**Assignment No 4**

**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 code:**

**package Cal\_Loan;**

**import java.util.Scanner;**

**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;**

**}**

**// 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 getLoanTerm() {**

**return loanTerm;**

**}**

**public void setLoanTerm(int loanTerm) {**

**this.loanTerm = loanTerm;**

**}**

**// Business Logic: Calculate Monthly Payment**

**public double calculateMonthlyPayment() {**

**double monthlyInterestRate = (annualInterestRate / 12) / 100;**

**int numberOfMonths = loanTerm \* 12;**

**double monthlyPayment = principal \* (monthlyInterestRate \* Math.*pow*(1 + monthlyInterestRate, numberOfMonths))**

**/ (Math.*pow*(1 + monthlyInterestRate, numberOfMonths) - 1);**

**return monthlyPayment;**

**}**

**// Business Logic: Calculate Total Payment over the life of the loan**

**public double calculateTotalPayment() {**

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

**}**

**// Overriding the toString method to display loan information**

**@Override**

**public String toString() {**

**return "Loan Details:\n" +**

**"Principal: ₹" + principal + "\n" +**

**"Annual Interest Rate: " + annualInterestRate + "%\n" +**

**"Loan Term: " + loanTerm + " years\n" +**

**"Monthly Payment: ₹" + String.*format*("%.2f", calculateMonthlyPayment()) + "\n" +**

**"Total Payment: ₹" + String.*format*("%.2f", calculateTotalPayment());**

**}**

**}**

**class LoanAmortizationCalculatorUtil {**

**private LoanAmortizationCalculator loanCalculator;**

**// Method to accept user input**

**public void acceptRecord() {**

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

**System.*out*.print("Enter Principal Amount (₹): ");**

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

**System.*out*.print("Enter Annual Interest Rate (%): ");**

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

**System.*out*.print("Enter Loan Term (in years): ");**

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

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

**}**

**// Method to print loan details**

**public void printRecord() {**

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

**System.*out*.println(loanCalculator); // Calls the overridden toString() method**

**} else {**

**System.*out*.println("No loan details found.");**

**}**

**}**

**// Method to display menu**

**public void menuList() {**

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

**int choice;**

**do {**

**System.*out*.println("\nLoan Amortization Calculator Menu:");**

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

**System.*out*.println("2. Display Loan Details");**

**System.*out*.println("3. Exit");**

**System.*out*.print("Choose an option: ");**

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

**switch (choice) {**

**case 1:**

**acceptRecord();**

**break;**

**case 2:**

**printRecord();**

**break;**

**case 3:**

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

**break;**

**default:**

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

**}**

**} while (choice != 3);**

**}**

**}**

**public class Loan\_Cal{**

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

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

**util.menuList(); // Start the menu and accept input from the user**

**}**

**}**

**Output:**

**Loan Amortization Calculator Menu:**

**1. Enter Loan Details**

**2. Display Loan Details**

**3. Exit**

**Choose an option: 1**

**Enter Principal Amount (₹): 500000**

**Enter Annual Interest Rate (%): 7.5**

**Enter Loan Term (in years): 20**

**Loan Amortization Calculator Menu:**

**1. Enter Loan Details**

**2. Display Loan Details**

**3. Exit**

**Choose an option: 2**

**Loan Details:**

**Principal: ₹500000.0**

**Annual Interest Rate: 7.5%**

**Loan Term: 20 years**

**Monthly Payment: ₹4027.97**

**Total Payment: ₹966711.83**

**Loan Amortization Calculator Menu:**

**1. Enter Loan Details**

**2. Display Loan Details**

**3. Exit**

**Choose an option:**

**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 code:**

**package Cal\_Loan;**

**import java.util.Scanner;**

**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 futureValue = principal \* Math.*pow*((1 + annualInterestRate / numberOfCompounds / 100), numberOfCompounds \* years);**

**return futureValue;**

**}**

**public double calculateTotalInterest() {**

**return calculateFutureValue() - principal;**

**}**

**@Override**

**public String toString() {**

**return "Investment Details:\n" +**

**"Principal: ₹" + principal + "\n" +**

**"Annual Interest Rate: " + annualInterestRate + "%\n" +**

**"Compounded " + numberOfCompounds + " times per year\n" +**

**"Duration: " + years + " years\n" +**

**"Future Value: ₹" + String.*format*("%.2f", calculateFutureValue()) + "\n" +**

**"Total Interest Earned: ₹" + String.*format*("%.2f", calculateTotalInterest());**

**}**

**}**

**class CompoundInterestCalculatorUtil {**

**private CompoundInterestCalculator calculator;**

**public void acceptRecord() {**

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

**System.*out*.print("Enter Initial Investment (₹): ");**

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

**System.*out*.print("Enter Annual Interest Rate (%): ");**

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

**System.*out*.print("Enter Number of Times Interest Compounded per Year: ");**

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

**System.*out*.print("Enter Investment Duration (in years): ");**

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

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

**}**

**public void printRecord() {**

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

**System.*out*.println(calculator); // Calls the overridden toString() method**

**} else {**

**System.*out*.println("No investment details found.");**

**}**

**}**

**public void menuList() {**

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

**int choice;**

**do {**

**System.*out*.println("\nCompound Interest Calculator Menu:");**

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

**System.*out*.println("2. Display Investment Details");**

**System.*out*.println("3. Exit");**

**System.*out*.print("Choose an option: ");**

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

**switch (choice) {**

**case 1:**

**acceptRecord();**

**break;**

**case 2:**

**printRecord();**

**break;**

**case 3:**

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

**break;**

**default:**

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

**}**

**} while (choice != 3);**

**}**

**}**

**public class Compound\_Interest {**

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

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

**util.menuList(); // Start the menu and accept input from the user**

**}**

**}**

**Output:**

**Compound Interest Calculator Menu:**

**1. Enter Investment Details**

**2. Display Investment Details**

**3. Exit**

**Choose an option: 1**

**Enter Initial Investment (₹): 100000**

**Enter Annual Interest Rate (%): 6.5**

**Enter Number of Times Interest Compounded per Year: 4**

**Enter Investment Duration (in years): 10**

**Compound Interest Calculator Menu:**

**1. Enter Investment Details**

**2. Display Investment Details**

**3. Exit**

**Choose an option: 2**

**Investment Details:**

**Principal: ₹100000.0**

**Annual Interest Rate: 6.5%**

**Compounded 4 times per year**

**Duration: 10 years**

**Future Value: ₹190555.88**

**Total Interest Earned: ₹90555.88**

**Compound Interest Calculator Menu:**

**1. Enter Investment Details**

**2. Display Investment Details**

**3. Exit**

**Choose an option:**

**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 code:**

**package Cal\_Loan;**

**import java.util.Scanner;**

**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 "BMI Details:\n" +**

**"Weight: " + weight + " kg\n" +**

**"Height: " + height + " m\n" +**

**"BMI: " + String.*format*("%.2f", calculateBMI()) + "\n" +**

**"Classification: " + classifyBMI();**

**}**

**}**

**class BMITrackerUtil {**

**private BMITracker bmiTracker;**

**public void acceptRecord() {**

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

**System.*out*.print("Enter Weight (in kg): ");**

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

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

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

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

**}**

**public void printRecord() {**

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

**System.*out*.println(bmiTracker); // Calls the overridden toString() method**

**} else {**

**System.*out*.println("No BMI details found.");**

**}**

**}**

**public void menuList() {**

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

**int choice;**

**do {**

**System.*out*.println("\nBMI Tracker Menu:");**

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

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

**System.*out*.println("3. Exit");**

**System.*out*.print("Choose an option: ");**

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

**switch (choice) {**

**case 1:**

**acceptRecord();**

**break;**

**case 2:**

**printRecord();**

**break;**

**case 3:**

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

**break;**

**default:**

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

**}**

**} while (choice != 3);**

**}**

**}**

**public class BMI {**

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

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

**util.menuList();**

**}**

**}**

**Output:**

**BMI Tracker Menu:**

**1. Enter BMI Details**

**2. Display BMI Details**

**3. Exit**

**Choose an option: 1**

**Enter Weight (in kg): 74**

**Enter Height (in meters): 6.1**

**BMI Tracker Menu:**

**1. Enter BMI Details**

**2. Display BMI Details**

**3. Exit**

**Choose an option: 2**

**BMI Details:**

**Weight: 74.0 kg**

**Height: 6.1 m**

**BMI: 1.99**

**Classification: Underweight**

**BMI Tracker Menu:**

**1. Enter BMI Details**

**2. Display BMI Details**

**3. Exit**

**Choose an option:**

**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.**

**Program code:**

**package Cal\_Loan;**

**import java.util.Scanner;**

**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 "Discount Details:\n" +**

**"Original Price: ₹" + originalPrice + "\n" +**

**"Discount Rate: " + discountRate + "%\n" +**

**"Discount Amount: ₹" + String.*format*("%.2f", calculateDiscountAmount()) + "\n" +**

**"Final Price: ₹" + String.*format*("%.2f", calculateFinalPrice());**

**}**

**}**

**class DiscountCalculatorUtil {**

**private DiscountCalculator discountCalculator;**

**public void acceptRecord() {**

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

**System.*out*.print("Enter Original Price (in ₹): ");**

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

**System.*out*.print("Enter Discount Rate (in %): ");**

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

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

**}**

**public void printRecord() {**

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

**System.*out*.println(discountCalculator); // Calls the overridden toString() method**

**} else {**

**System.*out*.println("No discount details found.");**

**}**

**}**

**public void menuList() {**

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

**int choice;**

**do {**

**System.*out*.println("\nDiscount Calculator Menu:");**

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

**System.*out*.println("2. Display Discount Details");**

**System.*out*.println("3. Exit");**

**System.*out*.print("Choose an option: ");**

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

**switch (choice) {**

**case 1:**

**acceptRecord();**

**break;**

**case 2:**

**printRecord();**

**break;**

**case 3:**

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

**break;**

**default:**

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

**}**

**} while (choice != 3);**

**}**

**}**

**public class Retail\_Sale {**

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

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

**util.menuList(); // Start the menu and accept input from the user**

**}**

**}**

**Output:**

**Discount Calculator Menu:**

**1. Enter Discount Details**

**2. Display Discount Details**

**3. Exit**

**Choose an option: 1**

**Enter Original Price (in ₹): 100**

**Enter Discount Rate (in %): 20**

**Discount Calculator Menu:**

**1. Enter Discount Details**

**2. Display Discount Details**

**3. Exit**

**Choose an option: 2**

**Discount Details:**

**Original Price: ₹100.0**

**Discount Rate: 20.0%**

**Discount Amount: ₹20.00**

**Final Price: ₹80.00**

**Discount Calculator Menu:**

**1. Enter Discount Details**

**2. Display Discount Details**

**3. Exit**

**Choose an option:**

**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.**

**Program code:**

**package Cal\_Loan;**

**import java.util.Scanner;**

**class TollBoothRevenueManager {**

**private double carRate;**

**private double truckRate;**

**private double motorcycleRate;**

**private int numberOfCars;**

**private int numberOfTrucks;**

**private int numberOfMotorcycles;**

**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 getNumberOfCars() {**

**return numberOfCars;**

**}**

**public void setNumberOfCars(int numberOfCars) {**

**this.numberOfCars = numberOfCars;**

**}**

**public int getNumberOfTrucks() {**

**return numberOfTrucks;**

**}**

**public void setNumberOfTrucks(int numberOfTrucks) {**

**this.numberOfTrucks = numberOfTrucks;**

**}**

**public int getNumberOfMotorcycles() {**

**return numberOfMotorcycles;**

**}**

**public void setNumberOfMotorcycles(int numberOfMotorcycles) {**

**this.numberOfMotorcycles = numberOfMotorcycles;**

**}**

**public double calculateTotalRevenue() {**

**return (numberOfCars \* carRate) + (numberOfTrucks \* truckRate) + (numberOfMotorcycles \* motorcycleRate);**

**}**

**public int calculateTotalVehicles() {**

**return numberOfCars + numberOfTrucks + numberOfMotorcycles;**

**}**

**@Override**

**public String toString() {**

**return "Toll Booth Revenue Details:\n" +**

**"Number of Cars: " + numberOfCars + "\n" +**

**"Number of Trucks: " + numberOfTrucks + "\n" +**

**"Number of Motorcycles: " + numberOfMotorcycles + "\n" +**

**"Total Vehicles: " + calculateTotalVehicles() + "\n" +**

**"Total Revenue: ₹" + String.*format*("%.2f", calculateTotalRevenue());**

**}**

**}**

**class TollBoothRevenueManagerUtil {**

**private TollBoothRevenueManager tollManager;**

**public void acceptRecord() {**

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

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

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

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

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

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

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

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

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

**tollManager.setNumberOfCars(sc.nextInt());**

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

**tollManager.setNumberOfTrucks(sc.nextInt());**

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

**tollManager.setNumberOfMotorcycles(sc.nextInt());**

**}**

**public void printRecord() {**

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

**System.*out*.println(tollManager);**

**} else {**

**System.*out*.println("No toll booth data available.");**

**}**

**}**

**public void menuList() {**

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

**int choice;**

**do {**

**System.*out*.println("\nToll Booth Revenue Menu:");**

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

**System.*out*.println("2. Display Revenue and Vehicle Count");**

**System.*out*.println("3. Exit");**

**System.*out*.print("Choose an option: ");**

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

**switch (choice) {**

**case 1:**

**acceptRecord();**

**break;**

**case 2:**

**printRecord();**

**break;**

**case 3:**

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

**break;**

**default:**

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

**}**

**} while (choice != 3);**

**}**

**}**

**public class Toll\_Managment {**

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

**TollBoothRevenueManagerUtil util = new TollBoothRevenueManagerUtil();**

**util.menuList();**

**}**

**}**

**Output:**

**Toll Booth Revenue Menu:**

**1. Enter Toll Rates and Vehicle Counts**

**2. Display Revenue and Vehicle Count**

**3. Exit**

**Choose an option: 1**

**Enter toll rate for Cars (₹): 50**

**Enter toll rate for Trucks (₹): 100**

**Enter toll rate for Motorcycles (₹): 30**

**Enter the number of Cars: 10**

**Enter the number of Trucks: 5**

**Enter the number of Motorcycles: 20**

**Toll Booth Revenue Menu:**

**1. Enter Toll Rates and Vehicle Counts**

**2. Display Revenue and Vehicle Count**

**3. Exit**

**Choose an option: 2**

**Toll Booth Revenue Details:**

**Number of Cars: 10**

**Number of Trucks: 5**

**Number of Motorcycles: 20**

**Total Vehicles: 35**

**Total Revenue: ₹1600.00**

**Toll Booth Revenue Menu:**

**1. Enter Toll Rates and Vehicle Counts**

**2. Display Revenue and Vehicle Count**

**3. Exit**

**Choose an option:**