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.

```
package in.Cdac.LoanAmortizationCalculator;

import java.util.Scanner;

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 LoanAmortizationCalculator() {
		// TODO Auto-generated constructor stub
	}
```

```
	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 / 100) /
12;
		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;
	}

	public String toString() {
		return "LoanAmortizationCalculator [principal=" + principal
+ "\n annualInterestRate=" + annualInterestRate
				+ "\n loanTerm=" + loanTerm + "\n
getPrincipal()=" + getPrincipal() + "\n getAnnualInterestRate()="
				+ getAnnualInterestRate() + "\n getLoanTerm()="
+ getLoanTerm() + "\n calculateMonthlyPayment()="
				+ calculateMonthlyPayment() + "\n
calculateTotalPayment()=" + calculateTotalPayment() + "]";
	}

}

class LoanAmortizationCalculatorUtil {
	private LoanAmortizationCalculator loan;

	public LoanAmortizationCalculatorUtil() {
		this.loan = new LoanAmortizationCalculator();
	}
```

```
	private static Scanner sc = new Scanner(System.in);

	public void acceptRecord() {
		Scanner scanner = new Scanner(System.in);
		System.out.print("Enter Principal Amount (Rs.): ");
		double principal = scanner.nextDouble();
		System.out.print("Enter Annual Interest Rate (%): ");
		double annualInterestRate = scanner.nextDouble();
		System.out.print("Enter Loan Term (years): ");
		int loanTerm = scanner.nextInt();
		loan = new LoanAmortizationCalculator(principal,
annualInterestRate, loanTerm);
	}

	public void printRecord() {
		System.out.println(loan.toString());
	}

	public static int menuList() {
		System.out.println("0.Exit");
		System.out.println("1.Accept Record");
		System.out.println("2.Print Record");
		System.out.print("Enter choice	:	");
		return sc.nextInt();
	}

	public static void releaseResource() {
		sc.close();
	}
}

public class Program {
	public static void main(String[] args) {
		int choice;
		LoanAmortizationCalculatorUtil util = new
LoanAmortizationCalculatorUtil();
		while ((choice = LoanAmortizationCalculatorUtil.menuList())
!= 0) {
			switch (choice) {
			case 1:
				util.acceptRecord();
				break;
			case 2:
				util.printRecord();
				break;
			}
		}
		LoanAmortizationCalculatorUtil.releaseResource();
	}
}
```

```
<terminated> Program [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe  (10-Sep-2024, 12:18:14 pm – 12:18:44 pm) [pid: 13236]
0.Exit
1.Accept Record
2.Print Record
Enter choice    :       1
Enter Principal Amount (Rs.): 10000
Enter Annual Interest Rate (%): 0.5
Enter Loan Term (years): 4
0.Exit
1.Accept Record
2.Print Record
Enter choice    :       2
LoanAmortizationCalculator [principal=10000.0
 annualInterestRate=0.5
 loanTerm=4
 getPrincipal()=10000.0
 getAnnualInterestRate()=0.5
 getLoanTerm()=4
 calculateMonthlyPayment()=210.46700938277783
 calculateTotalPayment()=10102.416450373335]
0.Exit
1.Accept Record
2.Print Record
Enter choice    :       0
```

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 in.Cdac.CompoundInterestCalculatorforInvestment;

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 CompoundInterestCalculator() {
		// TODO Auto-generated constructor stub
	}
	public double getPrincipal() {
		return principal;
	}

	public void setPrincipal(double principal) {
		this.principal = principal;
	}

	public double getAnnualInterestRate() {
		return annualInterestRate;
	}

	public void setAnnualInterestRate(double annualInterestRate) {
		this.annualInterestRate = annualInterestRate;
	}

	public int getNumberOfCompounds() {
		return numberOfCompounds;
	}

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

	public int getYears() {
		return years;
	}

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


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


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

```
	}

	public String toString() {
		return "CompoundInterestCalculator [\nprincipal=" +
principal + "\n annualInterestRate=" + annualInterestRate
			+ "\n numberOfCompounds=" + numberOfCompounds +
"\n years=" + years + "\n getPrincipal()=" + getPrincipal()
			+ "\n getAnnualInterestRate()=" +
getAnnualInterestRate() + "\n getNumberOfCompounds()="
			+ getNumberOfCompounds() + "\n getYears()=" +
getYears() + "\n calculateFutureValue()="
			+ calculateFutureValue() + "\n
calculateTotalInterest()=" + calculateTotalInterest() + "\n]";
	}

}

class CompoundInterestCalculatorUtil {
	private CompoundInterestCalculator investment;

	public CompoundInterestCalculatorUtil() {
		this.investment = new CompoundInterestCalculator();
	}

	private static Scanner sc = new Scanner(System.in);

	public void acceptRecord() {
		Scanner scanner = new Scanner(System.in);
		System.out.print("Enter Initial Investment Amount (Rs.): ");
		double principal = scanner.nextDouble();
		System.out.print("Enter Annual Interest Rate (%): ");
		double annualInterestRate = scanner.nextDouble();
		System.out.print("Enter Number of Compounds Per Year: ");
		int numberOfCompounds = scanner.nextInt();
		System.out.print("Enter Investment Duration (years): ");
		int years = scanner.nextInt();
		investment = new CompoundInterestCalculator(principal,
annualInterestRate, numberOfCompounds, years);
	}

	public void printRecord() {
		System.out.println(investment.toString());
	}

	public static int menuList() {
		System.out.println("0.Exit");
		System.out.println("1.Accept Record");
		System.out.println("2.Print Record");
		System.out.print("Enter choice	:	");
		return sc.nextInt();
	}

	public static void releaseResource() {
		sc.close();
	}
}

public class Program {
```

```
	public static void main(String[] args) {
		int choice;
		CompoundInterestCalculatorUtil util = new
CompoundInterestCalculatorUtil();
		while ((choice = CompoundInterestCalculatorUtil.menuList())
!= 0) {
			switch (choice) {
			case 1:
				util.acceptRecord();
				break;
			case 2:
				util.printRecord();
				break;
			}
		}
		CompoundInterestCalculatorUtil.releaseResource();
	}
}
```

```
Problems  Javadoc  Declaration  Console  Coverage
<terminated> Program (1) [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe (10-Sep-2024, 12:34:50 pm – 12:35:37 pm) [pid: 11652]
0.Exit
1.Accept Record
2.Print Record
Enter choice    :       1
Enter Initial Investment Amount (Rs.): 10000
Enter Annual Interest Rate (%): 0.5
Enter Number of Compounds Per Year: 12
Enter Investment Duration (years): 4
0.Exit
1.Accept Record
2.Print Record
Enter choice    :       2
CompoundInterestCalculator [
principal=10000.0
 annualInterestRate=0.5
 numberOfCompounds=12
 years=4
 getPrincipal()=10000.0
 getAnnualInterestRate()=0.5
 getNumberOfCompounds()=12
 getYears()=4
 calculateFutureValue()=10201.970903771129
 calculateTotalInterest()=201.9709037711291
]
0.Exit
1.Accept Record
2.Print Record
Enter choice    :       0
```

3. BMI (Body Mass Index) Tracker

Create a system to calculate and classify Body Mass Index (BMI). The system should:

1. Accept weight (in kilograms) and height (in meters) from the user.
2. Calculate the BMI using the formula:
   - **BMI Calculation:** `BMI = weight / (height * height)`
3. Classify the BMI into one of the following categories:
   - Underweight: BMI < 18.5
   - Normal weight: 18.5 ≤ BMI < 24.9
   - Overweight: 25 ≤ BMI < 29.9

- Obese: BMI ≥ 30

4. Display the BMI value and its classification.

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

```
package in.Cdac.BMITracker;

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";
		}
```

```
	}

	public String toString() {
		return "BMITracker [\nweight=" + weight + "\n height=" +
height + "\n getWeight()=" + getWeight()
			+ "\n getHeight()=" + getHeight() + "\n
calculateBMI()=" + calculateBMI() + "\n classifyBMI()="
			+ classifyBMI() + "\n]";
	}

}

class BMITrackerUtil {
	private BMITracker bmiTracker;
	private static Scanner sc = new Scanner(System.in);

	public void acceptRecord() {
		Scanner scanner = new Scanner(System.in);
		System.out.print("Enter Weight (in kilograms): ");
		double weight = scanner.nextDouble();
		System.out.print("Enter Height (in meters): ");
		double height = scanner.nextDouble();
		bmiTracker = new BMITracker(weight, height);
	}

	public void printRecord() {
		System.out.println(bmiTracker.toString());
	}

	public static int menuList() {
		System.out.println("0.Exit");
		System.out.println("1.Accept Record");
		System.out.println("2.Print Record");
		System.out.print("Enter choice	:	");
		return sc.nextInt();
	}

	public static void releaseResource() {
		sc.close();
	}
}

public class Program {
	public static void main(String[] args) {
		int choice;
		BMITrackerUtil util = new BMITrackerUtil();
		while ((choice = BMITrackerUtil.menuList()) != 0) {
			switch (choice) {
			case 1:
				util.acceptRecord();
				;
				break;
			case 2:
				util.printRecord();
				;
				break;
			}
		}
```

```
            BMITrackerUtil.releaseResource();
        }
}
```

```
<terminated> Program (2) [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe  (10-Sep-2024, 1:03:00 pm – 1:03:27 pm) [pid: 1552]
0.Exit
1.Accept Record
2.Print Record
Enter choice    :       1
Enter Weight (in kilograms): 55
Enter Height (in meters): 3
0.Exit
1.Accept Record
2.Print Record
Enter choice    :       2
BMITracker [
weight=55.0
 height=3.0
 getWeight()=55.0
 getHeight()=3.0
 calculateBMI()=6.111111111111111
 classifyBMI()=Underweight
]
0.Exit
1.Accept Record
2.Print Record
Enter choice    :       0
```

4. Discount Calculation for Retail Sales

Design a system to calculate the final price of an item after applying a discount. The system should:

1. Accept the original price of an item and the discount percentage from the user.
2. Calculate the discount amount and the final price using the following formulas:
   - **Discount Amount Calculation:** `discountAmount = originalPrice * (discountRate / 100)`
   - **Final Price Calculation:** `finalPrice = originalPrice - discountAmount`
3. Display the discount amount and the final price of the item, in Indian Rupees (₹).

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

```
package in.Cdac.DiscountCalculationforRetailSales;

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();
	}

	public String toString() {
		return "DiscountCalculator [\noriginalPrice=" +
originalPrice + "\n discountRate=" + discountRate
			+ "\n getOriginalPrice()=" + getOriginalPrice()
+ "\n getDiscountRate()=" + getDiscountRate()
			+ "\n calculateDiscountAmount()=" +
calculateDiscountAmount() + "\n calculateFinalPrice()="
			+ calculateFinalPrice() + "\n]";
	}

}

class DiscountCalculatorUtil {
	private DiscountCalculator discountCalculator;
	private static Scanner sc = new Scanner(System.in);

	public void acceptRecord() {
		Scanner scanner = new Scanner(System.in);
		System.out.print("Enter Original Price (Rs.): ");
		double originalPrice = scanner.nextDouble();
		System.out.print("Enter Discount Rate (%): ");
		double discountRate = scanner.nextDouble();
		discountCalculator = new DiscountCalculator(originalPrice,
discountRate);
	}
```

```
	public void printRecord() {
		if (discountCalculator != null) {
			System.out.println(discountCalculator.toString());
		} else {
			System.out.println("No record found!");
		}
	}

	public static int menuList() {
		System.out.println("0.Exit");
		System.out.println("1.Accept Record");
		System.out.println("2.Print Record");
		System.out.print("Enter choice	:	");
		return sc.nextInt();
	}

	public static void releaseResource() {
		sc.close();
	}
}

public class Program {
	public static void main(String[] args) {
		int choice;
		DiscountCalculatorUtil util = new DiscountCalculatorUtil();
		while ((choice = DiscountCalculatorUtil.menuList()) != 0) {
			switch (choice) {
			case 1:
				util.acceptRecord();
				break;
			case 2:
				util.printRecord();
				break;
			}
		}
		DiscountCalculatorUtil.releaseResource();
	}
}
```

```
<terminated> Program (3) [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe  (10-Sep-2024, 1:12:29 pm – 1:12:51 pm) [pid: 24456]
0.Exit
1.Accept Record
2.Print Record
Enter choice    :       1
Enter Original Price (Rs.): 1000
Enter Discount Rate (%): 20
0.Exit
1.Accept Record
2.Print Record
Enter choice    :       2
DiscountCalculator [
originalPrice=1000.0
 discountRate=20.0
 getOriginalPrice()=1000.0
 getDiscountRate()=20.0
 calculateDiscountAmount()=200.0
 calculateFinalPrice()=800.0
]
0.Exit
1.Accept Record
2.Print Record
Enter choice    :       0
```

5. Toll Booth Revenue Management

Develop a system to simulate a toll booth for collecting revenue. The system should:

1. Allow the user to set toll rates for different vehicle types: Car, Truck, and Motorcycle.
2. Accept the number of vehicles of each type passing through the toll booth.
3. Calculate the total revenue based on the toll rates and number of vehicles.
4. Display the total number of vehicles and the total revenue collected, in Indian Rupees (₹).

- **Toll Rate Examples:**
  - Car: ₹50.00
  - Truck: ₹100.00
  - Motorcycle: ₹30.00

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

```
package in.Cdac.TollBoothRevenueManagement;

import java.util.Scanner;

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

	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 getNumCars() {
		return numCars;
	}

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

	public int getNumTrucks() {
		return numTrucks;
	}

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

```
	public int getNumMotorcycles() {
		return numMotorcycles;
	}

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

	public double calculateTotalRevenue() {
		return (numCars * carRate) + (numTrucks * truckRate) +
(numMotorcycles * motorcycleRate);
	}

	public int calculateTotalVehicles() {
		return numCars + numTrucks + numMotorcycles;
	}

	public String toString() {
		return "TollBoothRevenueManager [\ncarRate=" + carRate + "\n
truckRate=" + truckRate + "\n motorcycleRate="
			+ motorcycleRate + "\n numCars=" + numCars + "\n
numTrucks=" + numTrucks + "\n numMotorcycles="
			+ numMotorcycles + "\n getCarRate()=" +
getCarRate() + "\n getTruckRate()=" + getTruckRate()
			+ "\n getMotorcycleRate()=" +
getMotorcycleRate() + "\n getNumCars()=" + getNumCars()
			+ "\n getNumTrucks()=" + getNumTrucks() + "\n
getNumMotorcycles()=" + getNumMotorcycles()
			+ "\n calculateTotalRevenue()=" +
calculateTotalRevenue() + "\n calculateTotalVehicles()="
			+ calculateTotalVehicles() + "\n]";
	}

}

class TollBoothRevenueManagerUtil {
	private TollBoothRevenueManager tollBooth;
	private static Scanner sc = new Scanner(System.in);

	public void acceptRecord() {
		//Scanner sc = new Scanner(System.in);
		System.out.print("Enter number of Cars: ");
		int numCars = sc.nextInt();
		System.out.print("Enter number of Trucks: ");
		int numTrucks = sc.nextInt();
		System.out.print("Enter number of Motorcycles: ");
		int numMotorcycles = sc.nextInt();

		tollBooth.setNumCars(numCars);
		tollBooth.setNumTrucks(numTrucks);
		tollBooth.setNumMotorcycles(numMotorcycles);
		//sc.close();
	}

	public void setTollRates() {
		//Scanner sc = new Scanner(System.in);
		System.out.print("Enter toll rate for Cars (Rs.): ");
```

```
            double carRate = sc.nextDouble();
            System.out.print("Enter toll rate for Trucks (Rs.): ");
            double truckRate = sc.nextDouble();
            System.out.print("Enter toll rate for Motorcycles (Rs.): ");
            double motorcycleRate = sc.nextDouble();

            tollBooth = new TollBoothRevenueManager(carRate, truckRate,
motorcycleRate);
            //sc.close();
      }

      public void printRecord() {
            System.out.println(tollBooth.toString());
      }

      public static int menuList() {
            System.out.println("0.Exit");
            System.out.println("1.setTollRates");
            System.out.println("2.Accept Record");
            System.out.println("3.Print Record");
            System.out.print("Enter choice     :     ");
            return sc.nextInt();
      }

      public static void releaseResource() {
            sc.close();
      }
}

public class Program {
      public static void main(String[] args) {
            int choice;
            TollBoothRevenueManagerUtil util = new
TollBoothRevenueManagerUtil();
            while ((choice = TollBoothRevenueManagerUtil.menuList()) !=
0) {
                  switch (choice) {
                  case 1:
                        util.setTollRates();;
                        break;
                  case 2:
                        util.acceptRecord();;
                        break;
                  case 3:
                        util.printRecord();;
                        break;
                  }
            }
            TollBoothRevenueManagerUtil.releaseResource();
      }
}
```

ASSIGNMENT NO.4

```
Program (4) [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe  (10-Sep-2024, 2:50:30 pm) [pid: 11876]
0.Exit
1.setTollRates
2.Accept Record
3.Print Record
Enter choice    :       1
Enter toll rate for Cars (Rs.): 100
Enter toll rate for Trucks (Rs.): 200
Enter toll rate for Motorcycles (Rs.): 50
0.Exit
1.setTollRates
2.Accept Record
3.Print Record
Enter choice    :       2
Enter number of Cars: 2
Enter number of Trucks: 3
Enter number of Motorcycles: 1
0.Exit
1.setTollRates
2.Accept Record
3.Print Record
Enter choice    :       3
TollBoothRevenueManager [
carRate=100.0
 truckRate=200.0
 motorcycleRate=50.0
 numCars=2
 numTrucks=3
 numMotorcycles=1
 getCarRate()=100.0
 getTruckRate()=200.0
 getMotorcycleRate()=50.0
 getNumCars()=2
 getNumTrucks()=3
 getNumMotorcycles()=1
 calculateTotalRevenue()=850.0
 calculateTotalVehicles()=6
]
0.Exit
1.setTollRates
2.Accept Record
3.Print Record
Enter choice    :
```