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

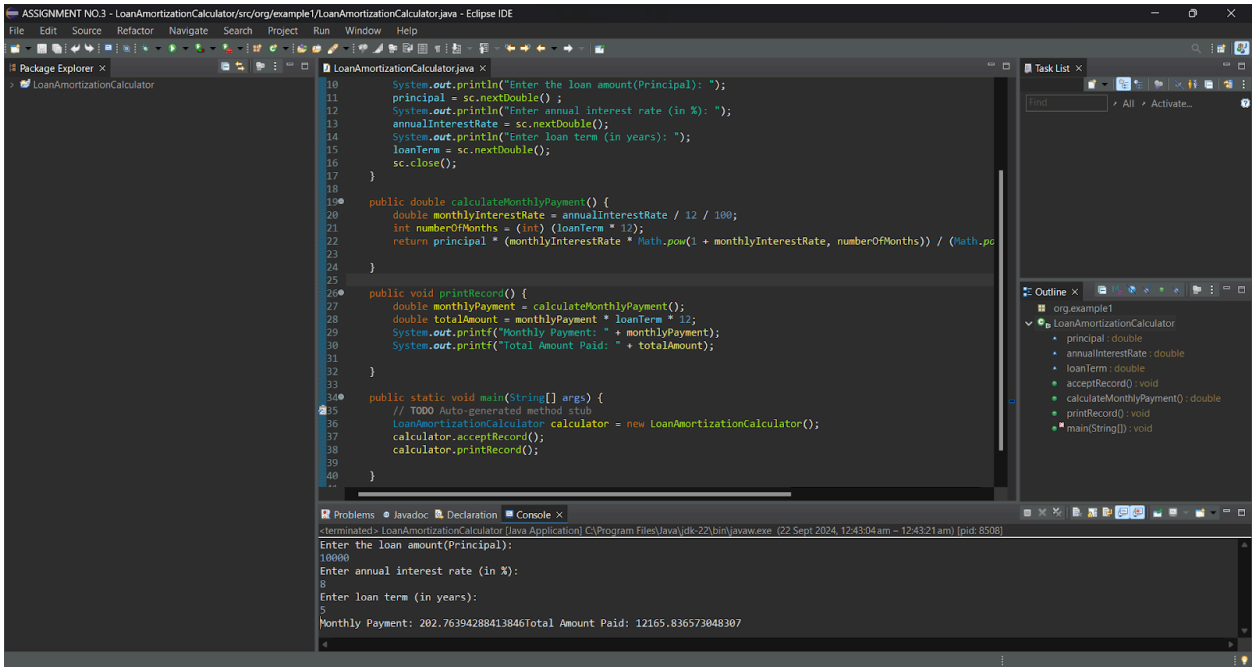
3. Display the monthly payment and the total amount paid over the life of the loan, in Indian Rupees (₹).

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

Program Code:

```
package org.example1;
import java.util.Scanner;
public class LoanAmortizationCalculator {
	double principal, annualInterestRate, loanTerm;

	public void acceptRecord() {
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter the loan amount(Principal): ");
		principal = sc.nextDouble()	;
		System.out.println("Enter annual interest rate (in %): ");
		annualInterestRate = sc.nextDouble();
		System.out.println("Enter loan term (in years): ");
		loanTerm = sc.nextDouble();
		sc.close();
	}

	public double calculateMonthlyPayment() {
		double monthlyInterestRate = annualInterestRate / 12 / 100;
		int numberOfMonths = (int) (loanTerm * 12);
		return principal * (monthlyInterestRate * Math.pow(1 + monthlyInterestRate,
numberOfMonths)) / (Math.pow(1 + monthlyInterestRate, numberOfMonths) - 1);

	}

	public void printRecord() {
		double monthlyPayment = calculateMonthlyPayment();
		double totalAmount = monthlyPayment * loanTerm * 12;
		System.out.printf("Monthly Payment: " + monthlyPayment);
		System.out.printf("Total Amount Paid: " + totalAmount);

	}
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		LoanAmortizationCalculator calculator = new LoanAmortizationCalculator();
		calculator.acceptRecord();
		calculator.printRecord();
	}
}
```

Output:



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.

Program Code:

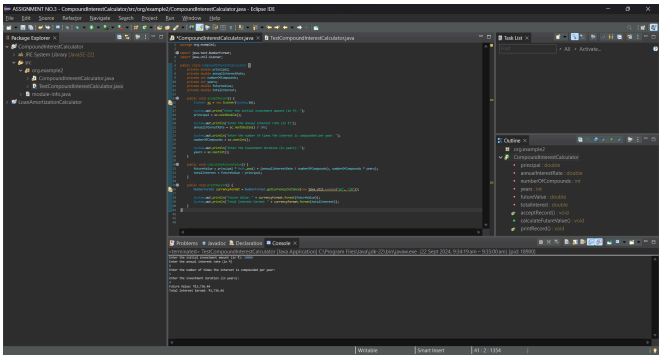
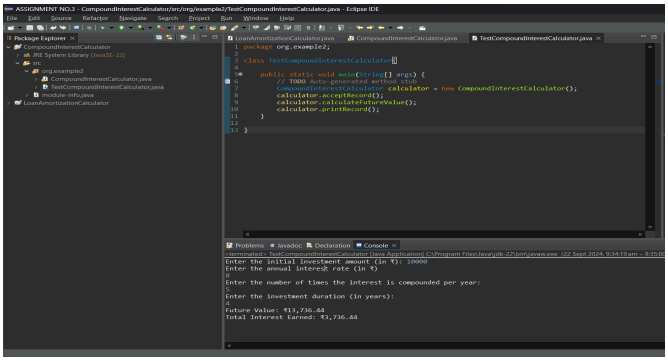
Class 1: CompoundInterestCalculator

```
package org.example2;
import java.text.NumberFormat;
import java.util.Scanner;
public class CompoundInterestCalculator {
	private double principal;
	private double annualInterestRate;
	private int numberOfCompounds;
	private int years;
	private double futureValue;
	private double totalInterest;

	public void acceptRecord() {
		Scanner sc = new Scanner(System.in);
		System.out.print("Enter the initial investment amount (in ₹): ");
		principal = sc.nextDouble();

		System.out.println("Enter the annual interest rate (in ₹)");
		annualInterestRate = sc.nextDouble() / 100;

		System.out.println("Enter the number of times the interest is compounded per
year: ");
		numberOfCompounds = sc.nextInt();

		System.out.println("Enter the investment duration (in years): ");
		years = sc.nextInt();
	}

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

	public void printRecord() {
		NumberFormat currencyFormat = NumberFormat.getCurrencyInstance(new
java.util.Locale("en", "IN"));

		System.out.println("Future Value: " + currencyFormat.format(futureValue));
		System.out.println("Total Interest Earned: " +
currencyFormat.format(totalInterest));
	}
}
```

Class 2: TestCompoundInterestCalculator

```
package org.example2;
class TestCompoundInterestCalculator{

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		CompoundInterestCalculator calculator = new CompoundInterestCalculator();
		calculator.acceptRecord();
		calculator.calculateFutureValue();
		calculator.printRecord();
	}
}
```

Output:

Class 1	Class 2
	

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.

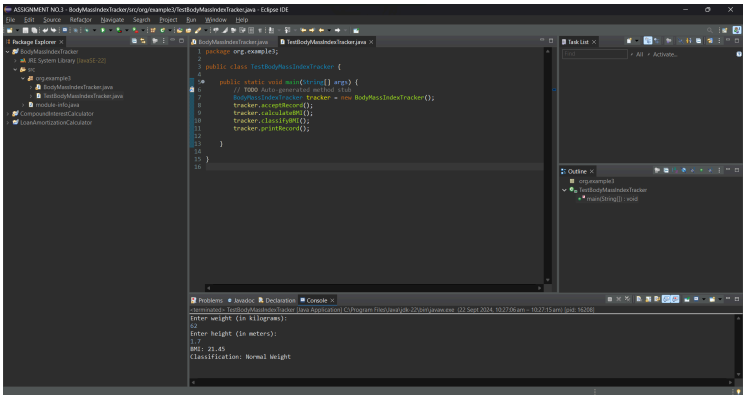
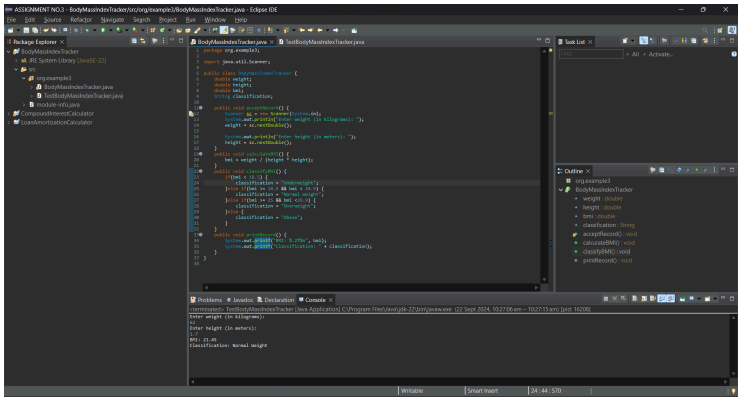
Program Code:
Class 1: BodyMassIndexTracker

```
package org.example3;
import java.util.Scanner;
public class BodyMassIndexTracker {
	double weight;
	double height;
	double bmi;
	String classification;

	public void acceptRecord() {
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter weight (in kilograms): ");
		weight = sc.nextDouble();

		System.out.println("Enter height (in meters): ");
		height = sc.nextDouble();
	}
	public void calculateBMI() {
		bmi = weight / (height * height);
	}
	public void classifyBMI() {
		if(bmi < 18.5) {
			classification = "Underweight";
		}else if(bmi >= 18.5 && bmi < 24.9) {
			classification = "Normal Weight";
		}else if(bmi >= 25 && bmi <29.9) {
			classification = "Overweight";
		}else {
			classification = "Obese";
		}
	}
	public void printRecord() {
		System.out.printf("BMI: %.2f%n", bmi);
		System.out.printf("Classification: " + classification);
	}
}
```

Class 2:TestBodyMassIndexTracker

```
package org.example3;
public class TestBodyMassIndexTracker {
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		BodyMassIndexTracker tracker = new BodyMassIndexTracker();
		tracker.acceptRecord();
		tracker.calculateBMI();
		tracker.classifyBMI();
		tracker.printRecord();
	}
}
```

Output:

Class 1	Class 2
	

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.

Program Code:

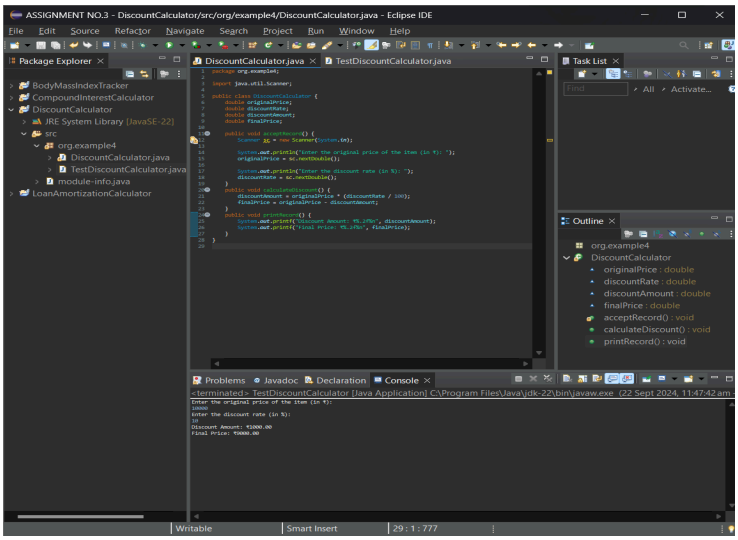
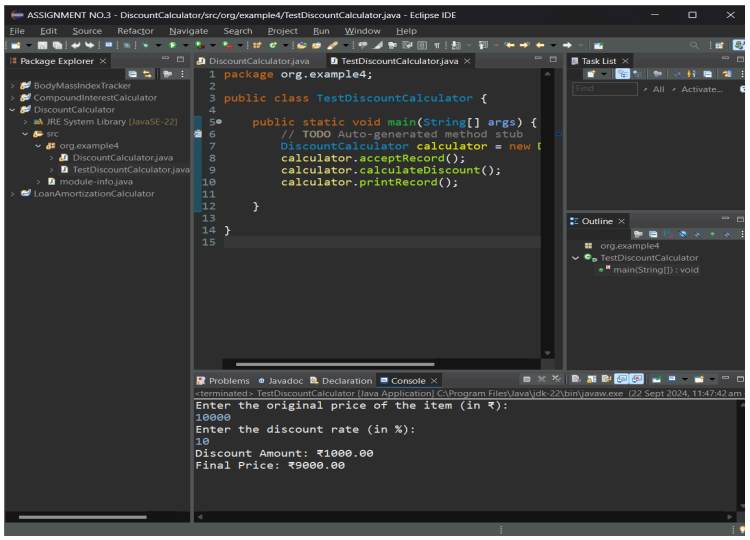
Class 1: DiscountCalculator

```
package org.example4;
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.println("Enter the original price of the item (in ₹): ");
		originalPrice = sc.nextDouble();

		System.out.println("Enter the discount rate (in %): ");
		discountRate = sc.nextDouble();
	}
	public void calculateDiscount() {
		discountAmount = originalPrice * (discountRate / 100);
		finalPrice = originalPrice - discountAmount;
	}
	public void printRecord() {
		System.out.printf("Discount Amount: ₹%.2f%n", discountAmount);
		System.out.printf("Final Price: ₹%.2f%n", finalPrice);
	}
}
```

Class 2: TestDiscountCalculator

```
package org.example4;
public class TestDiscountCalculator {
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		DiscountCalculator calculator = new DiscountCalculator();
		calculator.acceptRecord();
		calculator.calculateDiscount();
		calculator.printRecord();
	}
}
```

Output:

Class 1	Class 2
	

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.

Program Code:

Class 1: TollBoothRevenueManager

```
package org.example5;
import java.util.Scanner;
public class TollBoothRevenueManager {
	double carTollRate, truckTollRate, motorcycleTollRate;
	int carCount, truckCount, motorcycleCount;
	double totalRevenue;

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

		System.out.println("Enter the toll rate for Cars (₹): ");
		carTollRate = sc.nextDouble();

		System.out.println("Enter the toll rate for Trucks (₹): ");
		truckTollRate = sc.nextDouble();

		System.out.println("Enter the toll rate for Motorcycles (₹): ");
		motorcycleTollRate = sc.nextDouble();
	}
	public void acceptRecord() {
		Scanner sc = new Scanner(System.in);

		System.out.println("Enter the number of Cars: ");
		carCount = sc.nextInt();

		System.out.println("Enter the number of Trucks: ");
		truckCount = sc.nextInt();

		System.out.println("Enter the number of Motorcycles: ");
		motorcycleCount = sc.nextInt();
	}
	public void calculateRevenue() {
		totalRevenue = (carCount * carTollRate) + (truckCount * truckTollRate) +
(motorcycleCount * motorcycleTollRate);
	}
	public void printRecord() {
      int totalVehicles = carCount + truckCount + motorcycleCount;
		System.out.println("Total number of vehicles: " + totalVehicles);
      System.out.printf("Total revenue collected: ₹%.2f%n", totalRevenue);
	}
}
```

Class 2: TestTollBoothRevenueManager

```
package org.example5;
public class TestTollBoothRevenueManager {
	public static void main(String[] args) {
		// TODO Auto-generated method stub
		TollBoothRevenueManager manager = new TollBoothRevenueManager();
		manager.setTollRates();
		manager.acceptRecord();
		manager.calculateRevenue();
		manager.printRecord();
	}
}
```

Output:

Class 1	Class 2