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:

- Accept the principal amount (loan amount), annual interest rate, and loan term (in years) from the user.
- 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
- 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.

-->>

```
package org.example.demo1;

import java.util.Scanner;

class LoanAmortizationCalculator {
     private double principal;
     private double annualInterestRate;
     private int loanTerm;

     public double getPrincipal() {
          return principal;
     }

     public void setPrincipal(double principal) {
          this.principal = principal;
     }

     public double getAnnualInterestRate() {
          return annualInterestRate;
     }

     public void setAnnualInterestRate(double
annualInterestRate) {
          this.annualInterestRate = annualInterestRate;
     }

     public int getLoanTerm() {
          return loanTerm;
     }

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

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

          System.out.print("Enter the principal amount:
");
          setPrincipal(scanner.nextDouble());

          System.out.print("Enter the annual interest
rate: ");
          setAnnualInterestRate(scanner.nextDouble());

          System.out.print("Enter the loan term (in
years): ");
          setLoanTerm(scanner.nextInt());

          scanner.close();
     }

     public double calculateMonthlyPayment() {

          double monthlyInterestRate =
getAnnualInterestRate() / 12 / 100;
          int numberOfMonths = getLoanTerm() * 12;
          double monthlyPayment = getPrincipal()
                   * (monthlyInterestRate * Math.pow(1 +
monthlyInterestRate, numberOfMonths))
                   / (Math.pow(1 + monthlyInterestRate,
numberOfMonths) - 1);

          return monthlyPayment;
     }

     public void printRecord(double monthlyPayment) {
          int numberOfMonths = getLoanTerm() * 12;
          double totalPayment = monthlyPayment *
numberOfMonths;

          System.out.printf("Monthly Payment: " +
monthlyPayment);
          System.out.printf("Total Amount Paid Over the
Life of the Loan: " + totalPayment);
     }

     public static void main(String[] args) {
          LoanAmortizationCalculator loanCalculator =
new LoanAmortizationCalculator();
```

```
        loanCalculator.acceptRecord();

        double monthlyPayment =
loanCalculator.calculateMonthlyPayment();
        loanCalculator.printRecord(monthlyPayment);
    }
}
```

2. Compound Interest Calculator for Investment

Develop a system to compute the future value of an investment with compound interest. The system should:

- 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.
- 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`
- 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 org.example.demo1;


import java.util.Scanner;


public class CompoundInterestCalculator {
     private double principal;
     private double annualInterestRate;
     private int numberOfCompounds;
     private int years;

     public double getPrincipal() {
          return principal;
     }

     public void setPrincipal(double principal) {
          this.principal = principal;
     }

     public double getAnnualInterestRate() {
          return annualInterestRate;
     }

     public void setAnnualInterestRate(double
annualInterestRate) {
          this.annualInterestRate = annualInterestRate;
     }

     public int getNumberOfCompounds() {
         return numberOfCompounds;
     }

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

     public int getYears() {
         return years;
     }

     public void setYears(int years) {
         this.years = years;
     }

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

         System.out.print("Enter the initial investment
 amount : ");
         setPrincipal(scanner.nextDouble());

         System.out.print("Enter the annual interest
 rate : ");
         setAnnualInterestRate(scanner.nextDouble());

         System.out.print("Enter amount compounded: ");
         setNumberOfCompounds(scanner.nextInt());

         System.out.print("Enter the investment
 duration (in years): ");
         setYears(scanner.nextInt());

         scanner.close();
     }

     public double calculateFutureValue() {

         double rateAsDecimal = getAnnualInterestRate()
 / 100;

         double futureValue = getPrincipal()
                   * Math.pow((1 + rateAsDecimal /
 getNumberOfCompounds()), getNumberOfCompounds() *
 getYears());

         return futureValue;
     }
     public void printRecord(double futureValue) {

         double totalInterest = futureValue -
 getPrincipal();

         System.out.printf("Future Value: ",
 +futureValue);
         System.out.printf("Total Interest Earned: " +
 totalInterest);
     }

     public static void main(String[] args) {

         CompoundInterestCalculator calculator = new
 CompoundInterestCalculator();

         calculator.acceptRecord();

         double futureValue =
 calculator.calculateFutureValue();

         calculator.printRecord(futureValue);
     }
 }
```

3. BMI (Body Mass Index) Tracker

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

- Accept weight (in kilograms) and height (in meters) from the user.
- Calculate the BMI using the formula:
    - **BMI Calculation:** `BMI = weight / (height * height)`
- 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
- 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 org.example.demo1;

import java.util.*;

class BMITracker {

    private double weight;
    private double 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 void acceptRecord() {
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter weight in kg: ");
        setWeight(sc.nextDouble());

        System.out.println("Enter height in cms: ");
        setHeight(sc.nextDouble());

        sc.close();
    }

    public double calculateBMI() {
        double bmi = getWeight() / (getHeight() *
getHeight());
        return bmi;
    }

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

```
	}

	public void printRecord() {
		double bmi = calculateBMI();
		String classification = classifyBMI();

		System.out.printf("Your BMI is: " + bmi);
		System.out.println("BMI Classification: " + 
classification);
	}

	public static void main(String[] args) {
		BMITracker bmiTracker = new BMITracker();

		bmiTracker.acceptRecord();

		bmiTracker.printRecord();
	}
}
```

4. Discount Calculation for Retail Sales

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

- Accept the original price of an item and the discount percentage from the user.
- 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`
- 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 org.example.demo1;
```

```
import java.util.Scanner;

public class DiscountCalculator {

    private double originalPrice;
    private double 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 void acceptRecord() {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the original price of
the item: ");
        setOriginalPrice(scanner.nextDouble());

        System.out.print("Enter the discount rate %:
");
        setDiscountRate(scanner.nextDouble());

        scanner.close();
    }
    public double calculateDiscountAmount() {
        return getOriginalPrice() * (getDiscountRate()
/ 100);
    }

    public double calculateFinalPrice() {
        double discountAmount =
calculateDiscountAmount();
        return getOriginalPrice() - discountAmount;
    }

    public void printRecord() {
        double discountAmount =
calculateDiscountAmount();
        double finalPrice = calculateFinalPrice();

        System.out.printf("Discount Amount: ",
discountAmount);
        System.out.printf("Final Price: " +
finalPrice);
    }

    public static void main(String[] args) {
        DiscountCalculator calculator = new
DiscountCalculator();

        calculator.acceptRecord();

        calculator.printRecord();
    }
}
```

5. Toll Booth Revenue Management

Develop a system to simulate a toll booth for collecting revenue. The system should:

- Allow the user to set toll rates for different vehicle types: Car, Truck, and Motorcycle.

- Accept the number of vehicles of each type passing through the toll booth.
- Calculate the total revenue based on the toll rates and number of vehicles.
- 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 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 org.example.demo1;

import java.util.Scanner;

class TollBoothRevenueManager {

    private double carTollRate;
    private double truckTollRate;
    private double motorcycleTollRate;

    private int carCount;
    private int truckCount;
    private int motorcycleCount;

    public double getCarTollRate() {
        return carTollRate;
    }

    public void setCarTollRate(double carTollRate) {
        this.carTollRate = carTollRate;
    }
```

```
     }

     public double getTruckTollRate() {
          return truckTollRate;
     }

     public void setTruckTollRate(double truckTollRate)
{
          this.truckTollRate = truckTollRate;
     }

     public double getMotorcycleTollRate() {
          return motorcycleTollRate;
     }

     public void setMotorcycleTollRate(double
motorcycleTollRate) {
          this.motorcycleTollRate = motorcycleTollRate;
     }

     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 getMotorcycleCount() {
          return motorcycleCount;
     }
```

```
    public void setMotorcycleCount(int motorcycleCount)
{
        this.motorcycleCount = motorcycleCount;
    }

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

        System.out.print("Enter the toll rate for
Cars: ");
        setCarTollRate(scanner.nextDouble());

        System.out.print("Enter the toll rate for
Trucks: ");
        setTruckTollRate(scanner.nextDouble());

        System.out.print("Enter the toll rate for
Motorcycles: ");
        setMotorcycleTollRate(scanner.nextDouble());

        System.out.print("Enter the number of Cars
passed: ");
        setCarCount(scanner.nextInt());

        System.out.print("Enter the number of Trucks
passed: ");
        setTruckCount(scanner.nextInt());

        System.out.print("Enter the number of
Motorcycles passed: ");
        setMotorcycleCount(scanner.nextInt());

        scanner.close();
    }

    public double calculateTotalRevenue() {
        double carRevenue = getCarTollRate() *
getCarCount();
          double truckRevenue = getTruckTollRate() *
getTruckCount();
          double motorcycleRevenue =
getMotorcycleTollRate() * getMotorcycleCount();

          return carRevenue + truckRevenue +
motorcycleRevenue;
     }

     public int calculateTotalVehicles() {
          return getCarCount() + getTruckCount() +
getMotorcycleCount();
     }

     public void printRecord() {
          int totalVehicles = calculateTotalVehicles();
          double totalRevenue = calculateTotalRevenue();

          System.out.println("Total Vehicles Passed: " +
totalVehicles);
          System.out.printf("Total Revenue Collected: "
+ totalRevenue);
     }

     public static void main(String[] args) {
          TollBoothRevenueManager tbr = new
TollBoothRevenueManager();
          tbr.acceptRecord();
          tbr.printRecord();
     }
}
```