**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.

import java.util.Scanner;

class LoanAmortizationCalculator {

double principal;

double annualInterestRate;

int loanTerm;

void acceptRecord() {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter principal amount (₹): ");

principal = scanner.nextDouble();

System.out.print("Enter annual interest rate (%): ");

annualInterestRate = scanner.nextDouble();

System.out.print("Enter loan term (years): ");

loanTerm = scanner.nextInt();

}

void calculateMonthlyPayment() {

double monthlyInterestRate = annualInterestRate / 12 / 100;

int numberOfMonths = loanTerm \* 12;

double monthlyPayment = principal \* (monthlyInterestRate \* Math.pow(1 + monthlyInterestRate, numberOfMonths)) / (Math.pow(1 + monthlyInterestRate, numberOfMonths) - 1);

System.out.println("Monthly Payment: ₹" + monthlyPayment);

System.out.println("Total Amount Paid: ₹" + (monthlyPayment \* numberOfMonths));

}

void printRecord() {

System.out.println("Principal Amount: ₹" + principal);

System.out.println("Annual Interest Rate: " + annualInterestRate + "%");

System.out.println("Loan Term: " + loanTerm + " years");

}

}

public class LoanAmortizationCalculatorTest {

public static void main(String[] args) {

LoanAmortizationCalculator calculator = new LoanAmortizationCalculator();

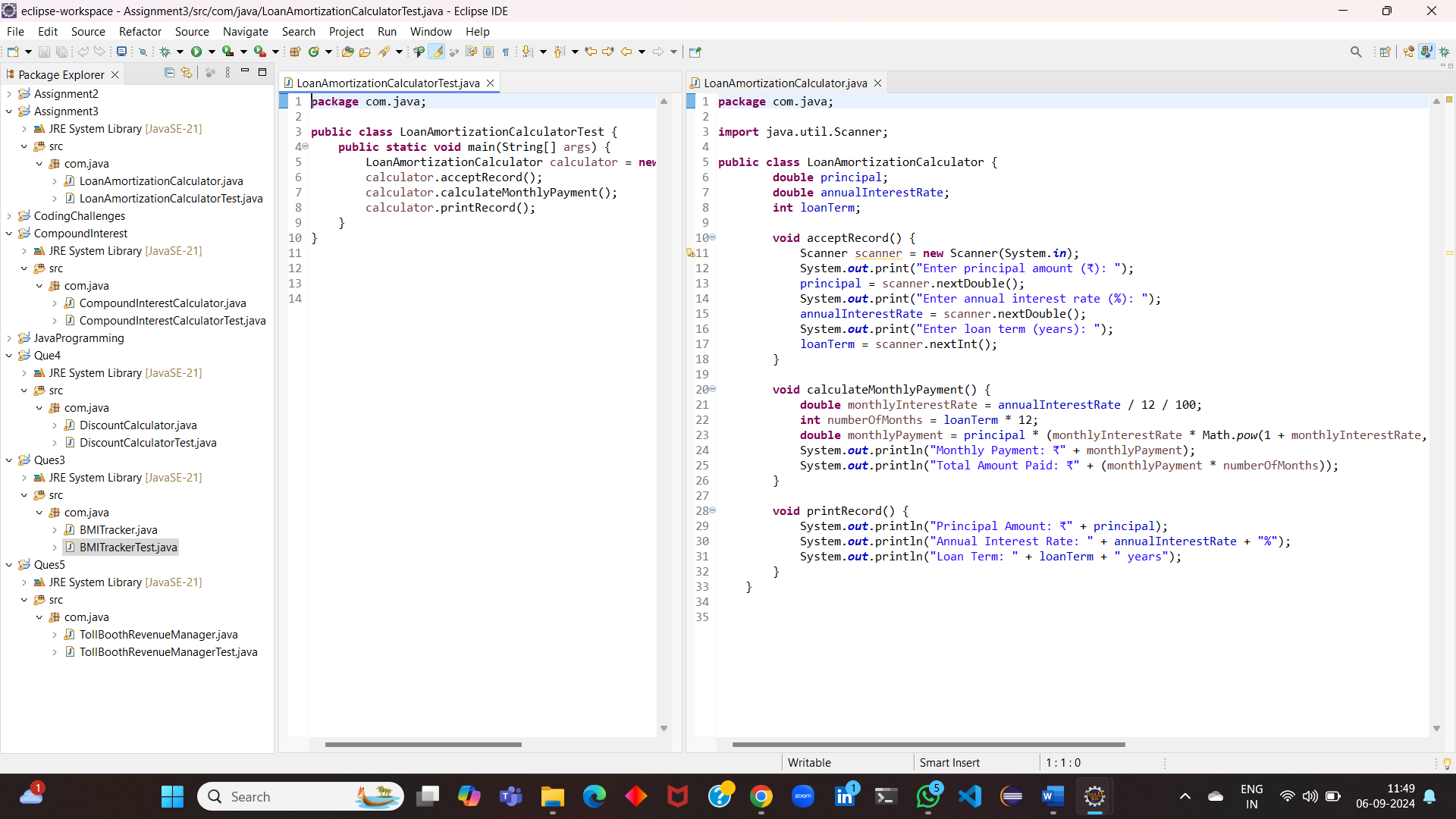
calculator.acceptRecord();

calculator.calculateMonthlyPayment();

calculator.printRecord();

}

}



**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.

import java.util.Scanner;

class CompoundInterestCalculator {

double principal;

double annualInterestRate;

int numberOfCompounds;

int years;

void acceptRecord() {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter initial investment amount (₹): ");

principal = scanner.nextDouble();

System.out.print("Enter annual interest rate (%): ");

annualInterestRate = scanner.nextDouble();

System.out.print("Enter number of times interest is compounded per year: ");

numberOfCompounds = scanner.nextInt();

System.out.print("Enter investment duration (years): ");

years = scanner.nextInt();

}

void calculateFutureValue() {

double futureValue = principal \* Math.pow(1 + annualInterestRate / numberOfCompounds, numberOfCompounds \* years);

double totalInterest = futureValue - principal;

System.out.println("Future Value: ₹" + futureValue);

System.out.println("Total Interest Earned: ₹" + totalInterest);

}

void printRecord() {

System.out.println("Initial Investment: ₹" + principal);

System.out.println("Annual Interest Rate: " + annualInterestRate + "%");

System.out.println("Number of Compounds per Year: " + numberOfCompounds);

System.out.println("Investment Duration: " + years + " years");

}

}

public class CompoundInterestCalculatorTest {

public static void main(String[] args) {

CompoundInterestCalculator calculator = new CompoundInterestCalculator();

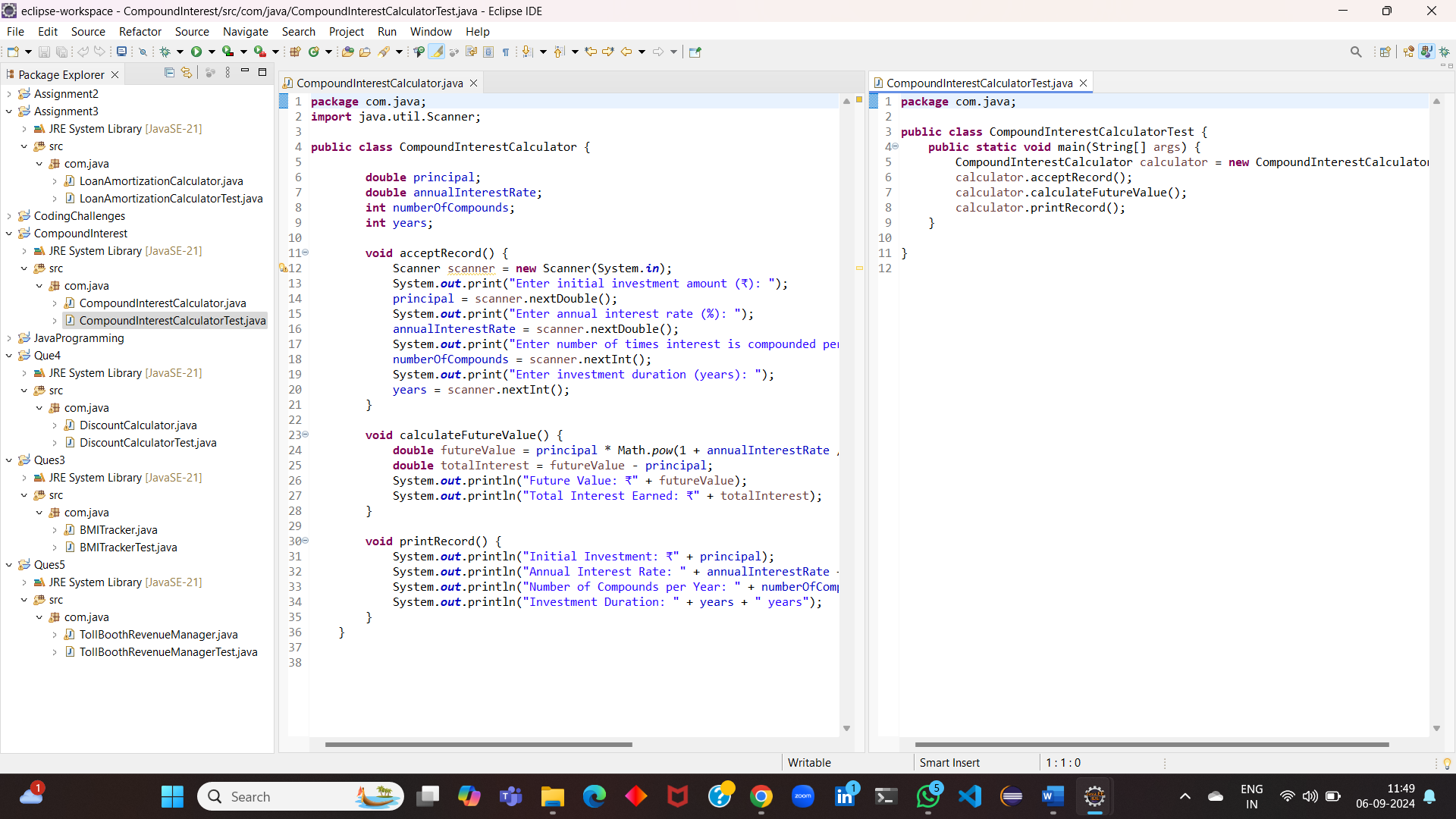
calculator.acceptRecord();

calculator.calculateFutureValue();

calculator.printRecord();

}

}



**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.

import java.util.Scanner;

class BMITracker {

double weight;

double height;

void acceptRecord() {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter weight (kg): ");

weight = scanner.nextDouble();

System.out.print("Enter height (m): ");

height = scanner.nextDouble();

}

double calculateBMI() {

return weight / (height \* height);

}

String classifyBMI(double bmi) {

if (bmi < 18.5) {

return "Underweight";

} else if (bmi < 25) {

return "Normal weight";

} else if (bmi < 30) {

return "Overweight";

} else {

return "Obese";

}

}

void printRecord() {

double bmi = calculateBMI();

String classification = classifyBMI(bmi);

System.out.println("BMI: " + bmi);

System.out.println("Classification: " + classification);

}

}

public class BMITrackerTest {

public static void main(String[] args) {

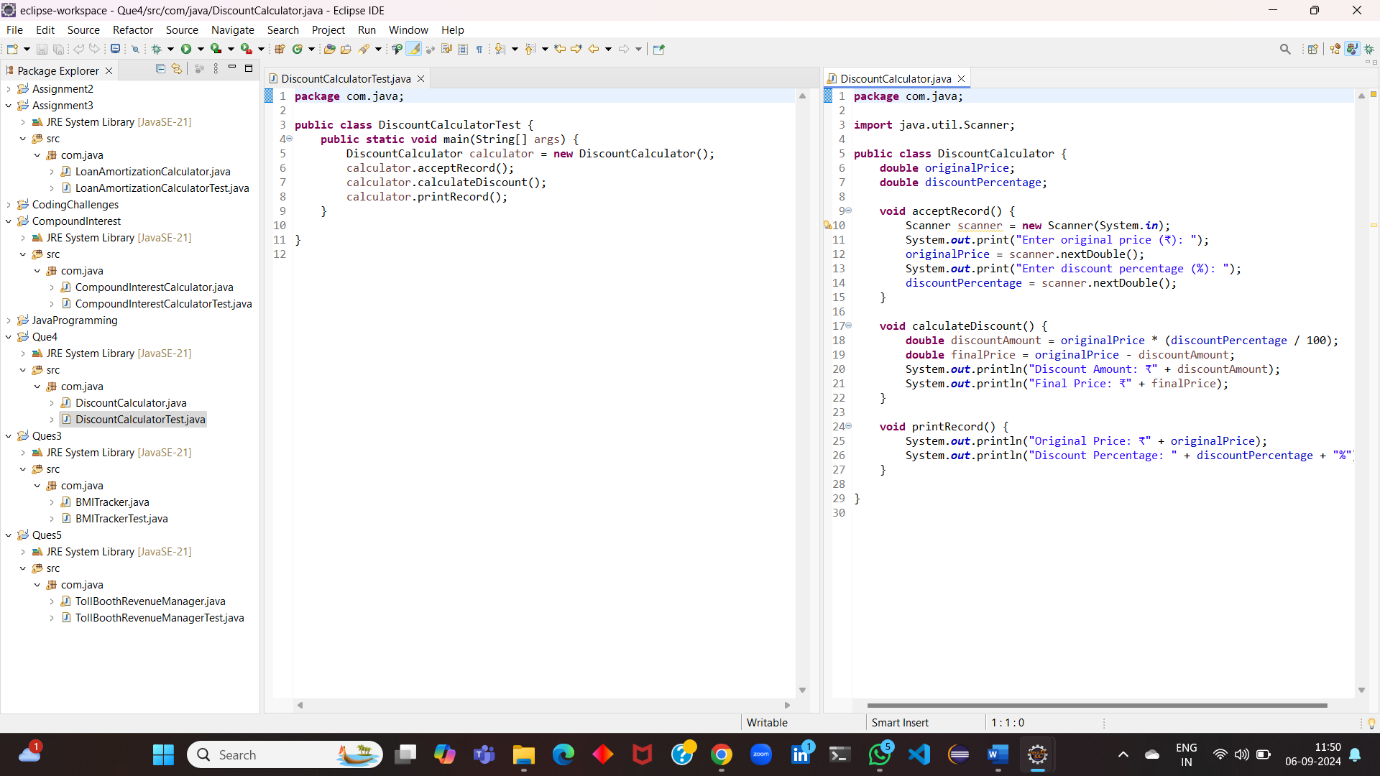
BMITracker tracker = new BMITracker();

tracker.acceptRecord();

tracker.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:

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.

import java.util.Scanner;

class DiscountCalculator {

double originalPrice;

double discountPercentage;

void acceptRecord() {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter original price (₹): ");

originalPrice = scanner.nextDouble();

System.out.print("Enter discount percentage (%): ");

discountPercentage = scanner.nextDouble();

}

void calculateDiscount() {

double discountAmount = originalPrice \* (discountPercentage / 100);

double finalPrice = originalPrice - discountAmount;

System.out.println("Discount Amount: ₹" + discountAmount);

System.out.println("Final Price: ₹" + finalPrice);

}

void printRecord() {

System.out.println("Original Price: ₹" + originalPrice);

System.out.println("Discount Percentage: " + discountPercentage + "%");

}

}

public class DiscountCalculatorTest {

public static void main(String[] args) {

DiscountCalculator calculator = new DiscountCalculator();

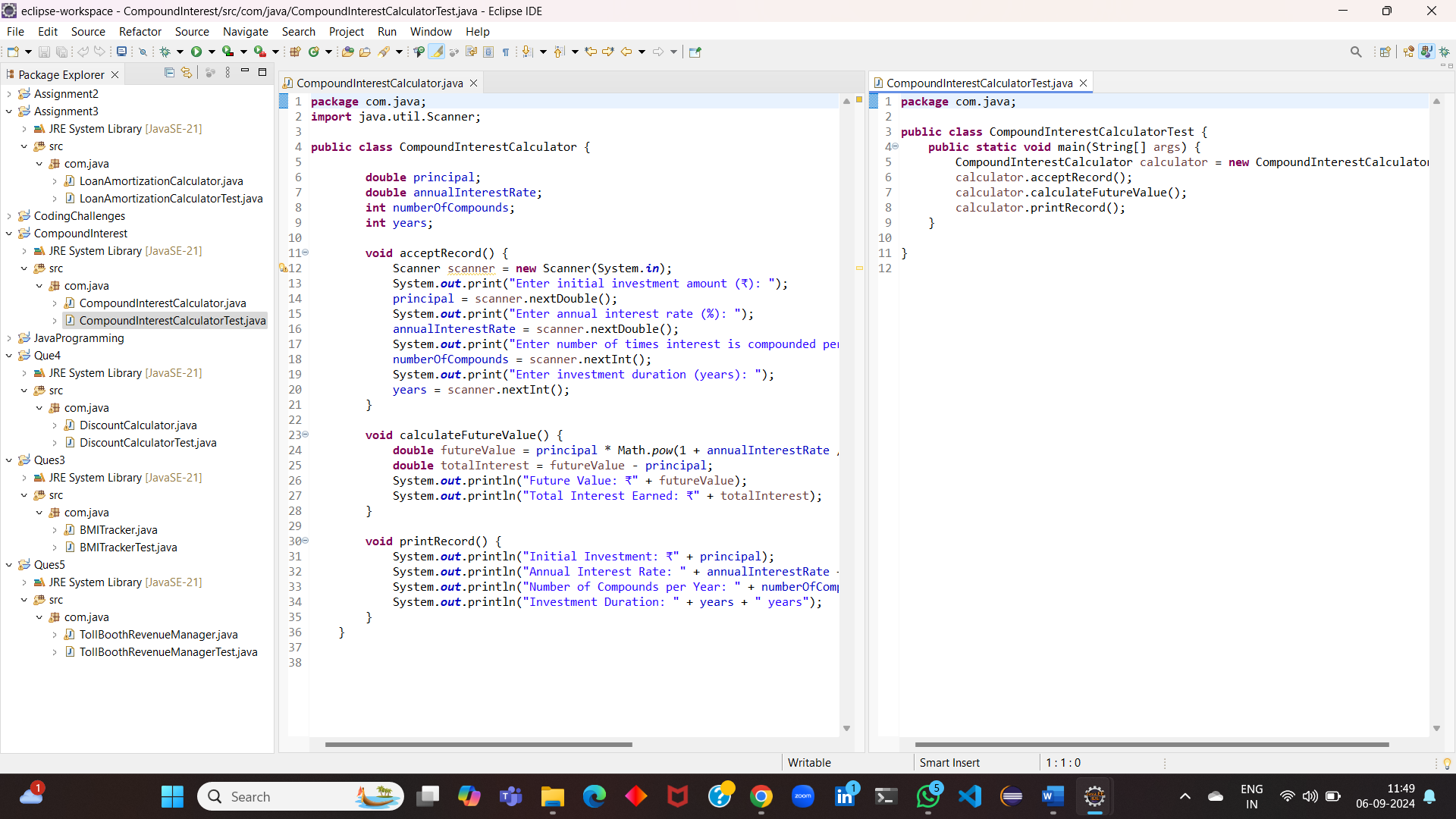
calculator.acceptRecord();

calculator.calculateDiscount();

calculator.printRecord();

}

}



**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.

import java.util.Scanner;

class TollBoothRevenueManager {

double carToll;

double truckToll;

double motorcycleToll;

int carCount;

int truckCount;

int motorcycleCount;

void setTollRates() {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter toll rate for cars (₹): ");

carToll = scanner.nextDouble();

System.out.print("Enter toll rate for trucks (₹): ");

truckToll = scanner.nextDouble();

System.out.print("Enter toll rate for motorcycles (₹): ");

motorcycleToll = scanner.nextDouble();

}

void acceptRecord() {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter number of cars: ");

carCount = scanner.nextInt();

System.out.print("Enter number of trucks: ");

truckCount = scanner.nextInt();

System.out.print("Enter number of motorcycles: ");

motorcycleCount = scanner.nextInt();

}

void calculateRevenue() {

double totalRevenue = carToll \* carCount + truckToll \* truckCount + motorcycleToll \* motorcycleCount;

System.out.println("Total Revenue: ₹" + totalRevenue);

}

void printRecord() {

System.out.println("Toll Rates:");

System.out.println("Car: ₹" + carToll);

System.out.println("Truck: ₹" + truckToll);

System.out.println("Motorcycle: ₹" + motorcycleToll);

System.out.println("Number of Vehicles:");

System.out.println("Cars: " + carCount);

System.out.println("Trucks: " + truckCount);

System.out.println("Motorcycles: " + motorcycleCount);

}

}

public class TollBoothRevenueManagerTest {

public static void main(String[] args) {

TollBoothRevenueManager manager = new TollBoothRevenueManager();

manager.setTollRates();

manager.acceptRecord();

manager.calculateRevenue();

manager.printRecord();

}

}

