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.

Code:

```
import java.util.Scanner;

class LoanAmortizationCalculator{

	private float loanAmount;
	private float annualInterestRate;
	private int loanTerm;
	private float monthlyPayment;
	private float monthlyInterestRate;
	private int numberOfMonths;

	public void acceptRecord()
	{
		Scanner sc=new Scanner(System.in);
		System.out.println("Enter loanAmount: ");
		this.loanAmount=sc.nextFloat();
		System.out.println("Enter annualInterestRate: ");
		this.annualInterestRate=sc.nextFloat();
		System.out.println("Enter Loan Term (year): ");
		this.loanTerm = sc.nextInt();
		sc.close();
	}
	
	public void calculateMonthlyPayment()
	{
		monthlyInterestRate= annualInterestRate/12/100;
		System.out.println("monthlyInterestRate is: "+monthlyInterestRate);
		numberOfMonths = loanTerm * 12 ;
		System.out.println(" Number of months is: "+numberOfMonths);
		monthlyPayment = (float)(loanAmount * (monthlyInterestRate * 
Math.pow((1+monthlyInterestRate),numberOfMonths))/(Math.pow((1+monthlyInterestRate
),numberOfMonths)-1));
		System.out.println("monthly payment is: "+monthlyPayment);
	}
	public void printRecord()
	{
			System.out.println("loanAmount is: "+loanAmount);
			System.out.println("annualInterestRate: "+annualInterestRate);
			System.out.println("loan term : "+loanTerm);
	}
}
public class Moneymanagement {

	public static void main(String[] args) {
		LoanAmortizationCalculator l1 = new LoanAmortizationCalculator();
		l1.acceptRecord();
		l1.calculateMonthlyPayment();
		l1.printRecord();
	}

}
```

Output:

```
Enter loanAmount:
50000
Enter annualInterestRate:
2.5
Enter Loan Term (year):
5
monthlyInterestRate is: 0.0020833332
 Number of months is: 60
monthly payment is: 887.38074
loanAmount is: 50000.0
annualInterestRate: 2.5
loan term : 5
```

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 CompoundInterestCalculator{
	private float principal;
	private float annualInterestRate;
	private int numberOfCompounds;
	private int years;
	private float futureValue;
	private float totalInterest;

	public void acceptRecord()
	{
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter Principle Amount: ");
		this.principal=sc.nextFloat();
		System.out.println("Enter Annualinterest Rate: ");
		this.annualInterestRate=sc.nextFloat();
		System.out.println("Enter number of times the interest is compounded 
per year : ");
		this.numberOfCompounds=sc.nextInt();
		System.out.println("Enter investment duration (in years): ");
		this.years=sc.nextInt();

	}

	public void calculateFutureValue()
	{
		futureValue
=(float)(principal*Math.pow(1+(annualInterestRate/numberOfCompounds),
numberOfCompounds*years));
	System.out.println("Future Value is: "+futureValue);
	}
	public void printRecoord() {
		totalInterest=futureValue-principal;
		System.out.println("total interest is: "+totalInterest);
	}
}
public class CInterest
{
	public static void main(String[] args)
	{
	CompoundInterestCalculator c =new CompoundInterestCalculator();
	c.acceptRecord();
```

```
       c.calculateFutureValue();
       c.printRecoord();
       }
}
```

Output:

```
Enter Principle Amount:
600000
Enter Annualinterest Rate:
5.5
Enter number of times the interest is compounded per year :
100
Enter investment duration (in years):
6
Future Value is: 5.3655477E19
total interest is: 5.3655477E19
```

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:

```
import java.util.Scanner;

class BMITracker{
       private float weight;
       private float height;
       private float bmiCalculate;

       public void acceptRecord()
       {
              Scanner sc=new Scanner(System.in);
              System.out.println("Entre the weight: ");
              this.weight=sc.nextFloat();
              System.out.println("Entre the height: ");
              this.height=sc.nextFloat();
```

```
		sc.close();
	}
	public void calculateBMI()
	{
		bmiCalculate=weight/(height*height);
		//System.out.println("bmiCalculate");
	}
	public void classifyBMI()
	{
		if(bmiCalculate<18.5)
		{
			System.out.println("Underweight");
		}
		else if(18.5<=bmiCalculate && bmiCalculate<24.9)
		{
			System.out.println("Normal Weight");
		}
		else if(25<=bmiCalculate && bmiCalculate<29.9)
		{
			System.out.println("Overweight");
		}
		else if(bmiCalculate>=30)
		{
			System.out.println("Obese");
		}
		else {
			System.out.println("Invalid input!");
		}
	}
	public void printRecord() {
		System.out.println("BMI is: "+bmiCalculate);
	}
}
public class WeightTracker {

	public static void main(String[] args) {
		BMITracker bmi = new BMITracker();
		bmi.acceptRecord();
		bmi.calculateBMI();
		bmi.classifyBMI();
		bmi.printRecord();

	}

}
```

Output:

```
Entre the weight:
50
Entre the height:
5.5
Underweight
BMI is: 1.6528926
```

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:

```java
import java.util.Scanner;

class DiscountCalculator{
	private float originalPrice;
	private float discountRate;
	private float discountAmount;
	private float finalPrice;

	public void acceptRecord() {
	Scanner sc =new Scanner (System.in);
	System.out.println("Enter the price of item: ");
	this.originalPrice=sc.nextFloat();
	System.out.println("Enter the discount rate: ");
	this.discountRate=sc.nextFloat();
	sc.close();
	}

	public void calculateDiscount() {
		discountAmount =originalPrice*(discountRate/100);
		finalPrice=originalPrice-discountAmount;
	}

	public void printRecord()
	{
		System.out.println("Discount Amount is: "+discountAmount);
		System.out.println("final price is: "+finalPrice);
	}
}
public class Discount {

	public static void main(String[] args) {
		DiscountCalculator dc = new DiscountCalculator();
		dc.acceptRecord();
		dc.calculateDiscount();
		dc.printRecord();
	}
}
```

Output:

```
456
Enter the discount rate:
```

```
20
Discount Amount is: 91.200005
final price is: 364.8
```

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.

Code:

```
public class Toll {

    private double carRate;
    private double truckRate;
    private double motorcycleRate;
    private int carCount;
    private int truckCount;
    private int motorcycleCount;

    public Toll() {
        this.carRate = 50.00;
        this.truckRate = 100.00;
        this.motorcycleRate = 30.00;
        this.carCount = 0;
        this.truckCount = 0;
        this.motorcycleCount = 0;
    }

    public void setTollRates(double carRate, double truckRate, double
motorcycleRate) {
        this.carRate = carRate;
        this.truckRate = truckRate;
        this.motorcycleRate = motorcycleRate;
    }
    public void acceptRecord(int carCount, int truckCount, int motorcycleCount) {
        this.carCount += carCount;
        this.truckCount += truckCount;
        this.motorcycleCount += motorcycleCount;
    }

    public double calculateRevenue() {
        return (carCount * carRate) + (truckCount * truckRate) + (motorcycleCount
* motorcycleRate);
    }

    public void printRecord() {
        int totalVehicles = carCount + truckCount + motorcycleCount;
        double totalRevenue = calculateRevenue();
        System.out.printf("Total number of vehicles: %d\n", totalVehicles);
        System.out.printf("Total revenue collected: ₹%.2f\n", totalRevenue);
    }



    public static void main(String[] args) {
        Toll tollBooth = new Toll();

        tollBooth.setTollRates(55.00, 130.00, 40.00);

        tollBooth.acceptRecord(10, 5, 8);
        tollBooth.printRecord();
        tollBooth.acceptRecord(7, 3, 4);
        tollBooth.printRecord();
    }

}
```

output:

```
Total number of vehicles: 23
Total revenue collected: ₹1520.00
Total number of vehicles: 37
Total revenue collected: ₹2455.00
```