Normal weight";
        } else if (bmi < 29.9) {
            return "Overweight";
        } else {
            return "Obese";
        }
    }

    @Override
    public String toString() {
        return String.format("BMI: %.2f\n%s", bmi, classifyBMI());
    }
}
```

2. BMITrackerUtil:

```
package com.in.IR;

import java.util.Scanner;

public class BMITrackerUtil {
    private Scanner sc = new Scanner(System.in);

    public BMITracker acceptRecord() {
        System.out.print("Enter weight : ");
        double weight = sc.nextDouble();
        System.out.print("Enter height : ");
        double height = sc.nextDouble();
        return new BMITracker(weight, height);
    }

    public void printRecord(BMITracker tracker) {
        System.out.println(tracker);
    }
```

```
    // Method to display the menu
    public void menuList() {
        System.out.println("1. Calculate BMI");
        System.out.println("2. Display Last BMI Record");
        System.out.println("3. Exit");
    }
}
```

3. Program class contain main method;

```
package com.in.IR;
import java.util.Scanner;
public class Program {

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

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

            switch (choice) {
                case 1:
                    tracker = util.acceptRecord();  // Accept new record
                    util.printRecord(tracker);      // Display the calculated BMI
                    break;
                case 2:

                        util.printRecord(tracker);  // Display the last BMI record
                    break;
                case 3:
                    System.out.println("Exiting...");
                    break;
                default:
                    System.out.println("Invalid choice, please try again.");
            }
        } while (choice != 3);
        sc.close();
    }
}
```

```
Program (2) [Java Application] C:\Users\
1. Calculate BMI
2. Display Last BMI Record
3. Exit
Enter your choice: 1
Enter weight : 75
Enter height : 1.67
BMI: 26.89
Overweight
```

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.

Sol:

1. **DiscountCalculator:**

```
package in.java.BR;

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

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


   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 getDiscountAmount() {
      return discountAmount;
```

```
    }

    public double getFinalPrice() {
        return finalPrice;
    }

    //logic to calculate discount and final price
    private void calculateDiscount() {
        discountAmount = originalPrice * (discountRate / 100);
        finalPrice = originalPrice - discountAmount;
    }

    @Override
    public String toString() {
        return String.format("Original Price: ₹%.2f\nDiscount Rate: %.2f%%\nDiscount
Amount: ₹%.2f\nFinal Price: ₹%.2f",
                originalPrice, discountRate, discountAmount, finalPrice);
    }
}
```

2. **DiscountCalculatorUtil:**

```
package in.java.BR;
import java.util.Scanner;

public class DiscountCalculatorUtil {

    private static DiscountCalculator lastRecord;

    public static void acceptRecord(Scanner scanner) {
        System.out.print("Enter original price: ₹");
        double originalPrice = scanner.nextDouble();
        System.out.print("Enter discount percentage: ");
        double discountRate = scanner.nextDouble();

        lastRecord = new DiscountCalculator(originalPrice, discountRate);  // Create new
record

        System.out.println("Discount calculation completed!");
    }

    public static void printRecord() {
        System.out.println(lastRecord);  // Use toString() of DiscountCalculator
    }
    // Method to display menu options
    public static void menuList() {
        System.out.println("Discount Calculator Menu:");
        System.out.println("1. Calculate Discount");
        System.out.println("2. Display Last Discount");
        System.out.println("3. Exit");
    }
}
```

3. **class Program: with main method:**

```
package in.java.BR;

import java.util.Scanner;
```

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

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

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

      scanner.close();
   }
}
```

Output:

```
Discount Calculator Menu:
1. Calculate Discount
2. Display Last Discount
3. Exit
Enter your choice: 1
Enter original price: ₹1000
Enter discount percentage: 30
Discount calculation completed!
Discount Calculator Menu:
1. Calculate Discount
2. Display Last Discount
3. Exit
Enter your choice: 2
Original Price: ₹1000.00
Discount Rate: 30.00%
Discount Amount: ₹300.00
Final Price: ₹700.00
Discount Calculator Menu:
1. Calculate Discount
2. Display Last Discount
3. Exit
Enter your choice: 3
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.

Sol:

1. **TollBoothRevenueManager:**

```
package com.in.DhamDhum;

public class TollBoothRevenueManager {
   private double carRate;
   private double truckRate;
   private double motorcycleRate;
   private int carCount;
   private int truckCount;
   private int motorcycleCount;

   // Constructor
   public TollBoothRevenueManager(double carRate, double truckRate, double
motorcycleRate) {
      this.carRate = carRate;
      this.truckRate = truckRate;
      this.motorcycleRate = motorcycleRate;
      this.carCount = 0;
      this.truckCount = 0;
      this.motorcycleCount = 0;
   }

   // 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 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;
    }

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

    // Updated toString method to display details on new lines
    @Override
    public String toString() {
        return "TollBoothRevenueManager Details:\n" +
                "Car Rate: ₹" + carRate + "\n" +
                "Truck Rate: ₹" + truckRate + "\n" +
                "Motorcycle Rate: ₹" + motorcycleRate + "\n" +
                "Number of Cars: " + carCount + "\n" +
                "Number of Trucks: " + truckCount + "\n" +
                "Number of Motorcycles: " + motorcycleCount + "\n" +
                "Total Revenue: ₹" + calculateTotalRevenue();
    }
}
```

2. TollBoothRevenueManagerUtil

```
package com.in.DhamDhum;

import java.util.Scanner;

public class TollBoothRevenueManagerUtil {

   private static Scanner scanner = new Scanner(System.in); // Single Scanner instance

   public static TollBoothRevenueManager acceptRecord() {
      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();

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

      System.out.print("Enter number of Cars: ");
      manager.setCarCount(scanner.nextInt());

      System.out.print("Enter number of Trucks: ");
      manager.setTruckCount(scanner.nextInt());

      System.out.print("Enter number of Motorcycles: ");
      manager.setMotorcycleCount(scanner.nextInt());

      return manager;
   }

   public static void printRecord(TollBoothRevenueManager manager) {
      System.out.println(manager.toString());
   }

   public static void menuList() {
      System.out.println("Toll Booth Revenue Management System");
      System.out.println("1. Accept Toll Rates and Vehicle Counts");
      System.out.println("2. Display Toll Booth Details");
      System.out.println("3. Exit");
   }
}
```

3. Program: contain main method:

```
package com.in.DhamDhum;

import java.util.Scanner;

public class Program {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      TollBoothRevenueManager manager = null;
      while (true) {
         TollBoothRevenueManagerUtil.menuList();
         System.out.print("Enter your choice: ");
```

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

            switch (choice) {

               case 1:
                  manager = TollBoothRevenueManagerUtil.acceptRecord();
                  break;
               case 2:
                  if (manager != null) {
                     TollBoothRevenueManagerUtil.printRecord(manager);
                  } else {
                     System.out.println("Please enter toll rates and vehicle counts first.");
                  }
                  break;
               case 3:
                  System.out.println("Exiting...");
                  scanner.close();
                  return;
               default:
                  System.out.println("Invalid choice. Please try again.");
            }
         }

      }
   }
```

Output:

```
Toll Booth Revenue Management System
1. Accept Toll Rates and Vehicle Counts
2. Display Toll Booth Details
3. Exit
Enter your choice: 1
Enter toll rate for Car : 50
Enter toll rate for Truck : 100
Enter toll rate for Motorcycle : 30
Enter number of Cars: 20
Enter number of Trucks: 10
Enter number of Motorcycles: 30
Toll Booth Revenue Management System
1. Accept Toll Rates and Vehicle Counts
2. Display Toll Booth Details
3. Exit
Enter your choice: 2
TollBoothRevenueManager Details:
Car Rate: ₹50.0
Truck Rate: ₹100.0
Motorcycle Rate: ₹30.0
Number of Cars: 20
Number of Trucks: 10
Number of Motorcycles: 30
Total Revenue: ₹2900.0
Toll Booth Revenue Management System
1. Accept Toll Rates and Vehicle Counts
2. Display Toll Booth Details
3. Exit
Enter your choice: 3
Exiting...
```