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:

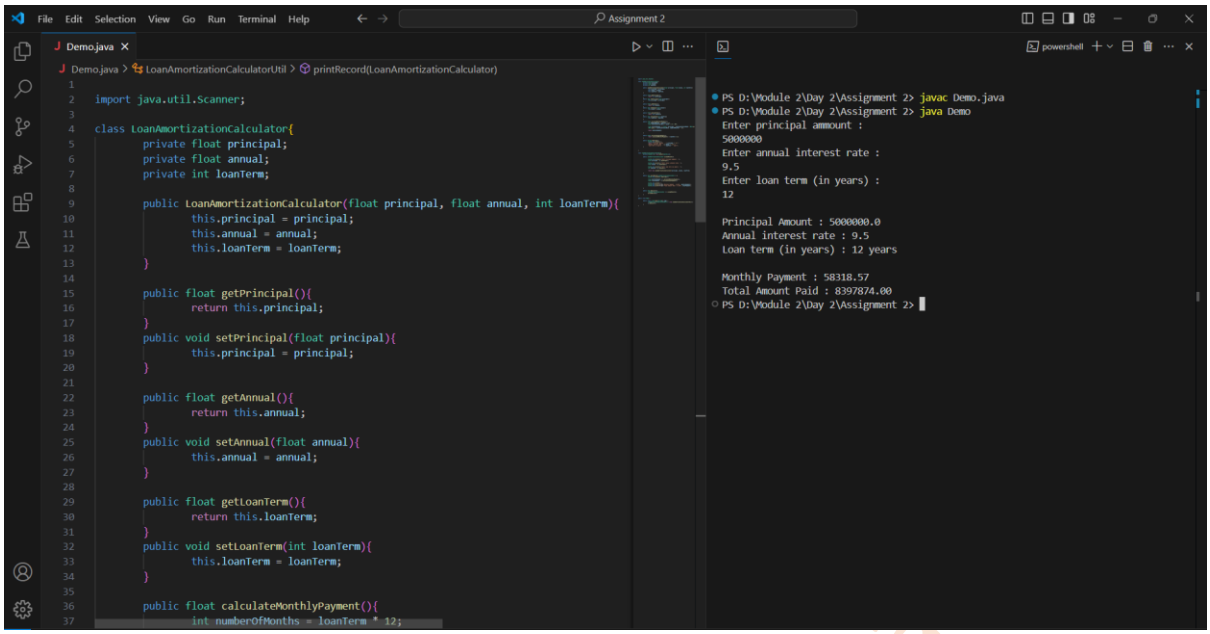
1. Accept the principal amount (loan amount), annual interest rate, and loan term (in years) from the user.
2. Calculate the monthly payment using the standard mortgage formula:
   - **Monthly Payment Calculation:**
     - `monthlyPayment = principal * (monthlyInterestRate * (1 + monthlyInterestRate)^(numberOfMonths)) / ((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.

```
import java.util.Scanner;

class LoanAmortizationCalculator{
    private float principal;
    private float annual;
    private int loanTerm;

    public LoanAmortizationCalculator(float principal, float annual, int loanTerm){
        this.principal = principal;
        this.annual = annual;
        this.loanTerm = loanTerm;
    }

    public float getPrincipal(){
        return this.principal;
    }
    public void setPrincipal(float principal){
        this.principal = principal;
```

```
    }

    public float getAnnual(){
        return this.annual;
    }
    public void setAnnual(float annual){
        this.annual = annual;
    }

    public float getLoanTerm(){
        return this.loanTerm;
    }
    public void setLoanTerm(int loanTerm){
        this.loanTerm = loanTerm;
    }

    public float calculateMonthlyPayment(){
        int numberOfMonths = loanTerm * 12;
        float monthlyInterestRate = annual / 12 / 100 ;

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

        return monthlyPayment;

    }

    public float calculateTotalPayment(){
        return calculateMonthlyPayment() * loanTerm * 12;
    }

    public String toString(){
        System.out.println( );
        return "Principal Amount : " + principal + "\n" +
        "Annual interest rate : " + annual + "\n" +
        "Loan term (in years) : " + loanTerm + " years";
    }

}

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

    public LoanAmortizationCalculator acceptRecord(){

        System.out.println("Enter principal ammount : ");
        float principal = sc.nextFloat();

        System.out.println("Enter annual interest rate : ");
```

```
        float annual = sc.nextFloat();

        System.out.println("Enter loan term (in years) : ");
        int loanTerm = sc.nextInt();

        return new LoanAmortizationCalculator(principal, annual, loanTerm);

    }

    public void printRecord(LoanAmortizationCalculator l ){
        System.out.println(l.toString());

        float monthlyPayment = l.calculateMonthlyPayment();
        float totalPayment = l.calculateTotalPayment();

        System.out.println( );
        System.out.printf("Monthly Payment : %.2f%n", monthlyPayment);
        System.out.printf("Total Amount Paid : %.2f%n", totalPayment);

    }

    public void menuList(){
        LoanAmortizationCalculator l = acceptRecord();
        printRecord(l);
    }
}

public class Demo {

    public static void main(String[] args) {
        LoanAmortizationCalculatorUtil u = new LoanAmortizationCalculatorUtil();
        u.menuList();
    }
}
```



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 (₹).

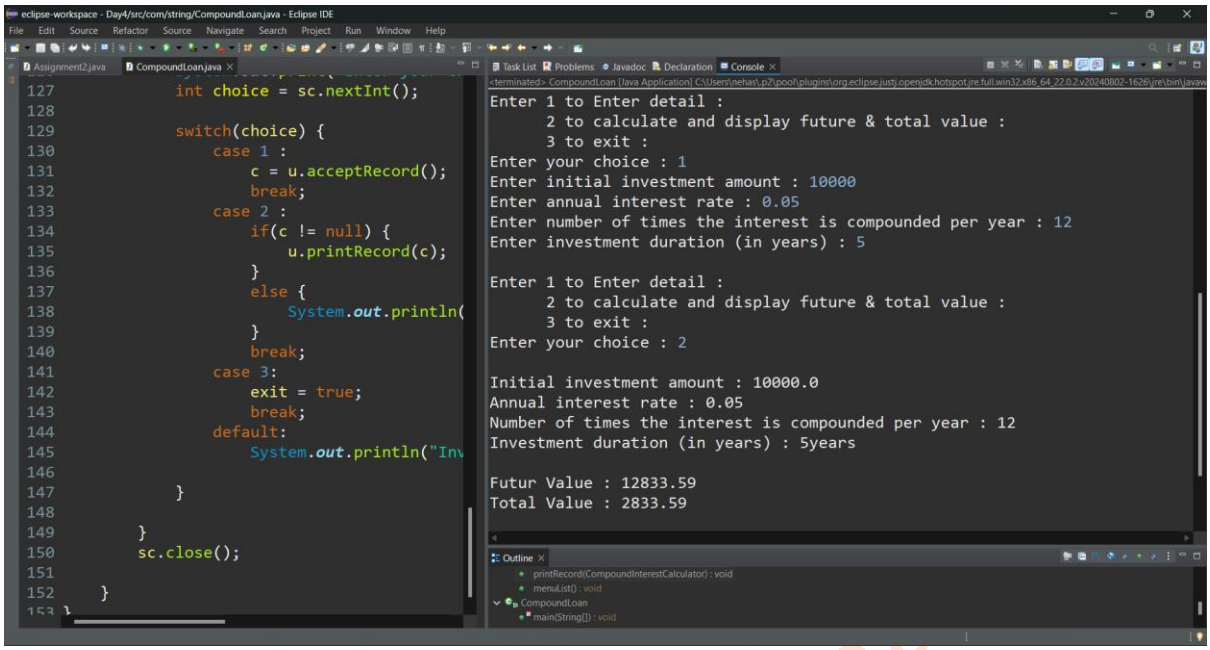
Define the class `CompoundInterestCalculator` with fields, an appropriate constructor, getter and setter methods, a `toString` method and business logic methods. Define the class `CompoundInterestCalculatorUtil` with methods `acceptRecord`, `printRecord`, and `menuList`. Define the class `Program` with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

```
package com.string;

import java.util.Scanner;

class CompoundInterestCalculator{
        private double amount;
        private double annual;
        private int numberOfCompounds;
        private int loanTerm;
```

```
	public CompoundInterestCalculator(double amount, double annual, int
numberOfCompounds, int loanTerm) {
		this.amount = amount;
		this.annual = annual;
		this.loanTerm = loanTerm;
		this.numberOfCompounds = numberOfCompounds;
	}

	public double getAmount() {
		return amount;
	}
	public void setAmount(double amount) {
		this.amount = amount;
	}

	public double getAnnual() {
		return annual;
	}
	public void setAnnual(double annual) {
		this.annual = annual;
	}

	public int getLoanTerm() {
		return loanTerm;
	}
	public void setLoanTerm(int loanTerm) {
		this.loanTerm = loanTerm;
	}

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

	public double futureValue() {

		return amount * Math.pow((1 + annual / numberOfCompounds),
numberOfCompounds * loanTerm);
	}

	public double totalInterest() {
		return  futureValue() - amount;
	}

	public String toString() {
		System.out.println( );
		return "Initial investment amount : " + amount + "\n" +
				"Annual interest rate : " + annual + "\n" +
```

```
                                "Number of times the interest is compounded per year : " +
numberOfCompounds + "\n" +
                                "Investment duration (in years) : " + loanTerm + "years";
        }

}

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

        public CompoundInterestCalculator acceptRecord() {
                System.out.print("Enter initial investment amount : ");
                double amount = sc.nextDouble();

                System.out.print("Enter annual interest rate : ");
                double annual = sc.nextDouble();

                System.out.print("Enter number of times the interest is compounded per year :
");
                int numberOfCompounds = sc.nextInt();

                System.out.print("Enter investment duration (in years) : ");
                int loanTerm = sc.nextInt();

                return new
CompoundInterestCalculator(amount,annual,numberOfCompounds,loanTerm);
        }

        public void printRecord(CompoundInterestCalculator c) {
                System.out.println(c.toString());

                System.out.println( );
                System.out.printf("Futur Value : %.2f%n", c.futureValue());
                System.out.printf("Total Value : %.2f%n", c.totalInterest());
        }

        public void menuList() {
                System.out.println( );
                System.out.println("Enter 1 to Enter detail :");
                System.out.println("     2 to calculate and display future & total value :");
                System.out.println("     3 to exit :");
        }
}

public class CompoundLoan {
        public static void main(String args[]) {
                CompoundInterestCalculatorUtil u = new CompoundInterestCalculatorUtil();
                CompoundInterestCalculator c = null;
                Scanner sc = new Scanner(System.in);
                boolean exit = false;
```

```
		while(!exit) {
			u.menuList();
			System.out.print("Enter your choice : ");
			int choice = sc.nextInt();

			switch(choice) {
				case 1 :
					c = u.acceptRecord();
					break;
				case 2 :
					if(c != null) {
						u.printRecord(c);
					}
					else {
						System.out.println("Please enter the detail
first!");
					}
				   break;
				case 3:
					exit = true;
					break;
				default:
					System.out.println("Invalid choice! Please try again...");

			}

		}
		sc.close();

	}
}
```



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.

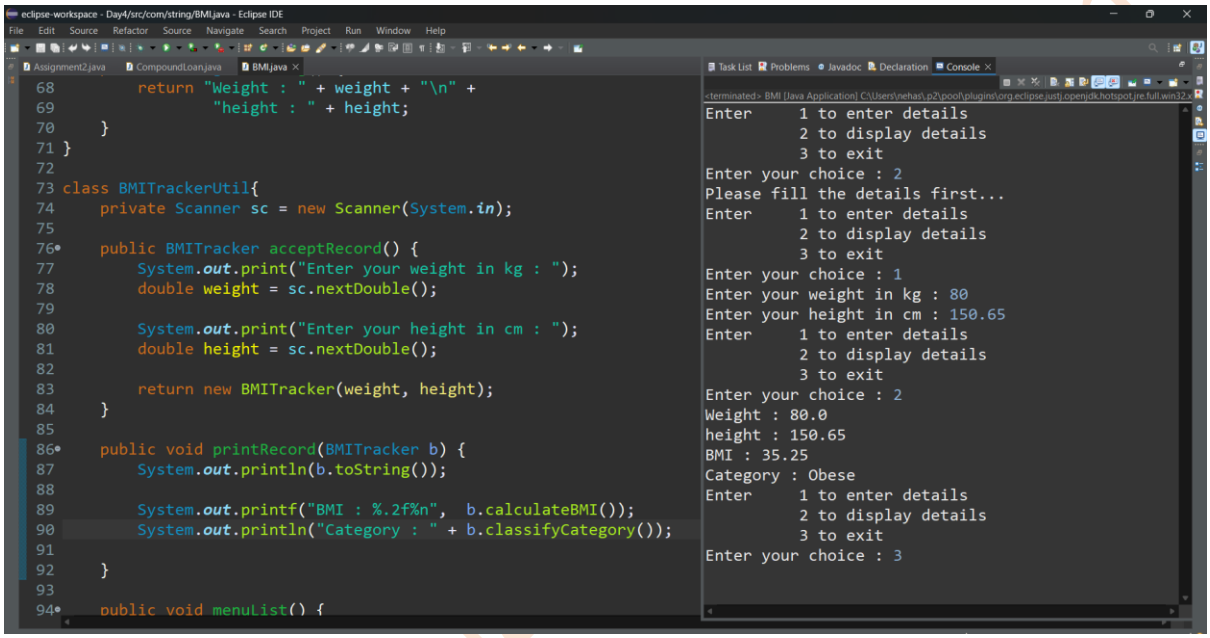
Define the class `BMITracker` with fields, an appropriate constructor, getter and setter methods, a `toString` method, and business logic methods. Define the class `BMITrackerUtil` with methods `acceptRecord`, `printRecord`, and `menuList`. Define the class `Program` with a `main` method to test the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

```
package com.string;
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() {
		double h = height / 100;
		double bmi = weight / (h * h);
		return bmi;
	}

	public String classifyCategory() {
		String category = "";
		double bmi = calculateBMI();
		if(bmi < 18.5) {
			category = "Underweight";
		}
		else if(bmi >= 18.5 && bmi < 24.9) {
			category = "Normal weight";
		}
		else if(bmi >= 25 && bmi < 29.9) {
			category = "Overweight";
		}
		else {
			category = "Obese";
		}
		return category;

	}

	public String toString() {
		return "Weight : " + weight + "\n" +
				"height : " + height;
	}
}

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

	public BMITracker acceptRecord() {
		System.out.print("Enter your weight in kg : ");
```

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

		System.out.print("Enter your height in cm : ");
		double height = sc.nextDouble();

		return new BMITracker(weight, height);
	}

	public void printRecord(BMITracker b) {
		System.out.println(b.toString());

		System.out.printf("BMI : %.2f%n",  b.calculateBMI());
		System.out.println("Category : " + b.classifyCategory());

	}

	public void menuList() {
		System.out.println("Enter     1 to enter details");
		System.out.println("     2 to display details");
		System.out.println("     3 to exit");
	}
}

public class BMI {
	public static void main(String args[]) {
		BMITrackerUtil u = new BMITrackerUtil();
		BMITracker b = null;
		Scanner sc = new Scanner(System.in);

		boolean exit = false;

		while(exit != true) {
			u.menuList();
			System.out.print("Enter your choice : ");
			int choice = sc.nextInt();

			switch(choice) {
				case 1:
					b = u.acceptRecord();
					break;
				case 2:
					if(b != null) {
						u.printRecord(b);
					}
					else {
						System.out.println("Please fill the details
first...");
					}
					break;
				case 3:
```

```
                        exit = true;
                        break;
                default:
                        System.out.println("Invalid choice! Please try again..");
                }
        }
        sc.close();
    }
}
```



4. Discount Calculation for Retail Sales

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

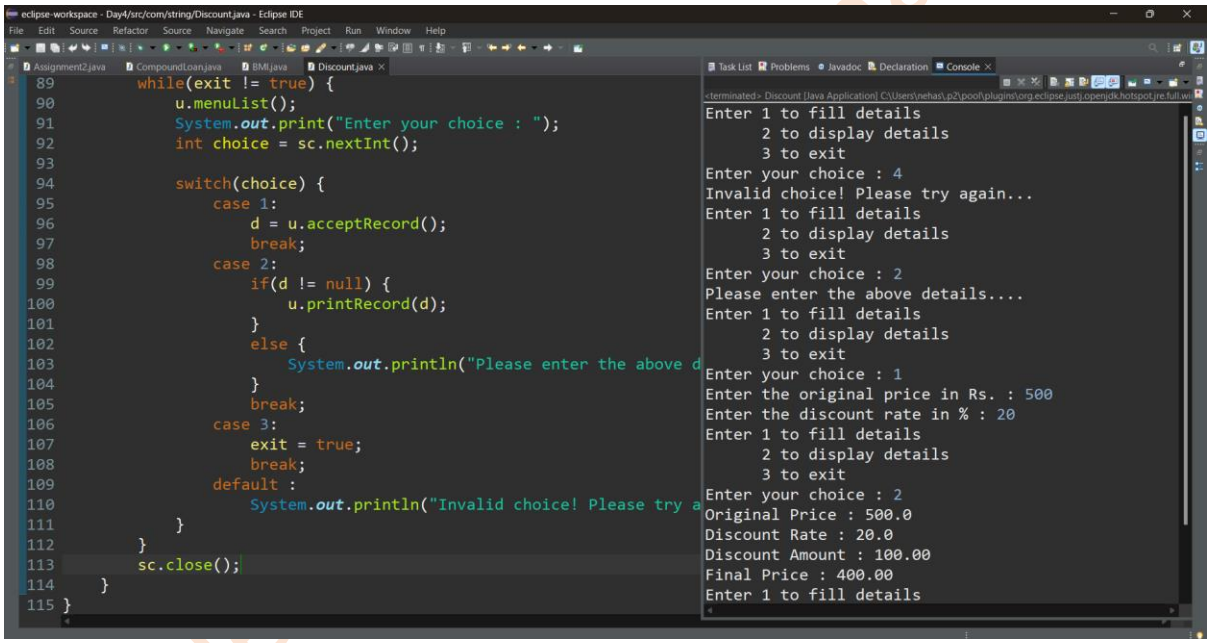
1. Accept the original price of an item and the discount percentage from the user.
2. Calculate the discount amount and the final price using the following formulas:
   - **Discount Amount Calculation:** `discountAmount = originalPrice * (discountRate / 100)`
   - **Final Price Calculation:** `finalPrice = 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 com.string;
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 discountAmount() {
		return originalPrice * (discountRate / 100);
	}

	public double finalPrice() {
		return originalPrice - discountAmount();
	}

	public String toString() {
		return "Original Price : " + originalPrice + "\n" +
				"Discount Rate : " + discountRate;
	}


}

class DiscountCalculatorUtil{
	private Scanner sc = new Scanner(System.in);
```

```
        public DiscountCalculator acceptRecord() {

                System.out.print("Enter the original price in Rs. : ");
                double originalPrice = sc.nextDouble();

                System.out.print("Enter the discount rate in % : ");
                double discountRate = sc.nextDouble();

                return new DiscountCalculator(originalPrice, discountRate);
        }

        public void printRecord(DiscountCalculator d) {
                System.out.println(d.toString());
                System.out.printf("Discount Amount : %.2f%n", d.discountAmount());
                System.out.printf("Final Price : %.2f%n", d.finalPrice());
        }

        public void menuList() {
                System.out.println("Enter 1 to fill details");
                System.out.println("      2 to display details");
                System.out.println("      3 to exit");
        }
}

public class Discount {
        public static void main(String[] args) {
                DiscountCalculatorUtil u = new DiscountCalculatorUtil();
                DiscountCalculator d = null;
                Scanner sc = new Scanner(System.in);

                boolean exit = false;

                while(exit != true) {
                        u.menuList();
                        System.out.print("Enter your choice : ");
                        int choice = sc.nextInt();

                        switch(choice) {
                                case 1:
                                        d = u.acceptRecord();
                                        break;
                                case 2:
                                        if(d != null) {
                                                u.printRecord(d);
                                        }
```

```
                                        else {
                                                System.out.println("Please enter the above
details....");
                                        }
                                        break;
                                case 3:
                                        exit = true;
                                        break;
                                default :
                                        System.out.println("Invalid choice! Please try again...");
                                }
                        }
                        sc.close();
                }
}
```



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

ASSIGNMENT NO.4

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 com.string;
import java.util.Scanner;

class TollBoothRevenueManager{
	private double carToll;
	private double truckToll;
	private double motorToll;

	private int carCount;
	private int truckCount;
	private int motorCount;

	public TollBoothRevenueManager(double carToll, double truckToll, double
motorToll, int carCount, int truckCount, int motorCount) {
		this.carToll = carToll;
		this.truckToll = truckToll;
		this.motorToll = motorToll;
		this.carCount = carCount;
		this.truckCount = truckCount;
		this.motorCount = motorCount;
	}

	public double getCarToll() {
		return carToll;
	}
	public void setCarToll(double carToll) {
		this.carToll = carToll;
	}

	public double getTruckToll() {
		return truckToll;
	}
	public void setTruckToll(double truckToll) {
		this.truckToll = truckToll;
	}

	public double getMotorToll() {
		return motorToll;
	}
	public void setMotorToll(double motorToll) {
		this.motorToll = motorToll;
```

```
        }

        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 getMotorCount() {
                return motorCount;
        }
        public void setMotorCount(int motorCount) {
                this.motorCount = motorCount;
        }

        public int totalNoOfVehicles() {
                return carCount + truckCount + motorCount;
        }

        public double totalRevenue() {
                return (carCount*carToll) + (truckCount*truckToll) +(motorCount*motorToll);
        }

        public String toString() {
                return "Car Toll : " + carToll + "\n" +
                                "Truck Toll : " + truckToll + "\n" +
                                "Motorcycle Toll : " + motorToll + "\n" +
                                "Car Count : " + carCount + "\n" +
                                "Truck Count : " + truckCount + "\n" +
                                "Motorcycle Count : " + motorCount;
        }
}

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

        public TollBoothRevenueManager acceptRecord() {

                System.out.print("Enter car toll : ");
```

```
                double carToll = sc.nextDouble();

                System.out.print("Enter truck toll : ");
                double truckToll = sc.nextDouble();

                System.out.print("Enter motorcycle toll : ");
                double motorToll = sc.nextDouble();

                System.out.print("Enter car count : ");
                int carCount = sc.nextInt();

                System.out.print("Enter truck count : ");
                int truckCount = sc.nextInt();

                System.out.print("Enter Motorcycle count : ");
                int motorCount = sc.nextInt();

                return new
TollBoothRevenueManager(carToll,truckToll,motorToll,carCount,truckCount,motorCount);
        }

        public void printRecord(TollBoothRevenueManager t) {
                System.out.println(t.toString());
                System.out.println("Total no. of vehicles : " + t.totalNoOfVehicles());
                System.out.printf("Total Revenue : %.2f%n", t.totalRevenue());
        }

        public void menuList() {
                System.out.println("Enter 1 to fill details");
                System.out.println("     2 to display details");
                System.out.println("     3 to exit");
        }
}

public class Tollbooth {
        public static void main(String[] args) {
                TollBoothRevenueManagerUtil u = new TollBoothRevenueManagerUtil();
                TollBoothRevenueManager d = null;
                Scanner sc = new Scanner(System.in);

                boolean exit = false;

                while(exit != true) {
                        u.menuList();
                        System.out.print("Enter your choice : ");
                        int choice = sc.nextInt();
```

```
                        switch(choice) {
                                case 1:
                                        d = u.acceptRecord();
                                        break;
                                case 2:
                                        if(d != null) {
                                                u.printRecord(d);
                                        }
                                        else {
                                                System.out.println("Please enter the above
details....");
                                        }
                                        break;
                                case 3:
                                        exit = true;
                                        break;
                                default :
                                        System.out.println("Invalid choice! Please try again...");
                        }
                }
                sc.close();
        }
}
```