

**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:

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:**
    - $\text{monthlyPayment} = \text{principal} * (\text{monthlyInterestRate} * (1 + \text{monthlyInterestRate})^{\text{numberOfMonths}}) / ((1 + \text{monthlyInterestRate})^{\text{numberOfMonths}} - 1)$
    - Where  $\text{monthlyInterestRate} = \text{annualInterestRate} / 12 / 100$  and  $\text{numberOfMonths} = \text{loanTerm} * 12$
    - 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 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.

## 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:**
    - $\text{futureValue} = \text{principal} * (1 + \text{annualInterestRate} / \text{numberOfCompounds})^{\text{numberOfCompounds} * \text{years}}$
  - **Total Interest Earned:**  $\text{totalInterest} = \text{futureValue} - \text{principal}$
3. 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.

### 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:**  $BMI = \text{weight} / (\text{height} * \text{height})$
3. Classify the BMI into one of the following categories:
  - Underweight:  $BMI < 18.5$
  - Normal weight:  $18.5 \leq BMI < 24.9$
  - Overweight:  $25 \leq BMI < 29.9$
  - Obese:  $BMI \geq 30$
4. 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.

### 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:**  $\text{discountAmount} = \text{originalPrice} * (\text{discountRate} / 100)$
  - **Final Price Calculation:**  $\text{finalPrice} = \text{originalPrice} - \text{discountAmount}$
3. 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.

### 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 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.