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.

```
import java.util.Scanner;


class Loan{

    private double monthlyPayment;

    private double principal ;

    private double monthlyInterestRate;

    private int numberOfMonths;

    private double annualInterestRate;

    private int loanTerm;

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

```
        public void accpet() {

                this.principal=sc.nextDouble();

                this.annualInterestRate=sc.nextDouble();

                this.loanTerm=sc.nextInt();

        }

        public void calculateMonthlyPayment() {

                monthlyInterestRate=(annualInterestRate*100)/12;

                numberOfMonths=loanTerm*12;

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

        }

        public void printRecord() {

                System.out.println(monthlyPayment);

        }

}

public class Monthly {


        public static void main(String[] args) {


      Loan l=new Loan();

      l.accpet();

      l.calculateMonthlyPayment();

      l.printRecord();

      Loan.sc.close();

        }
```

}

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.

```
class Loan{

        private double principal ;

        private double annualInterestRate;

        private int loanTerm;

        private int numberOfCompounds;

        private double futureValue;

        private double totalIntreast;

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

        public void accpet() {

                this.principal=sc.nextDouble();

                this.annualInterestRate=sc.nextDouble();

                this.loanTerm=sc.nextInt();
                this.numberOfCompounds=sc.nextInt();

        }

        public void calculateMonthlyPayment() {


futureValue=principal*(1+annualInterestRate/Math.pow(numberOfCompoun
ds, numberOfCompounds*loanTerm));

                totalIntreast=futureValue-principal;

        }

        public void printRecord() {

                System.out.println(totalIntreast);

        }

}

public class Monthly {



        public static void main(String[] args) {



      Loan l=new Loan();

      l.accpet();

      l.calculateMonthlyPayment();

      l.printRecord();

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

```
import java.util.Scanner;

class Cal{
        private int weight;
        private int height;
        private double bm;
        private String  count;
        static Scanner sc =new Scanner(System.in);
        public void acceptRecord() {
                this.weight=sc.nextInt();
                this.height=sc.nextInt();
        }
        public void bmi() {
                bm=(weight*10000)/Math.pow(height, 2);
        }
        public void classyfy() {
                if(bm<18.5)
                        count="underWeight";
                else if (bm>=18.5||bm<=24.9)
                        count="normal";
                else if(bm>=25||bm<=29.9)
                        count="overweight";
                else if(bm>=30)
                        count="obese";

        }
        public void printRecord() {
                System.out.println(count);
        }
}
```

```
public class Bmi {

        public static void main(String[] args) {

    Cal c=new Cal();
    c.acceptRecord();
    c.bmi();
    c.classyfy();
    c.printRecord();
    Cal.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 - discnountAmout`
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.

```
import java.util.Scanner;

class Discount{
        private double price;
        private double discountPercentage;
        private double discountAmmount;
        private double finalPrice;
        static Scanner sc=new Scanner(System.in);
        public void acceptRecord() {
                this.price=sc.nextDouble();
                this.discountPercentage=sc.nextDouble();
        }
        public void discount() {
                discountAmmount=price*discountPercentage;
                finalPrice=price-discountAmmount;
        }
        public void printRecord() {
                System.out.println(finalPrice);
        }
}
public class Program {
 public static void main(String[] args) {
		// TODO Auto-generated method stub
	Discount d=new Discount();
	d.acceptRecord();
	d.discount();
	d.printRecord();
	Discount.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

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

```
import java.util.Scanner;
class Toll{
	private double carAmmount;
	private double truckAmmount;
	private double motAmmount;
	private double carNumber;
	private double truckNumber;
	private double motNumber;
	private double finalToll;
	static Scanner sc=new Scanner(System.in);
	public void acceptRecord() {
		this.carAmmount=sc.nextDouble();
		this.truckAmmount=sc.nextDouble();
		this.motAmmount=sc.nextDouble();
		this.carNumber=sc.nextDouble();
		this.truckNumber=sc.nextDouble();
```

```
                this.motNumber=sc.nextDouble();
        }
        public void toll() {

finalToll=carAmmount*carNumber+truckAmmount*truckNumber+motAmmount*motNumb
er;
        }
        public void printRecord() {
                System.out.println(finalToll);
        }
}
public class Program {

        public static void main(String[] args) {
                // TODO Auto-generated method stub
     Toll t=new Toll();
     t.acceptRecord();
     t.toll();
     t.printRecord();
     Toll.sc.close();
        }

}
```