ASSIGNMENT NO.3

Note:

- The assignment is designed to practice class, fields, and methods only.
- Create a separate project for each question.
- Do not use getter/setter methods or constructors for these assignments.
- Define two classes: one class to implement the logic and another class to test it.

1. Loan Amortization Calculator

Implement a system to calculate and display the monthly payments for a mortgage loan. The system should:

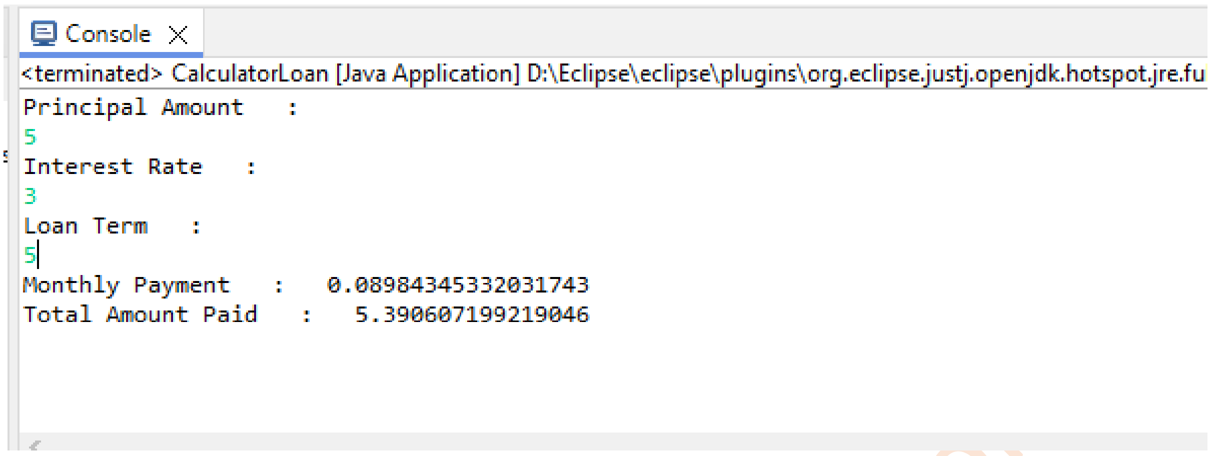
1. Accept the principal amount (loan amount), annual interest rate, and loan term (in years) from the user.
2. Calculate the monthly payment using the standard mortgage formula:
    - **Monthly Payment Calculation:**
        - `monthlyPayment = principal * (monthlyInterestRate * (1 + monthlyInterestRate)^(numberOfMonths)) / ((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 class LoanAmortizationCalculator with methods acceptRecord, calculateMonthlyPayment & printRecord and test the functionality in main method.

Ans:-

```
package org.examplejava;



import java.util.Scanner;



class LoanAmortizationCalculator{

    double principalAmount;

    double annualInterestRate;

    double loanTerm;

    double monthlyPayment;

    double totalAmountPaid;
    public void acceptrecord() {
        Scanner sc = new Scanner(System.in);

            System.out.println("Principal Amount   :   ");
            this.principalAmount = sc.nextDouble();

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

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

                sc.close();
    }

    public void calculateMonthlyPayment() {
        double monthlyInterestRate = (annualInterestRate / 12) / 100 ;
        double numberOfMonths = loanTerm * 12;
        monthlyPayment = principalAmount * (monthlyInterestRate *
Math.pow(1 + monthlyInterestRate, numberOfMonths)) / (Math.pow(1 +
monthlyInterestRate, numberOfMonths) - 1);
        totalAmountPaid = monthlyPayment * loanTerm * 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 class CalculatorLoan{

          public static void main(String[] args) {

                    // TODO Auto-generated method stub

                    LoanAmortizationCalculator loan = new
LoanAmortizationCalculator();

                    loan.acceptrecord();

                    loan.calculateMonthlyPayment();

                    loan.printRecord();



          }


}
```



```
<terminated> CalculatorLoan [Java Application] D:\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.fu
Principal Amount   :
5
Interest Rate   :
3
Loan Term   :
5
Monthly Payment   :   0.08984345332031743
Total Amount Paid   :   5.390607199219046
```

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 class CompoundInterestCalculator with methods acceptRecord , calculateFutureValue, printRecord and test the functionality in main method.

Ans:-

```
package org.javaques;

import java.util.Scanner;


class CompoundInterestCalculator{

	double principal;

	double annualInterestRate;

	int numberOfCompounds;
	int years;



	public void acceptrecord() {
		Scanner sc = new Scanner(System.in);

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

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

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

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

		sc.close();
	}

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

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

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

	}




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

		double totalInterest = futureValue - principal;

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

	}



	public void printRecord(double x, double y ) { // two parameters are passed as in main method

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

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

	}



}

public class CompoundInterestCalc {

	public static void main(String[] args) {



		CompoundInterestCalculator ci = new CompoundInterestCalculator();
```

ci.acceptrecord();

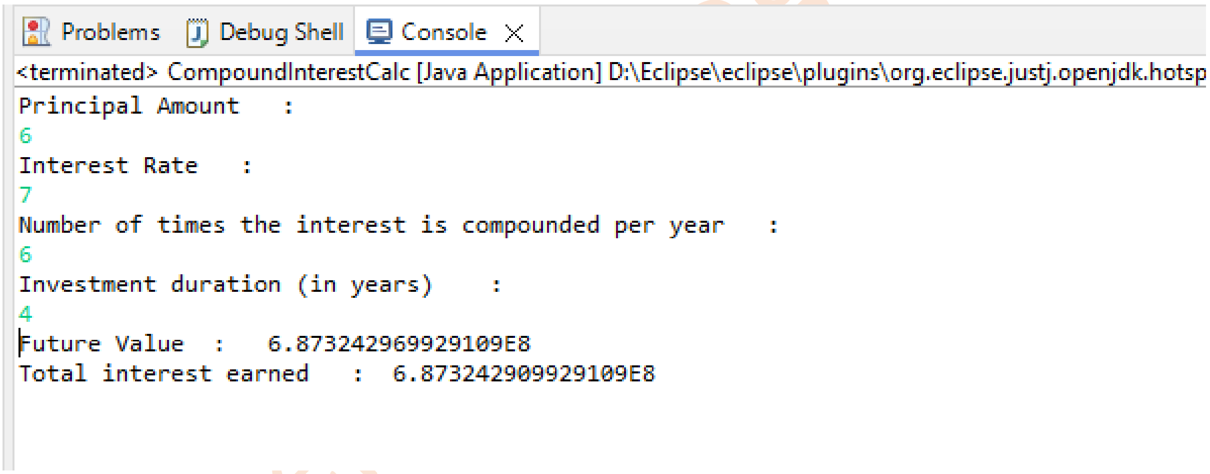
double fV = ci.calculateFutureValue(); // the calculated result is stored in 'fV' for returning it to print statement

double tI = ci.calculateTotalInterest(fV); //the fV is passes here as argument for the calculation of another formula then stored in 'tI'

ci.printRecord(fV , tI);//the calculated results are passed as arguments

}

}



```
<terminated> CompoundInterestCalc [Java Application] D:\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotsp
Principal Amount   :
6
Interest Rate   :
7
Number of times the interest is compounded per year   :
6
Investment duration (in years)    :
4
Future Value  :   6.873242969929109E8
Total interest earned   :  6.873242909929109E8
```

3. BMI (Body Mass Index) Tracker

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

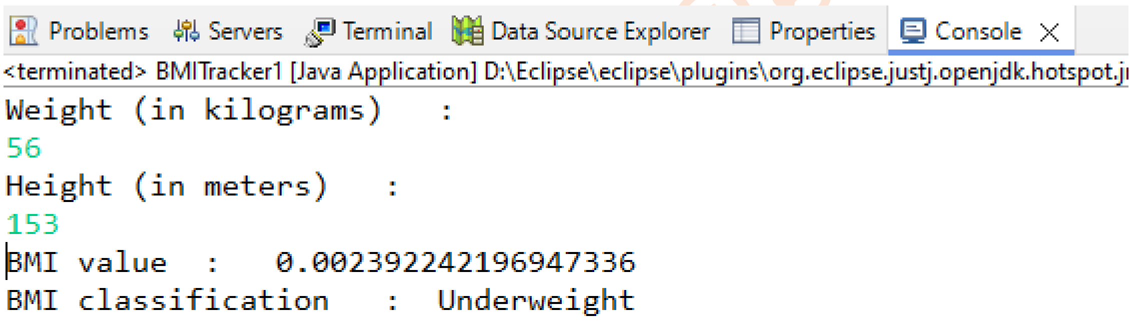
1. Accept weight (in kilograms) and height (in meters) from the user.
2. Calculate the BMI using the formula:
   - **BMI Calculation:** `BMI = weight / (height * height)`
3. Classify the BMI into one of the following categories:
   - Underweight: BMI < 18.5
   - Normal weight: 18.5 ≤ BMI < 24.9
   - Overweight: 25 ≤ BMI < 29.9
   - Obese: BMI ≥ 30
4. Display the BMI value and its classification.

Define class BMITracker with methods acceptRecord, calculateBMI, classifyBMI & printRecord and test the functionality in main method.

Ans:-

```
package org.javaques;
import java.util.Scanner;

class BMITracker{
	float weight;
	float height;


	public void acceptRecord() {
		Scanner sc = new Scanner(System.in);

		System.out.println("Weight (in kilograms)   :   ");
		this.weight = sc.nextFloat();

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

		sc.close();
	}

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

	}


	public String classifyBMI(float BMI) { // 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
```

```
        }

        public void printRecord(double BMI, String classifybmi ) { // two parameters are
passed as in main method
                System.out.println("BMI value  :   " + BMI);
                System.out.println("BMI classification  :  "+ classifybmi);
        }
}
public class BMITracker1 {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
                BMITracker c = new BMITracker();
                c.acceptRecord();
                float b = c.calculateBMI();
                String bi = c.classifyBMI(b);
                c.printRecord(b, bi);
        }

}
```


Problems Servers Terminal Data Source Explorer Properties Console

```
<terminated> BMITracker1 [Java Application] D:\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.ji
Weight (in kilograms)   :
56
Height (in meters)   :
153
BMI value  :   0.002392242196947336
BMI classification   :  Underweight
```

4. Discount Calculation for Retail Sales

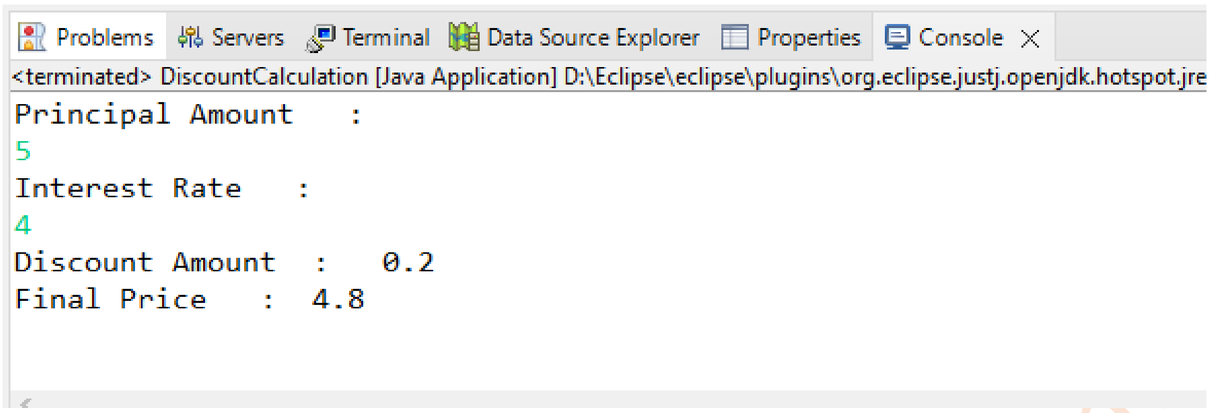
Design a system to calculate the final price of an item after applying a discount. The system should:

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 class DiscountCalculator with methods acceptRecord, calculateDiscount & printRecord and test the functionality in main method.

Ans:-

```
package org.javaques;

import java.util.Scanner;

class DiscountCalculator{
	double originalPrice;
	double discountRate;
	double discountAmount;
	double finalPrice;

	public void acceptrecord() {
		Scanner sc = new Scanner(System.in);

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

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

		sc.close();
	}

	public void calculateDiscount() {
		discountAmount = originalPrice * (discountRate / 100);
		finalPrice = originalPrice - discountAmount;
//		return discountAmount;//return is used as double is used as return type 
in method
	}


	public void printRecord() {
		System.out.println("Discount Amount  :   " + discountAmount);
		System.out.println("Final Price   :  "+ finalPrice);
	}
}
public class DiscountCalculation {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		DiscountCalculator dc = new DiscountCalculator();
		dc.acceptrecord();
		dc.calculateDiscount();
		dc.printRecord();
	}

}
```



```
<terminated> DiscountCalculation [Java Application] D:\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre
Principal Amount    :
5
Interest Rate    :
4
Discount Amount  :   0.2
Final Price    :  4.8
```

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 (₹).

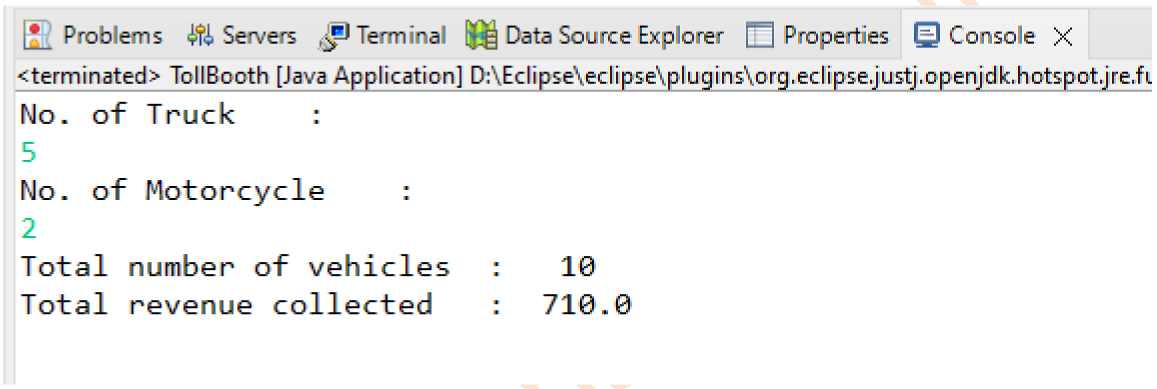
- **Toll Rate Examples:**
    - Car: ₹50.00
    - Truck: ₹100.00
    - Motorcycle: ₹30.00

Define class TollBoothRevenueManager with methods acceptRecord, setTollRates, calculateRevenue & printRecord and test the functionality in main method.

Ans:-

```
package org.javaques;

import java.util.Scanner;

class TollBoothRevenueManager{
        float cartoll;
        float trucktoll;
        float motorcycletoll;

        int NoOfCar;
        int NoOfTruck;
        int NoOfMotorcycle;
        int totalVehicles;
        public void acceptrecord() {
                Scanner sc = new Scanner(System.in);

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

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

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

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

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

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

                sc.close();
        }

//      public void setTollRates() {
//              this.cartoll = 50;
//              this.trucktoll = 100;
//              this.motorcycletoll = 30;
//      }


        public float calculateRevenue() {
                float carReveue = cartoll * NoOfCar;
                float truckReveue = trucktoll * NoOfTruck;
                float motorcycleReveue = motorcycletoll * NoOfMotorcycle;

                float totalRevenue = carReveue + truckReveue + motorcycleReveue;
                totalVehicles = NoOfCar + NoOfMotorcycle +  NoOfTruck ;

                return totalRevenue;//return is used as double is used as return type in
method
        }

        public void printRecord(double r) { // two parameters are passed as in main
method
                System.out.println("Total number of vehicles  :   " + totalVehicles);
                System.out.println("Total revenue collected   :  "+ r);
```

```
	}

}
public class TollBooth {

	public static void main(String[] args) {

		TollBoothRevenueManager tbm = new TollBoothRevenueManager();
		tbm.acceptrecord();
		float r = tbm.calculateRevenue();
		tbm.printRecord(r);
	}

}
```



```
<terminated> TollBooth [Java Application] D:\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.fu
No. of Truck     :
5
No. of Motorcycle     :
2
Total number of vehicles  :   10
Total revenue collected   :  710.0
```