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.

package org.example.loan;

class LoanAmortizationCalculator {

private double principal; // Loan amount

private double annualInterestRate; // Annual interest rate in percentage

private int loanTermYears; // Loan term in years

// Constructor

public LoanAmortizationCalculator(double principal, double annualInterestRate, int loanTermYears) {

this.principal = principal;

this.annualInterestRate = annualInterestRate;

this.loanTermYears = loanTermYears;

}

// 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 getLoanTermYears() {

return loanTermYears;

}

public void setLoanTermYears(int loanTermYears) {

this.loanTermYears = loanTermYears;

}

// Business logic to calculate monthly payment

public double calculateMonthlyPayment() {

double monthlyInterestRate = annualInterestRate / 12 / 100; // Convert annual rate to monthly

int numberOfMonths = loanTermYears \* 12;

// Mortgage formula for monthly payment

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

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

return monthlyPayment;

}

// Method to calculate total payment over the life of the loan

public double calculateTotalPayment() {

return calculateMonthlyPayment() \* loanTermYears \* 12;

}

@Override

public String toString() {

return "Loan Amount: ₹" + principal + "\n" +

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

"Loan Term: " + loanTermYears + " years\n" +

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

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

}

}

**package** org.example.loan;

**import** java.util.Scanner;

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

// Method to accept input from the user

**public** LoanAmortizationCalculator acceptRecord() {

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

System.***out***.println("Enter the loan amount (principal): ");

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

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

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

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

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

// Return a new LoanAmortizationCalculator object with user inputs

**return** **new** LoanAmortizationCalculator(principal, annualInterestRate, loanTermYears);

}

// Method to print the loan details and payment info

**public** **void** printRecord(LoanAmortizationCalculator loan) {

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

System.***out***.println(loan); // Uses the overridden toString() method of LoanAmortizationCalculator

}

// Menu list method

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

System.***out***.println("1. Calculate Loan Amortization");

System.***out***.println("2. Exit");

}

}

**package** org.example.loan;

**import** java.util.Scanner;

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

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

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

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

**while** (**true**) {

util.menuList();

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

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

**switch** (choice) {

**case** 1:

// Accept the loan record from the user

LoanAmortizationCalculator loan = util.acceptRecord();

// Print the loan details and calculated payments

util.printRecord(loan);

**break**;

**case** 2:

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

System.*exit*(0);

**break**;

**default**:

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

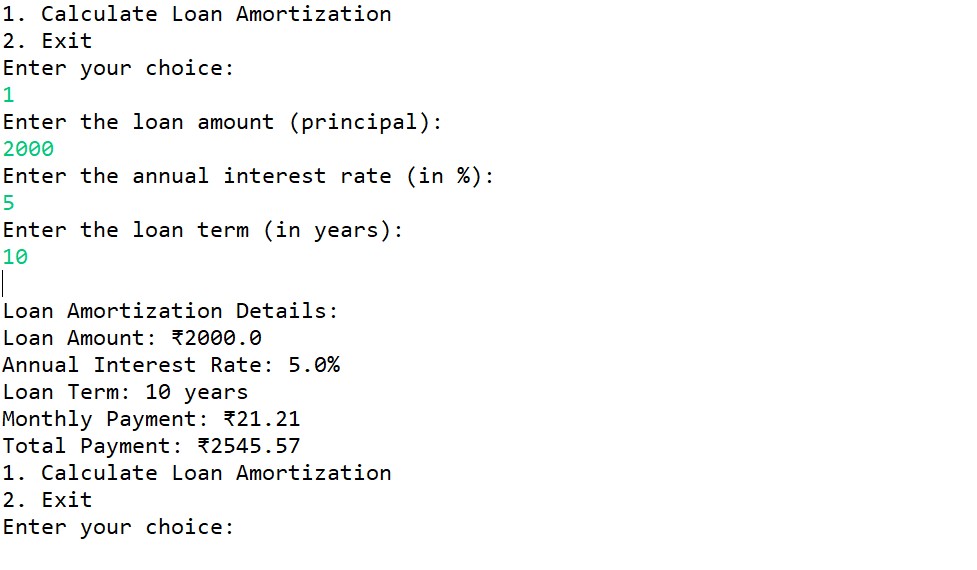
}

}

}

}

**Output –**



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.cicalci;

**public** **class** CompoundInterestCalculator {

**private** **double** initialInvestment;

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

**private** **double** compoundedNumber;

**private** **int** Years;

**public** CompoundInterestCalculator(**double** initialInvestment, **double** annualInterestRate, **double** compoundedNumber,

**int** years) {

**super**();

**this**.initialInvestment = initialInvestment;

**this**.annualInterestRate = annualInterestRate;

**this**.compoundedNumber = compoundedNumber;

Years = years;

}

**public** **double** getInitialInvestment() {

**return** initialInvestment;

}

**public** **void** setInitialInvestment(**double** initialInvestment) {

**this**.initialInvestment = initialInvestment;

}

**public** **double** getAnnualInterestRate() {

**return** annualInterestRate;

}

**public** **void** setAnnualInterestRate(**double** annualInterestRate) {

**this**.annualInterestRate = annualInterestRate;

}

**public** **double** getCompoundedNumber() {

**return** compoundedNumber;

}

**public** **void** setCompoundedNumber(**double** compoundedNumber) {

**this**.compoundedNumber = compoundedNumber;

}

**public** **int** getYears() {

**return** Years;

}

**public** **void** setYears(**int** years) {

Years = years;

}

**public double futureValue() {**

**return initialInvestment \* Math.*pow*(1 + (annualInterestRate / compoundedNumber)/100, compoundedNumber \* Years);**

**}**

**public double totalInterestEarned() {**

**return futureValue() - initialInvestment;**

**}**

**@Override**

**public String toString() {**

**return "InitialInvestment : "+initialInvestment+ "\n"+**

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

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

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

**}**

**}**

**CompoundInterestCalculatorUtil –**

**package org.example.cicalci;**

**import java.util.Scanner;**

**public class CompoundInterestCalculatorUtil {**

**public CompoundInterestCalculator acceptRecord() {**

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

**System.*out*.println("Enter the Initial Investment Amount(Principal) : ");**

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

**System.*out*.println("Enter the Annual Interest Rate : ");**

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

**System.*out*.println("Enter the number of times of compounded : ");**

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

**System.*out*.println("Enter the Years : ");**

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

**return new CompoundInterestCalculator(initialinvestment, annualiInterestRate, compoundedNumber, Years);**

**}**

**public void printRecord(CompoundInterestCalculator calculator) {**

**System.*out*.println("Compound Interest : ");**

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

**}**

**public void menuList() {**

**System.*out*.println("1.Compound Interest : ");**

**System.*out*.println("2.Exit ");**

**}**

**}**

**Main Program –**

**package org.example.cicalci;**

**import java.util.Scanner;**

**public class Program {**

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

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

**CompoundInterestCalculatorUtil calculator = new CompoundInterestCalculatorUtil();**

**while(true) {**

**calculator.menuList();**

**System.*out*.println("Enter the choice : ");**

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

**switch(choice) {**

**case 1:**

**CompoundInterestCalculator interestEarned = calculator.acceptRecord();**

**calculator.printRecord(interestEarned);**

**break;**

**case 2:**

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

**System.*exit*(0);**

**break;**

**default:**

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

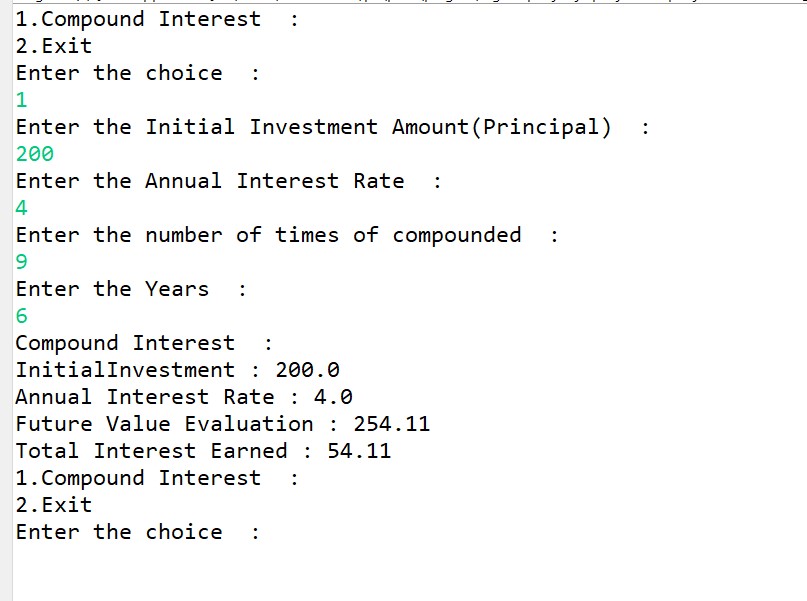
**}**

**}**

**}**

**}**

**Output –**

****

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.

**BMITracker -**

package org.example.bmi;

public class BMITracker {

private double weight;

private double height;

public BMITracker(double weight, double height) {

this.weight = weight;

this.height = height;

}

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 BMICalculation() {

return weight / (Math.*pow*(height, height));

}

public String classifyBMI() {

double bmi = BMICalculation();

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

"Weight :" +weight+ "\n"+

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

"Classification : "+ classifyBMI();

}

}

**BMITrackerUtil –**

package org.example.bmi;

import java.util.Scanner;

public class BMITrackerUtil {

public BMITracker acceptRecord() {

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

System.*out*.println("Enter your weight in kg: ");

double weight = sc.nextDouble();

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

double height = sc.nextDouble();

return new BMITracker(weight, height);

}

// Method to display BMI record

public void printRecord(BMITracker tracker) {

System.*out*.println("BMI Tracker : ");

System.*out*.println(tracker);

}

public void menuList() {

System.*out*.println("1.Calculate BMI :");

System.*out*.println("2.Exit");

}

}

**Main Program -**

**package** org.example.bmi;

**import** java.util.Scanner;

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

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

// **TODO** Auto-generated method stub

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

BMITrackerUtil abc = **new** BMITrackerUtil();

**while**(**true**) {

abc.menuList();

System.***out***.println("Enter choice : ");

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

**switch**(ch) {

**case** 1:

BMITracker tracker = abc.acceptRecord();

abc.printRecord(tracker);

**break**;

**case** 2:

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

**break**;

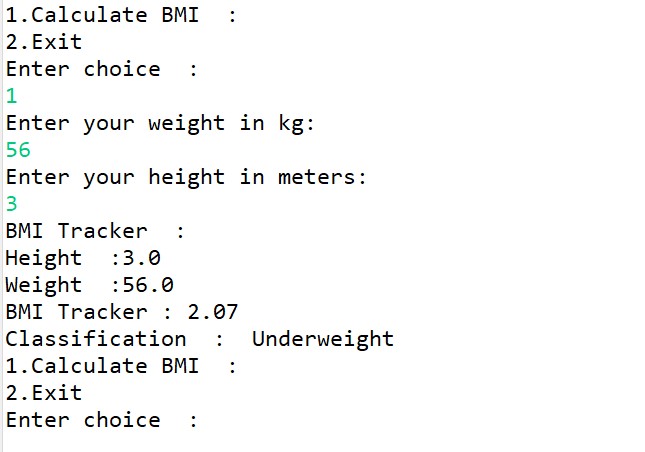
}

}

}

}

**Output –**

****

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.

**BMITracker -**

**package** org.example.bmi;

**public** **class** BMITracker {

**private** **double** weight;

**private** **double** height;

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

**this**.weight = weight;

**this**.height = height;

}

**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** BMICalculation() {

**return** weight / (Math.*pow*(height, height));

}

**public** String classifyBMI() {

**double** bmi = BMICalculation();

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

"Weight :" +weight+ "\n"+

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

"Classification : "+ classifyBMI();

}

}

**BMITrackerUtil –**

**package** org.example.bmi;

**import** java.util.Scanner;

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

**public** BMITracker acceptRecord() {

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

System.***out***.println("Enter your weight in kg: ");

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

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

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

**return** **new** BMITracker(weight, height);

}

// Method to display BMI record

**public** **void** printRecord(BMITracker tracker) {

System.***out***.println("BMI Tracker : ");

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

}

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

System.***out***.println("1.Calculate BMI :");

System.***out***.println("2.Exit");

}

}

**Main Program –**

**package** org.example.bmi;

**import** java.util.Scanner;

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

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

// **TODO** Auto-generated method stub

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

BMITrackerUtil abc = **new** BMITrackerUtil();

**while**(**true**) {

abc.menuList();

System.***out***.println("Enter choice : ");

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

**switch**(ch) {

**case** 1:

BMITracker tracker = abc.acceptRecord();

abc.printRecord(tracker);

**break**;

**case** 2:

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

**break**;

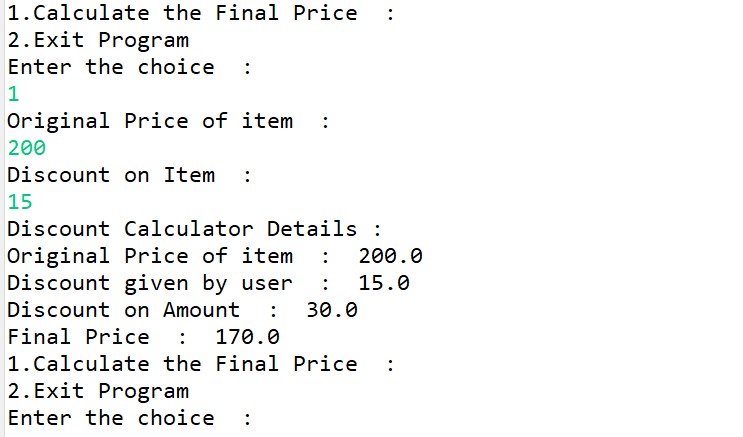
}

}

}

}

**Output –**



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.

**TollBoothRevenueManager -**

**package** org.example.toll;

**public** **class** TollBoothRevenueManager {

**private** **double** carTollRate;

**private** **double** truckTollRate;

**private** **double** motorcycleTollRate;

**private** **int** carCount;

**private** **int** truckCount;

**private** **int** motorcycleCount;

**public** TollBoothRevenueManager(**double** carTollRate, **double** truckTollRate, **double** motorcycleTollRate) {

**super**();

**this**.carTollRate = carTollRate;

**this**.truckTollRate = truckTollRate;

**this**.motorcycleTollRate = motorcycleTollRate;

}

**public** **double** getCarTollRate() {

**return** carTollRate;

}

**public** **void** setCarTollRate(**double** carTollRate) {

**this**.carTollRate = carTollRate;

}

**public** **double** getTruckTollRate() {

**return** truckTollRate;

}

**public** **void** setTruckTollRate(**double** truckTollRate) {

**this**.truckTollRate = truckTollRate;

}

**public** **double** getMotorcycleTollRate() {

**return** motorcycleTollRate;

}

**public** **void** setMotorcycleTollRate(**double** motorcycleTollRate) {

**this**.motorcycleTollRate = motorcycleTollRate;

}

**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 \* carTollRate) + (truckCount \* truckTollRate) + (motorcycleCount \* motorcycleTollRate);

}

**public** **int** calculateTotalVehicles() {

**return** carCount + truckCount + motorcycleCount;

}

@Override

**public** String toString() {

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

"Total Revenue : "+ calculateTotalRevenue();

}

}

**TollBoothRevenueManagerUtil -**

**package** org.example.toll;

**import** java.util.Scanner;

**public** **class** TollBoothRevenueManagerUtil {

**public** TollBoothRevenueManager acceptRecord() {

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

System.***out***.println("Enter the toll rate for cars : ");

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

System.***out***.println("Enter the toll rate for truck : ");

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

System.***out***.println("Enter the toll rate for motorcycle : ");

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

System.***out***.println("Enter the number of passing cars : ");

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

System.***out***.println("Enter the number of passing trucks : ");

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

System.***out***.println("Enter the number of passing motorcycles : ");

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

TollBoothRevenueManager manager = **new** TollBoothRevenueManager(carTollRate, truckTollRate, motorcycleTollRate);

manager.setCarCount(carCount);

manager.setTruckCount(truckCount);

manager.setMotorcycleCount(motorcycleCount);

**return** manager;

}

**public** **void** printRecord(TollBoothRevenueManager manager) {

System.***out***.println("Toll Booth Revenue Summary:");

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

}

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

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

System.***out***.println("2.Exit Program");

}

}

**Main Program -**

**package** org.example.toll;

**import** java.util.Scanner;

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

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

// **TODO** Auto-generated method stub

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

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

**while**(**true**) {

util.menuList();

System.***out***.println("Enter choice : ");

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

**switch**(ch) {

**case** 1:

TollBoothRevenueManager manager = util.acceptRecord();

util.printRecord(manager);

**break**;

**case** 2:

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

**break**;

**default**:

System.***out***.println("invalid entry");

**break**;

}

}

}

}

**Output –**

