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.

Ans:-

LoanAmortizationCalculator1.java -

```
package org.example4;

import java.util.Scanner;


class LoanAmortizationCalculator1 {


        double principalAmount;

    double annualInterestRate;
```

```
        double loanTerm;
        double monthlyPayment;
        double totalAmountPaid;

        public LoanAmortizationCalculator1() {
            this.principalAmount = 0.0;
            this.annualInterestRate = 0.0;
            this.loanTerm = 0.0;
        }

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

        public void acceptrecord(){
                        System.out.println("Principal Amount   :   ");
//                      this.principalAmount = sc.nextDouble();
                        setPrincipalAmount(sc.nextDouble());

                        System.out.println("Interest Rate   :   ");
//                      this.annualInterestRate = sc.nextDouble();
                        setAnnualInterestRate(sc.nextDouble());

                        System.out.println("Loan Term   :   ");
//                      this.loanTerm = sc.nextDouble();
                        setLoanTerm(sc.nextDouble());
```

```
        }
        //getter and setter methods
        public double getPrincipalAmount() { //Inspector / Selector / Getter method
                return principalAmount;
            }

            public double getAnnualInterestRate() { //Inspector / Selector / Getter
method
                return annualInterestRate;
            }

            public double getLoanTerm() {  //Inspector / Selector / Getter method
                return loanTerm;
            }

            public void setPrincipalAmount(double principalAmount) { //Mutator /
Setter method
                this.principalAmount = principalAmount;
            }

            public void setAnnualInterestRate(double annualInterestRate) {
//Mutator / Setter method
                this.annualInterestRate = annualInterestRate;
            }
```

```
public void setLoanTerm(double loanTerm) { //Mutator / Setter method

        this.loanTerm = loanTerm;

}


public void calculateMonthlyPayment() {

  double monthlyInterestRate = (this.getAnnualInterestRate() / 12) / 100 ;

  double numberOfMonths = this.getLoanTerm() * 12;

  monthlyPayment = this.getPrprincipalAmount() * (monthlyInterestRate *
Math.pow(1 + monthlyInterestRate, numberOfMonths)) / (Math.pow(1 +
monthlyInterestRate, numberOfMonths) - 1);

  totalAmountPaid = monthlyPayment * this.getLoanTerm() * 12;


//                return monthlyPayment;//return is used as double is used as return type
in method


}

public void printRecord() {

//                      double monthlyPayment = calculateMonthlyPayment() ;//uss
value to call kiya h wo print krni h kyuki wo yha presnt nii h

                      System.out.println("Monthly Payment   :   " +
monthlyPayment);

                      System.out.println("Total Amount Paid   :   " +
totalAmountPaid);

                }



public static int menuList() {

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

```
                System.out.println("1. Accept Record.");
                System.out.println("2. Print Record.");
                System.out.println("Enter your choice      :          ");
                int choice = sc.nextInt();
                return choice;
            }
        }
```

LoanCalculator.java -

```
package org.example4;


public class LoanCalculator {


    public static void main(String[] args) {
        int choice;
        LoanAmortizationCalculator1 l = new LoanAmortizationCalculator1();


        while ((choice = LoanAmortizationCalculator1.menuList()) != 0) {
            switch (choice) {
            case 1: {


                l.acceptrecord();
                l.calculateMonthlyPayment();
                break;
            }
            case 2:
```

```
                    l.printRecord();

                    break;

                }

            }

        }

}
```

2. Compound Interest Calculator for Investment

Develop a system to compute the future value of an investment with compound interest. The system should:

1. Accept the initial investment amount, annual interest rate, number of times the interest is compounded per year, and investment duration (in years) from the user.
2. Calculate the future value of the investment using the formula:
   - **Future Value Calculation:**
     - `futureValue = principal * (1 + annualInterestRate / numberOfCompounds)^(numberOfCompounds * years)`
   - **Total Interest Earned:** `totalInterest = futureValue - principal`
3. Display the future value and the total interest earned, in Indian Rupees (₹).

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

Ans-

CompoundInterestCalculator.java -

```
package cdac.compound;


public class CompoundInterestCalculator {

    double principal;

    double annualInterestRate;

    int numberOfCompounds;

    int years;
```

```
double futureValue;

double totalInterest;


public CompoundInterestCalculator() {

        this.principal = 0.0;

        this.annualInterestRate = 0.0;

        this.numberOfCompounds= 0;

        this.years = 0;

}


//getter methods

public double getprincipal() { //Inspector / Selector / Getter method

        return principal;

}


public double getannualInterestRate() {

        return annualInterestRate;

}


public int getnumberOfCompounds() {

        return numberOfCompounds;

}

public int getyears() {

        return years;

}
```

```
//      setter methods
        public void setprincipal( double principal) { //Mutator / Setter method
                this.principal = principal;
        }

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

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

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

//      public double getFutureValue() {
//              return calculateFutureValue();
//      }
//
//      public double getTotalInterest() {
//              return calculateTotalInterest();
```

```
//        }


        public double calculateFutureValue() { //this method returns the value of the
formula

                futureValue = this.getprincipal() * Math.pow((1 +
this.getannualInterestRate() /
this.getnumberOfCompounds()),(this.getnumberOfCompounds() * this.getyears()));

                System.out.println(this.getprincipal() +" "+getannualInterestRate()+"
"+getnumberOfCompounds()+" "+getyears());

                return futureValue;//return is used as double is used as return type in
method

        }



        public double calculateTotalInterest() { // here the parameter futureValue is
coming from the function call from main method

                totalInterest = futureValue - this.getprincipal();

                return totalInterest;//return is used as double is used as return type in
method

        }

}
```

CompoundInterestCalculatorUtil.java –

```
package cdac.compound;


import java.util.Scanner;

//import cdac.compound.CompoundInterestCalculator;


public class CompoundInterestCalculatorUtil {
```

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

        private CompoundInterestCalculator cmp = new
CompoundInterestCalculator();

        public void acceptrecord() {

                System.out.println("Principal Amount   :   ");
//              this.principal = sc.nextDouble();
                cmp.setprincipal(sc.nextDouble());

                System.out.println("Interest Rate   :   ");
//              this.annualInterestRate = sc.nextDouble();
                cmp.setannualInterestRate(sc.nextDouble());

                System.out.println("Number of times the interest is compounded per year
:   ");
//              this.numberOfCompounds = sc.nextInt();
                cmp.setnumberOfCompounds(sc.nextInt());

                System.out.println("Investment duration (in years)    :   ");
//              this.years = sc.nextInt();
                cmp.setyears(sc.nextInt());

        }
```

```
        public void printRecord() { // two parameters are passed as in main method

                System.out.println("Future Value  :   " + cmp.calculateFutureValue());

                System.out.println("Total interest earned   :  " +
cmp.calculateTotalInterest());

        }

        public static int menuList() {

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

                System.out.println("1. Accept Record.");

                System.out.println("2. Print Record.");

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

                int choice = sc.nextInt();

                return choice;

        }

}
```

Program.java –

```
package cdac.compound;


import java.util.Scanner;

//import cdac.compound.CompoundInterestCalculator;


public class CompoundInterestCalculatorUtil {


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

```
        private CompoundInterestCalculator cmp = new
CompoundInterestCalculator();


        public void acceptrecord() {


                System.out.println("Principal Amount   :   ");
//              this.principal = sc.nextDouble();
                cmp.setprincipal(sc.nextDouble());


                System.out.println("Interest Rate   :   ");
//              this.annualInterestRate = sc.nextDouble();
                cmp.setannualInterestRate(sc.nextDouble());


                System.out.println("Number of times the interest is compounded per year
:   ");
//              this.numberOfCompounds = sc.nextInt();
                cmp.setnumberOfCompounds(sc.nextInt());


                System.out.println("Investment duration (in years)    :   ");
//              this.years = sc.nextInt();
                cmp.setyears(sc.nextInt());


        }


        public void printRecord() { // two parameters are passed as in main method
```

```
            System.out.println("Future Value  :   " + cmp.calculateFutureValue());

            System.out.println("Total interest earned   :  " +
cmp.calculateTotalInterest());

        }


        public static int menuList() {

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

            System.out.println("1. Accept Record.");

            System.out.println("2. Print Record.");

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

            int choice = sc.nextInt();

            return choice;

        }

}
```

```
Console
Program [Java Application] D:\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.fu
0. Exit.
1. Accept Record.
2. Print Record.
Enter your choice       :
1
Principal Amount   :
33
Interest Rate   :
45
Number of times the interest is compounded per year   :
6
Investment duration (in years)    :
7
0. Exit.
1. Accept Record.
2. Print Record.
Enter your choice       :
2
Future Value  :   3.5818617550536527E40
Total interest earned   :  3.5818617550536527E40
0. Exit.
1. Accept Record.
2. Print Record.
Enter your choice       :
```

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.

Ans-

BMITracker.java –

```
package cdac.bmi;

public class BMITracker {
	float weight;
	float height;
	float BMI;

	public BMITracker() {
		this.weight = 0.0f;
		this.height = 0.0f;
	}

	// getter
	public float getweight() {
		return weight;
	}

	public float getheight() {
		return weight;
	}

	// setter
	public void setweight(float weight) {
		this.weight = weight;
	}
```

```
	public void setheight(float height) {
		this.height = height;
	}

	public float calculateBMI() { // this method returns the value of the formula
		BMI = weight / (height * height);
		return BMI;// return is used as double is used as return type in method
	}

	public String classifyBMI() { // here the parameter futureValue is coming from
the function call from main
								// method
		String bm;
		if (BMI < 18.5) {
			bm = "Underweight";
		} else if (BMI >= 18.5 && BMI < 24.9) {
			bm = "Normal weight";
		} else if (BMI > 25 && BMI < 29.9) {
			bm = "Overweight";
		} else {
			bm = "Obese";
		}

		return bm;// return is used as double is used as return type in method
	}
}
```

BMITrackerUtil.java –

```
package cdac.bmi;

import java.util.Scanner;

public class BMITrackerUtil {

	BMITracker bm = new BMITracker();

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

	public void acceptRecord() {
		System.out.println("Weight (in kilograms)   :   ");
		bm.weight = sc.nextFloat();

		System.out.println("Height (in meters)   :   ");
		bm.height = sc.nextFloat();

	}

	public void printRecord() { // two parameters are passed as in main method
```

```
		System.out.println("BMI value  :   " + bm.calculateBMI());
		System.out.println("BMI classification   :  " + bm.classifyBMI());
	}

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

Program.java -

```
package cdac.bmi;

//import cdac.compound.CompoundInterestCalculatorUtil;

public class Program {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		int choice;

		BMITrackerUtil bmi = new BMITrackerUtil();

		while ((choice = BMITrackerUtil.menuList()) != 0) {
			switch (choice) {
			case 1: {

				bmi.acceptRecord();
//				ci.calculateFutureValue(); // the calculated result is
stored in 'fV' for returning it to print statement
//				ci.calculateTotalInterest(); //the fV is passes here as
argument for the calculation of another formula then stored in 'tI'
				break;
			}
			case 2:
				bmi.printRecord();
				;// the calculated results are passed as arguments
				break;
			}

		}

	}
```

}

```
Console
Program (1) [Java Application] D:\Eclipse\eclipse\plugins\org.eclipse.justj.openjd
0. Exit.
1. Accept Record.
2. Print Record.
Enter your choice       :
1
Weight (in kilograms)   :
23
Height (in meters)   :
123
0. Exit.
1. Accept Record.
2. Print Record.
Enter your choice       :
2
BMI value  :   0.0015202591
BMI classification   :  Underweight
0. Exit.
1. Accept Record.
2. Print Record.
Enter your choice       :
```

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.

Ans-

DiscountCalculator.java -

```
package cdac.discount;

public class DiscountCalculator {
        private double originalPrice;
```

```
	private double discountRate;
	private double discountAmount;
	private double finalPrice;

	public DiscountCalculator() { // constructor
		this.originalPrice = 0.0;
		this.discountRate = 0.0;
	}

	public double getoriginalPrice() { // getter
		return originalPrice;
	}

	public double getdiscountRate() { // getter
		return discountRate;
	}

	public void setoriginalPrice(double originalPrice) { // setter
		this.originalPrice = originalPrice;
	}

	public void setdiscountRate(double discountRate) { // setter
		this.discountRate = discountRate;
	}

	// biusiness logic
	public double calculateDiscount() {
		discountAmount = this.getoriginalPrice() * (this.getdiscountRate() / 100);
		return discountAmount;// return is used as double is used as return type
in method
	}

	public double calculatefinalPrice() {
		finalPrice = this.getoriginalPrice() - discountAmount;
		return finalPrice;// return is used as double is used as return type in
method
	}
}
```

DiscountCalculatorUtil.java -

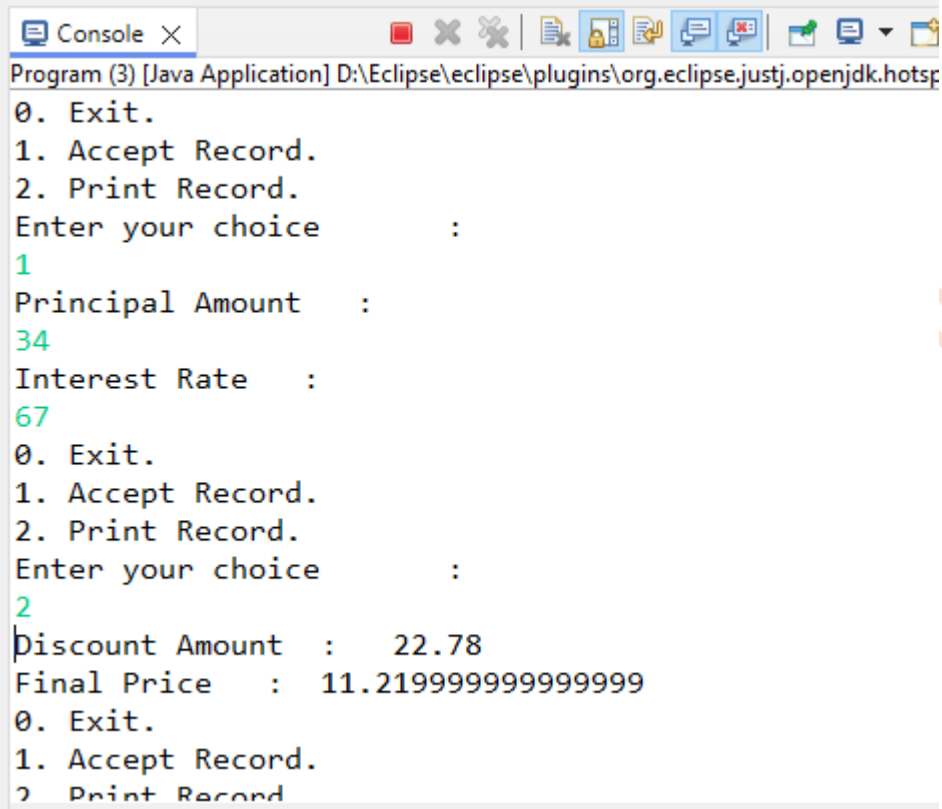
```
package cdac.discount;

import java.util.Scanner;

public class DiscountCalculatorUtil {

	DiscountCalculator dc = new DiscountCalculator();

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

```
        public void acceptrecord() {

                System.out.println("Principal Amount   :   ");
//              this.originalPrice = sc.nextDouble();
                dc.setoriginalPrice(sc.nextDouble());

                System.out.println("Interest Rate   :   ");
//              this.discountRate = sc.nextDouble();
                dc.setdiscountRate(sc.nextDouble());

        }

        public void printRecord() {
                System.out.println("Discount Amount  :   " + dc.calculateDiscount());
                System.out.println("Final Price   :  " + dc.calculatefinalPrice());
        }

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

Program.java –

```
package cdac.discount;

public class Program {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                DiscountCalculatorUtil dcl = new DiscountCalculatorUtil();

                int choice;
                while ((choice = DiscountCalculatorUtil.menuList()) != 0) {
                        switch (choice) {
                        case 1: {

                                dcl.acceptrecord();
                                break;
                        }
                        case 2: {

                                dcl.printRecord();
                                break;
```

```
                    }

                    }
                }
            }
}
```



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.

Ans-

TollBoothRevenueManager.java -

```
package cdac.toll.booth;

public class TollBoothRevenueManager {

	float carToll;
	float truckToll;
	float motorcycleToll;

	int NoOfCar;
	int NoOfTruck;
	int NoOfMotorcycle;
	int totalVehicles;

	public TollBoothRevenueManager() {
		this.carToll = 0.0f;
		this.truckToll = 0.0f;
		this.motorcycleToll = 0.0f;
	}

	public float getcarToll() { // getter
		return carToll;
	}

	public float gettruckToll() { // getter
		return truckToll;
	}

	public float getmotorcycleToll() { // getter
		return motorcycleToll;
	}

	public float getNoOfCar() { // getter
		return NoOfCar;
	}

	public float getNoOfTruck() { // getter
		return NoOfTruck;
	}

	public float getNoOfMotorcycle() { // getter
```

```
		return NoOfMotorcycle;
	}

	public void setcarToll(float carToll) { // setter
		this.carToll = carToll;
	}

	public void settruckToll(float truckToll) { // setter
		this.truckToll = truckToll;
	}

	public void setmotorcycleToll(float motorcycleToll) { // setter
		this.motorcycleToll = carToll;
	}

	public void setNoOfCar(int NoOfCar) { // setter
		this.NoOfCar = NoOfCar;
	}

	public void setNoOfTruck(int NoOfTruck) { // setter
		this.NoOfTruck = NoOfTruck;
	}

	public void setNoOfMotorcycle(int NoOfMotorcycle) { // setter
		this.NoOfMotorcycle = NoOfMotorcycle;
	}

	// business logic
	public float calculateRevenue() {
		float carReveue = this.getcarToll() * NoOfCar;
		float truckReveue = this.gettruckToll() * NoOfTruck;
		float motorcycleReveue = this.getmotorcycleToll() * NoOfMotorcycle;

		float totalRevenue = carReveue + truckReveue + motorcycleReveue;
		return totalRevenue;// return is used as float is used as return type in
method
	}

	public float calculateTotalVehicles() {
		totalVehicles = NoOfCar + NoOfMotorcycle + NoOfTruck;
		return totalVehicles;// return is used as float is used as return type in
method
	}

}
```

TollBoothRevenueManagerUtil.java –

```
package cdac.toll.booth;

import java.util.Scanner;

public class TollBoothRevenueManagerUtil {

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

	public void acceptrecord() {

		System.out.println("Car Toll   :   ");
//		this.cartoll = sc.nextFloat();
		tb.setcarToll(sc.nextFloat());

		System.out.println("Truck Toll   :   ");
//		this.trucktoll = sc.nextFloat();
		tb.settruckToll(sc.nextFloat());

		System.out.println("Motorcycle Toll   :   ");
//		this.motorcycletoll = sc.nextFloat();
		tb.setmotorcycleToll(sc.nextFloat());

		System.out.println("No. of Cars   :   ");
//		this.NoOfCar = sc.nextInt();
		tb.setNoOfCar(sc.nextInt());

		System.out.println("No. of Truck    :   ");
//		this.NoOfTruck = sc.nextInt();
		tb.setNoOfTruck(sc.nextInt());

		System.out.println("No. of Motorcycle    :   ");
//		this.NoOfMotorcycle = sc.nextInt();
		tb.setNoOfMotorcycle(sc.nextInt());
	}

	public void printRecord() { // two parameters are passed as in main method
		System.out.println("Total number of vehicles  :   " +
tb.calculateTotalVehicles());
		System.out.println("Total revenue collected   :  " + tb.calculateRevenue());
	}

	public static int menuList() {
		System.out.println("0. Exit. ");
		System.out.println("1. Accept Record.");
		System.out.println("2. Print Record.");
```

```
            System.out.println("Enter your choice        :        ");
            int choice = sc.nextInt();
            return choice;
        }
}
```

Program.java -

```
package cdac.toll.booth;

public class Program {

        public static void main(String[] args) {
            TollBoothRevenueManagerUtil tbr = new TollBoothRevenueManagerUtil();

            int choice;
            while ((choice = TollBoothRevenueManagerUtil.menuList()) != 0) {
                switch (choice) {
                case 1: {

                    tbr.acceptrecord();
                    break;
                }
                case 2: {

                    tbr.printRecord();
                    break;
                }

                }
            }

        }

}
```

ASSIGNMENT NO.4

```
@ Javadoc  Declaration  Console ×
Program (4) [Java Application] D:\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspo
0. Exit.
1. Accept Record.
2. Print Record.
Enter your choice       :
1
Car Toll    :
5
Truck Toll    :
8
Motorcycle Toll    :
9
No. of Cars     :
3
No. of Truck     :
4
No. of Motorcycle     :
6
0. Exit.
1. Accept Record.
2. Print Record.
Enter your choice       :
2
Total number of vehicles  :   13.0
Total revenue collected   :  77.0
0. Exit.
1. Accept Record.
2. Print Record.
Enter your choice       :
```