**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.

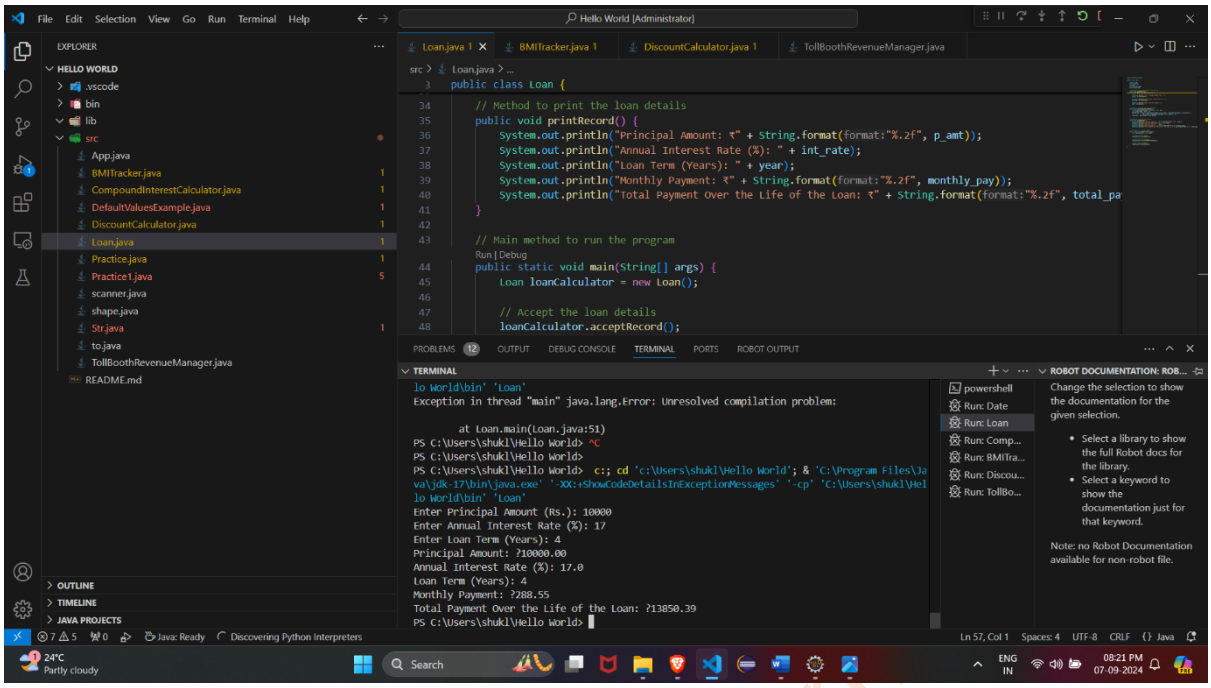
Code:

```
import java.util.Scanner;

public class Loan {

    // Fields for principal amount, interest rate, loan term, monthly payment, and total payment
    double p_amt;
    float int_rate;
    int year;
    double monthly_pay;
    double total_pay;

    // Method to accept user input (loan details)
    public void acceptRecord() {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter Principal Amount (Rs.): ");
        p_amt = sc.nextDouble();

        System.out.print("Enter Annual Interest Rate (%): ");
        int_rate = sc.nextFloat();
```

sandeepkulange@gmail.com

```
        System.out.print("Enter Loan Term (Years): ");
        year = sc.nextInt();
    }

    // Method to calculate the monthly payment using the loan formula
    public void calculateMonthlyPayment() {
        float monthlyInterestRate = int_rate / 12 / 100;
        int no_of_months = year * 12;

        // Using the standard formula for calculating monthly mortgage payments
        monthly_pay = p_amt * (monthlyInterestRate * Math.pow(1 + monthlyInterestRate,
no_of_months)) /
                (Math.pow(1 + monthlyInterestRate, no_of_months) - 1);

        total_pay = monthly_pay * no_of_months;  // Total payment over the loan term
    }

    // Method to display the loan details, monthly payment, and total payment
    public void printRecord() {
        System.out.println("Principal Amount : Rs." + p_amt);
        System.out.println("Annual Interest Rate : " + int_rate + "%");
        System.out.println("Loan Term : " + year + " years");
        System.out.println("Monthly Payment : Rs." + String.format("%.2f", monthly_pay));
        System.out.println("Total Payment : Rs." + String.format("%.2f", total_pay));
    }

    // Main method to drive the program
    public static void main(String[] args) {
        Loan loanCalculator = new Loan();

        // Accept the loan details
        loanCalculator.acceptRecord();

        // Calculate the monthly payment
        loanCalculator.calculateMonthlyPayment();

        // Print the loan details, monthly payment, and total payment
        loanCalculator.printRecord();
    }
}
```

sandeepkulange@gmail.com




## 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 (₹).

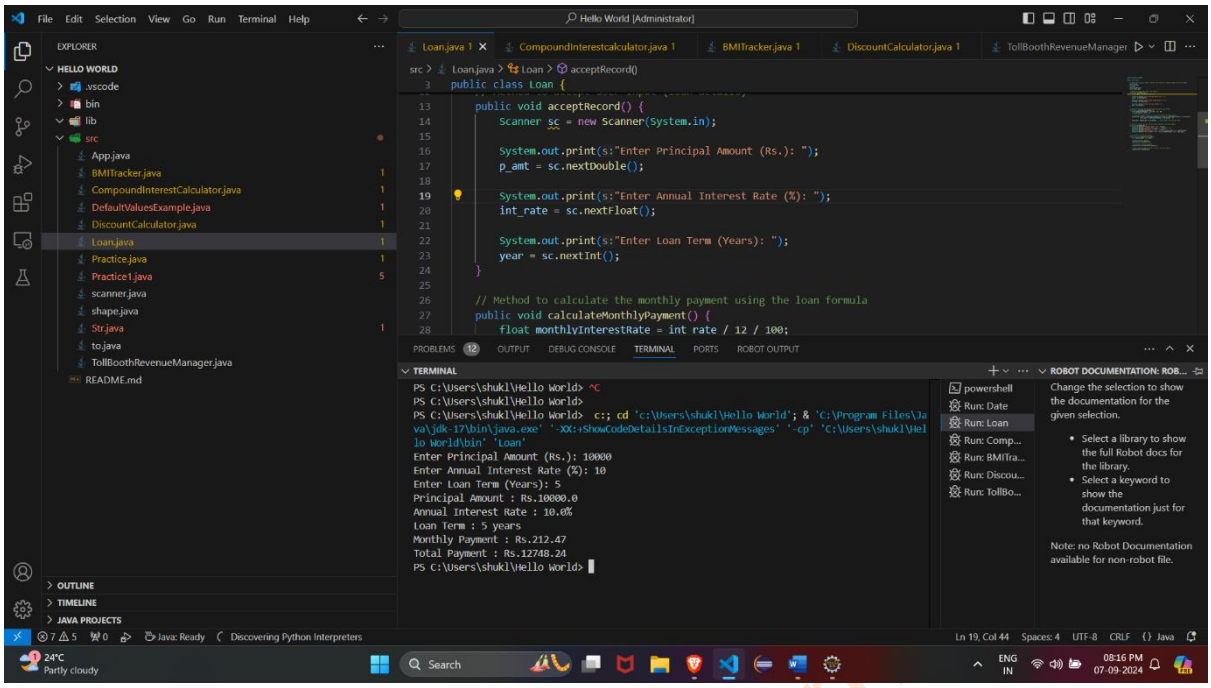
Define class CompoundInterestCalculator with methods acceptRecord , calculateFutureValue, printRecord and test the functionality in main method.

Code:

```
import java.util.Scanner;

public class CompoundInterestCalculator {

    double principal;
    double annualInterestRate;
```

```
    int numberOfCompounds;
    int investmentDuration;
    double futureValue;
    double totalInterest;

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

        System.out.print("Enter Initial Investment Amount (Rs.): ");
        principal = sc.nextDouble();

        System.out.print("Enter Annual Interest Rate (%): ");
        annualInterestRate = sc.nextDouble();

        System.out.print("Enter Number of Times Interest is Compounded Per Year: ");
        numberOfCompounds = sc.nextInt();

        System.out.print("Enter Investment Duration (Years): ");
        investmentDuration = sc.nextInt();
    }

    public void calculateFutureValue() {
        double annualInterestRateDecimal = annualInterestRate / 100;
        futureValue = principal * Math.pow((1 + annualInterestRateDecimal /
numberOfCompounds), numberOfCompounds * investmentDuration);
        totalInterest = futureValue - principal;
    }

    public void printRecord() {
        System.out.println("Future Value of Investment: ₹" +futureValue);
        System.out.println("Total Interest Earned: ₹" +totalInterest);
    }

    public static void main(String[] args) {
        CompoundInterestCalculator cic = new CompoundInterestCalculator();

        cic.acceptRecord();
        cic.calculateFutureValue();
        cic.printRecord();
    }
}
```

sandeeprulange@gmail.com




## 3. BMI (Body Mass Index) Tracker

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

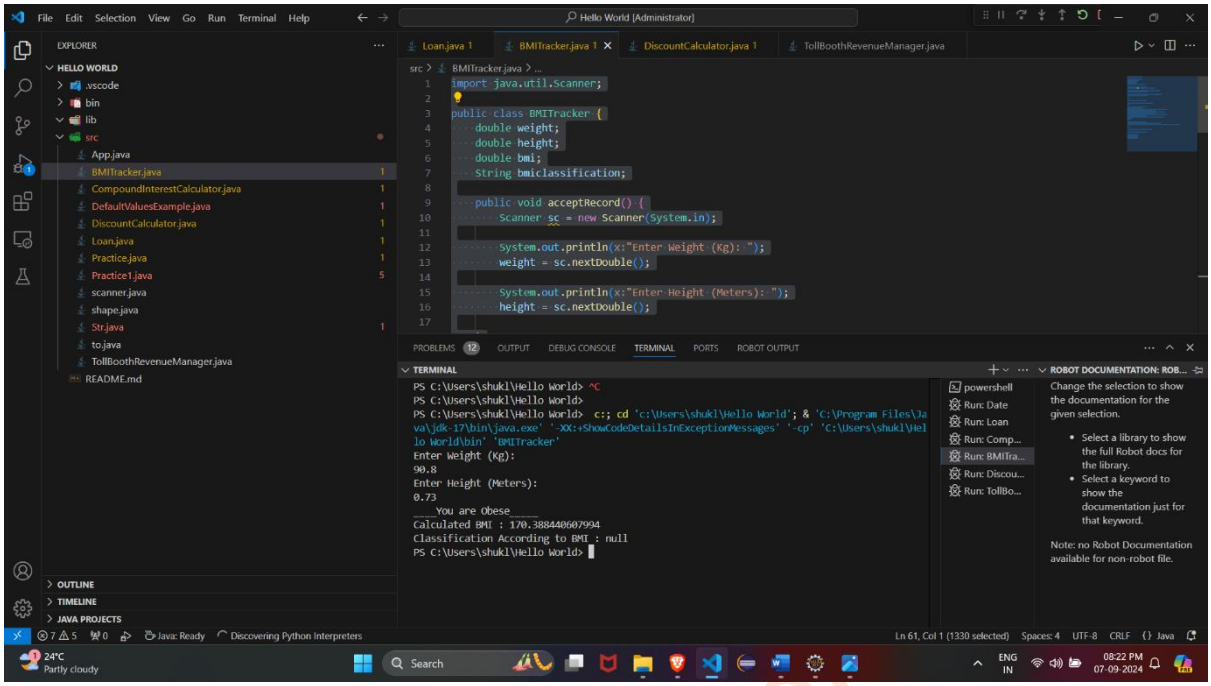
1. Accept weight (in kilograms) and height (in meters) from the user.
2. Calculate the BMI using the formula:
   - **BMI Calculation:** `BMI = weight / (height * height)`
3. Classify the BMI into one of the following categories:
   - Underweight: BMI < 18.5
   - Normal weight: 18.5 ≤ BMI < 24.9
   - 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;

public class BMITracker {
    double weight;
    double height;
    double bmi;
    String bmiclassification;
```

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

        System.out.println("Enter Weight (Kg): ");
        weight = sc.nextDouble();

        System.out.println("Enter Height (Meters): ");
        height = sc.nextDouble();

    }

    public void calculateBMI() {

        bmi = weight / (height * height);
    }

    public void classifyBMI() {
        if (bmi < 18.5) {
            System.out.println("____You are Underweight____");
        } else if (18.5 <= bmi && bmi < 24.9) {
            System.out.println("____You are Normal_____");
        } else if (25 <= bmi && bmi < 29.9) {
            System.out.println("____You are Overweight_____");
        } else if (bmi >= 30) {
            System.out.println("____You are Obese_____");
        }
    }

    public void printRecord() {
        System.out.println("Calculated BMI : " + bmi);
        System.out.println("Classification According to BMI : " + bmiclassification);

    }

public static void main(String args[]){

BMITracker BT = new BMITracker();

BT.acceptRecord();
BT.calculateBMI();
BT.classifyBMI();
BT.printRecord();

}




}
```

sandeepkulange@gmail.com




## 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 (₹).

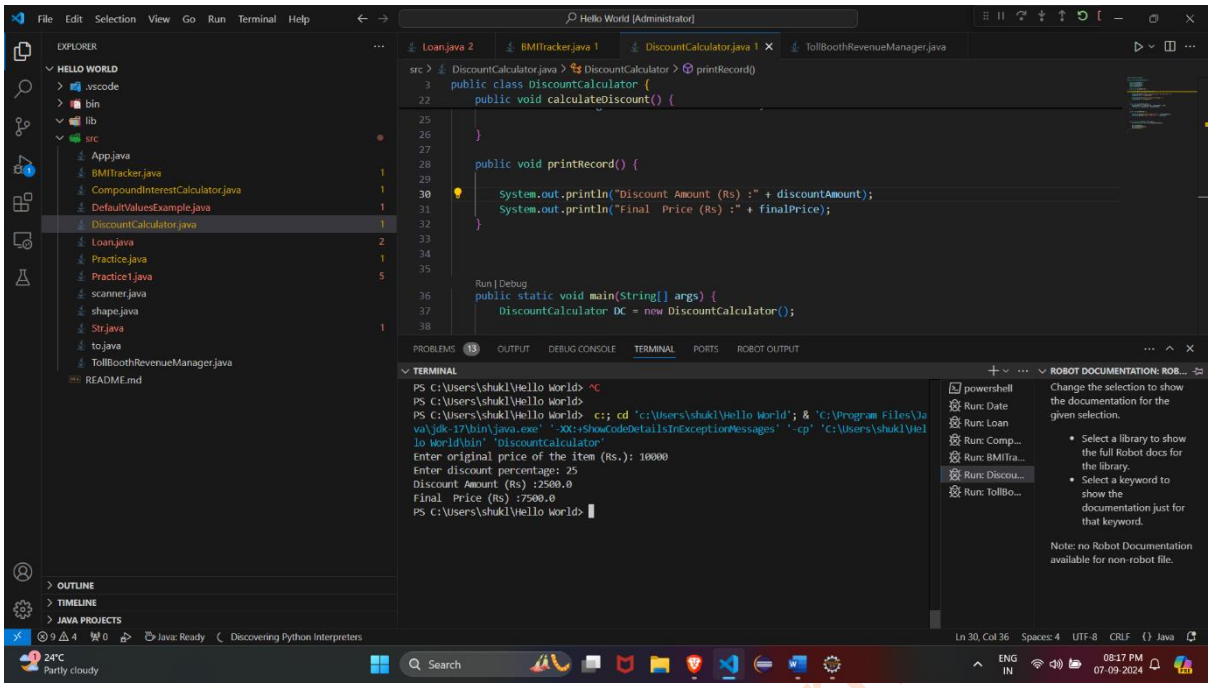
Define class DiscountCalculator with methods acceptRecord, calculateDiscount & printRecord and test the functionality in main method.

Code:

```
import java.util.Scanner;

public class DiscountCalculator {

    double originalPrice;
    double discountRate;
    double discountAmount;
    double finalPrice;
```

```
    public void acceptRecord() {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter original price of the item (Rs.): ");
        originalPrice = sc.nextDouble();

        System.out.print("Enter discount percentage: ");
        discountRate = sc.nextDouble();

    }

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

    }

    public void printRecord() {

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


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

        DC.acceptRecord();
        DC.calculateDiscount();
        DC.printRecord();
    }

}
```

sandeepnilange@gmail.com




## 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.

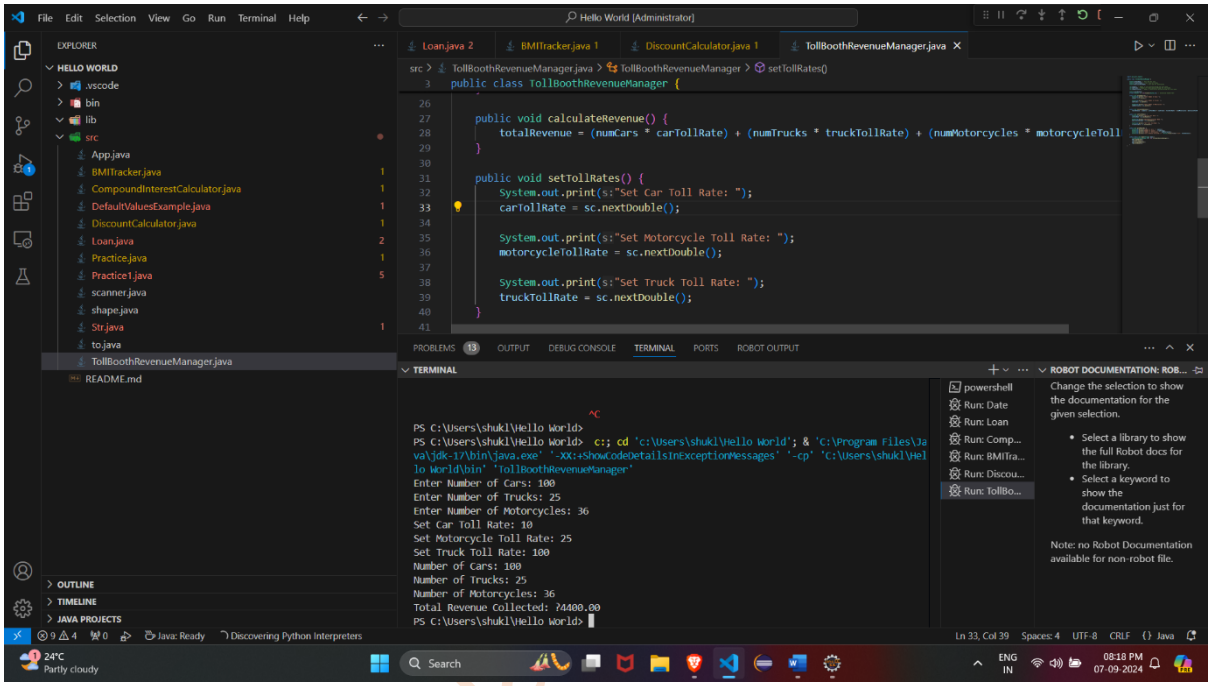
Code:

```
import java.util.Scanner;

public class TollBoothRevenueManager {
```

```
    double carTollRate; // Toll rate for cars.
    double truckTollRate; // Toll rate for trucks.
    double motorcycleTollRate; // Toll rate for motorcycles.

    int numCars; // Number of cars passing through the toll booth.
    int numTrucks; // Number of trucks passing through the toll booth.
    int numMotorcycles; // Number of motorcycles passing through the toll booth.

    double totalRevenue;
    private Scanner sc = new Scanner(System.in); // Initialize Scanner here

    public void acceptRecord() {
        System.out.print("Enter Number of Cars: ");
        numCars = sc.nextInt();

        System.out.print("Enter Number of Trucks: ");
        numTrucks = sc.nextInt();

        System.out.print("Enter Number of Motorcycles: ");
        numMotorcycles = sc.nextInt();
    }

    public void calculateRevenue() {
        totalRevenue = (numCars * carTollRate) + (numTrucks * truckTollRate) +
(numMotorcycles * motorcycleTollRate);
    }

    public void setTollRates() {
        System.out.print("Set Car Toll Rate: ");
        carTollRate = sc.nextDouble();

        System.out.print("Set Motorcycle Toll Rate: ");
        motorcycleTollRate = sc.nextDouble();

        System.out.print("Set Truck Toll Rate: ");
        truckTollRate = sc.nextDouble();
    }

    public void printRecord() {
        System.out.println("Number of Cars: " + numCars);
        System.out.println("Number of Trucks: " + numTrucks);
        System.out.println("Number of Motorcycles: " + numMotorcycles);
        System.out.println("Total Revenue Collected: ₹" + String.format("%.2f", totalRevenue));
    }

    public static void main(String[] args) {
```

sandeepkulange@gmail.com

```
        TollBoothRevenueManager TB = new TollBoothRevenueManager();
        TB.acceptRecord();
        TB.setTollRates();
        TB.calculateRevenue();
        TB.printRecord();
    }
}
```




sandeepkula