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.

Code:

```
package Toll;

import java.util.Scanner;

class LoanCalculator{
	int principal;
	int annualInterestRate;
	int loanTerm;

	float monthlyInterestRate;
	float monthlyPayment;
	int numberOfMonths;

	Scanner sc =new Scanner(System.in);
  public void acceptRecord() {
	System.out.print("Enter the Principl Ammont:   ");
	principal = sc.nextInt();
	System.out.print("Enter the Annual Interest Rate:   ");
	annualInterestRate = sc.nextInt();
	System.out.print("Enter Loan Term in Year:   ");
	loanTerm = sc.nextInt();

  }
  public void calculateMonthlyPayment() {
	monthlyInterestRate = annualInterestRate / 12f / 100f;
	numberOfMonths = loanTerm * 12;
	monthlyPayment = (float)(principal* (monthlyInterestRate * Math.pow((1 +
monthlyInterestRate),numberOfMonths))/ (Math.pow((1 + monthlyInterestRate),numberOfMonths) - 1));
	System.out.print("your Monthly Payment:   "+ monthlyPayment);


  }
```

```
    public void printRecord() {
        float totalAmountPaid = monthlyPayment * numberOfMonths;
        System.out.print("\nTotal Amount Paid Over the Life of the Loan: "+ totalAmountPaid);


    }





}


public class LoanAmortizationCalculator {
        public static void main(String[] args) {
                LoanCalculator inst1 = new LoanCalculator();
                inst1.acceptRecord();
                inst1.calculateMonthlyPayment();
                inst1.printRecord();
        }

}
```

Output:

```
Console ×
<terminated> LoanAmortizationCalculator [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe  (08-Sep-2024, 1:2
Enter the Principl Ammont:   1000000
Enter the Annual Interest Rate:   5
Enter Loan Term in Year:   15
your Monthly Payment:   7907.8647
Total Amount Paid Over the Life of the Loan: 1423415.6
```

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.

→

Code

```
import java.util.Scanner;

class InterestCalculator{
    Scanner sc = new Scanner(System.in);
    double principalAmmount;
    double annualInterestRate;
    int numberOfCompounds;
    int years;

    double futureValue;
    double totalInterest;

    void acceptRecord() {
            System.out.print("Enter the Principal Ammount:   ");
            principalAmmount= sc.nextDouble();
            System.out.print("Enter The Annual Interest Rate:   ");
            annualInterestRate= sc.nextDouble()/100;
            System.out.print("Enter the number of times interest is compounded per year: ");
            numberOfCompounds = sc.nextInt();
            System.out.print("Enter the investment duration (in years): ");
            years = sc.nextInt();
            }

    void calculateFutureValue() {
            futureValue = principalAmmount * Math.pow(1 + (annualInterestRate / numberOfCompounds),
numberOfCompounds * years);

            totalInterest = futureValue - principalAmmount;

    }

    void printRecord() {
             System.out.print("\nFuture Value: "+ futureValue);
             System.out.println("\nTotal Interest Earned: " + totalInterest);
    }



}
```

```
public class CompoundInterestCalculator {
    public static void main(String[] args) {
            InterestCalculator inst1 = new InterestCalculator();
            inst1.acceptRecord();
            inst1.calculateFutureValue();
            inst1.printRecord();
    }

}
```

Output :

```
Console ×
<terminated> CompoundInterestCalculator [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe  (08-Sep-2024, 12:24:25 am – 12:24:34
Enter the Principal Ammount:   2000
Enter The Annual Interest Rate:   7
Enter the number of times interest is compounded per year: 20
Enter the investment duration (in years): 3

Future Value: 2466.4516425800034
Total Interest Earned: 466.4516425800034
```

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.

➔ Code:

```
package Assignment3;
import java.util.Scanner;

class BMITracker{
	private float weight;
	private float height;
	private float meter;
	private float bmi;
	Scanner sc = new Scanner(System.in);
	public void acceptRecord(){
		System.out.println("Enter your Weight in kg: ");
		this.weight = sc.nextFloat();
		System.out.println("Enter Your height Centimeter: ");
		this.height = sc.nextFloat();
	}
		public void calculateBMI() {
			this.meter = height/100;
			this.bmi = weight/(meter*meter);
		}
		public void  classifyBMI() {
			if (bmi<=18.4) {
				System.out.println("Your BMI is Underweight");
			}
			else if(18.4<=bmi && 24.9>=bmi) {
				System.out.println("Your BMI is Normal");
			}
			else {
				System.out.println("Your BMI is Overweighted");
			}
		}

		public void  printRecord () {
			System.out.println("BMI is: "+ bmi);
		}
	}
```

```
package Assignment3;
public class BmiCalculator {
public static void main(String[] args) {
        BMITracker BMI = new BMITracker();
        BMI.acceptRecord();
        BMI.calculateBMI();
        BMI.printRecord ();
        BMI.classifyBMI();

}
```

```
<terminated> BmiCalculator (1) [Java Application] C:\Program File
Enter your Weight in kg:
75
Enter Your height Centimeter:
182
BMI is: 22.642193
Your BMI is Normal
```

}

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.

→

Code:

```
package Toll;

import java.util.Scanner;

class Calculator{
	int discountRate;
	int originalPrice;
	float discountAmount;
	float  finalPrice;
	Scanner sc = new Scanner(System.in);

	void acceptRecord() {
		System.out.print("Enter The Original price: ");
		originalPrice= sc.nextInt();
		System.out.print("Enter the Discount Rate: ");
		discountRate=sc.nextInt();

	}

	void calculateDiscount() {
		discountAmount = originalPrice * (discountRate/100.0f);
		finalPrice = originalPrice-discountAmount;

	}

	void  printRecord () {
		System.out.print("Total Discount Amount: " + discountAmount);
		System.out.println("\nFinal Price is: "+ finalPrice);
	}
}

public class DiscountCalculator {
public static void main(String[] args) {
//	Scanner sc = new Scanner(System.in);
	Calculator inst1 = new Calculator();
	inst1.acceptRecord();
	inst1.calculateDiscount();
	inst1.printRecord();
}
}
```

Output:

```
<terminated> DiscountCalculator [Java Application] C:\Program Files\Java\jdk-17.0.1\bin\javaw.exe  (07-Sep-2024, 10:30:39 pm
Enter The Original price: 2000
Enter the Discount Rate: 10
Total Discount Amount: 200.0
Final Price is: 1800.0
```

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

Code:

```
package Toll;
import java.util.Scanner;
class TollRevenue{
        private int numberOfCar;
        private int numberOfTruck;
        private int numberOfMotorcycle;

        private float tollOfCar;
        private float tollOfTruck;
        private float tollOfMotorcycle;
        private float totalRevenue;

        Scanner sc = new Scanner(System.in);
  public void acceptRecord() {
          System.out.print("Total No. of Car: ");
          numberOfCar=sc.nextInt();
          System.out.print("Total No. of Truck: ");
          numberOfTruck=sc.nextInt();
          System.out.print("Total No. of MotorCycle: ");
          numberOfMotorcycle=sc.nextInt();
          System.out.println("");

  }

  public void setTollRates() {
          System.out.println("Enter the Toll Rate " + "\n");
          System.out.print("Toll Rate of Car ");
          tollOfCar=sc.nextFloat();
          System.out.print("Toll Rate for Truck: ");
          tollOfTruck=sc.nextFloat();
          System.out.print("Toll Rate for MotorCycle: ");
          tollOfMotorcycle=sc.nextFloat();
  }

  public void  calculateRevenue() {

          totalRevenue =(numberOfCar*tollOfCar)+(numberOfTruck*tollOfTruck)+
(numberOfMotorcycle*tollOfMotorcycle);
    // System.out.println("Total Revenue is: " + totalRevenue);
```

```
  }

  public void printRecord() {
            int vehicle= numberOfCar+numberOfTruck+numberOfMotorcycle  ;
            System.out.println("Total No. Of vehicle: " + vehicle);
          System.out.println("Total revenue collected: " + totalRevenue );
  }




}

public class TollBoothRevenueManager {
public static void main(String[] args) {
          Scanner sc = new Scanner(System.in);
           TollRevenue ref1 = new  TollRevenue();

           ref1.acceptRecord();
           ref1.setTollRates();
           ref1.calculateRevenue();
           ref1.printRecord();
           sc.close();

}
}
```

Output:

```
Total No. of Car: 50
Total No. of Truck: 70
Total No. of MotorCycle: 200

Enter the Toll Rate

Toll Rate of Car 50
Toll Rate for Truck: 80
Toll Rate for MotorCycle: 10
Total No. Of vehicle: 320
Total revenue collected: 10100.0
```