**ASSIGNMENT4**

**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 (₹).
4. import java.util.Scanner;
5. class LoanAmortizationCalculator{
6. private double  principal;
7. private double annualInterestRate;
8. private int loanTerm;
9. public void acceptRecord() {
10. Scanner sc= new Scanner(System.in);
12. System.out.println("enter principle");
13. principal=sc.nextDouble();
14. System.out.println("enter annualInterestRate");
15. annualInterestRate=sc.nextDouble();
16. System.out.println("enter loan term in year");
17. loanTerm=sc.nextInt();
18. }
19. public double calculateMonthlyPayment() {
20. double monthlyInterestRate = ((annualInterestRate / 12) / 100);
21. int numberOfMonths = (loanTerm \* 12);
22. double monthlyPayment;

25. if (monthlyInterestRate !=0)
26. {
27. monthlyPayment = principal \* (monthlyInterestRate \* Math.pow(1 + monthlyInterestRate, numberOfMonths)) / (Math.pow(1 + monthlyInterestRate, numberOfMonths) - 1);
28. }
29. else {
30. monthlyPayment = principal / numberOfMonths;
31. }
32. return monthlyPayment;

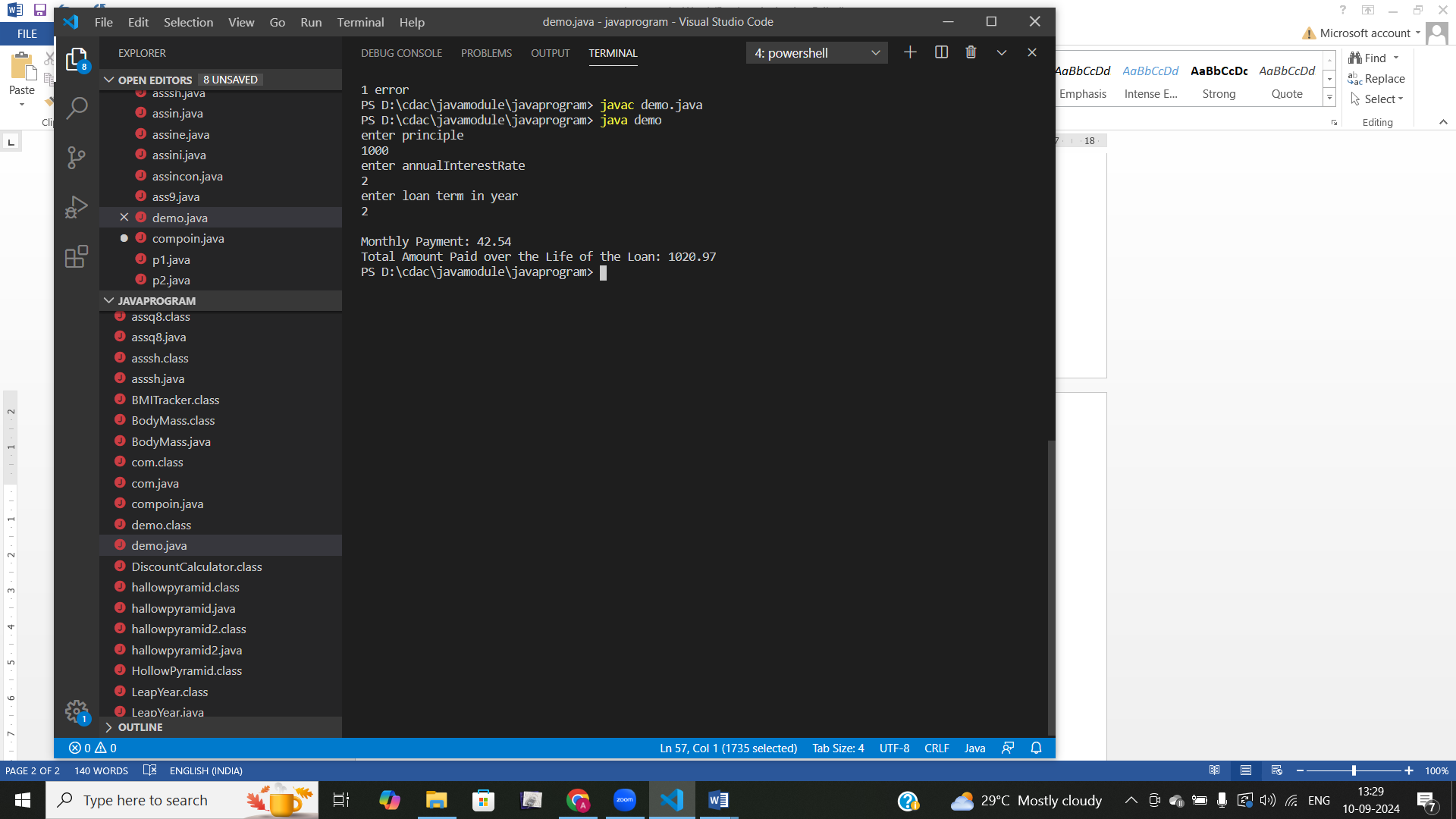

36. }
37. public void printRecord(double monthlyPayment) {
38. double totalPayment = monthlyPayment \* loanTerm \* 12;
39. System.out.printf("\nMonthly Payment: %.2f", monthlyPayment);
40. System.out.printf("\nTotal Amount Paid over the Life of the Loan: %.2f\n", totalPayment);
41. }
42. }
43. public class demo {
44. public static void main(String[] args) {
45. LoanAmortizationCalculator loanCalculator = new LoanAmortizationCalculator();

48. loanCalculator.acceptRecord();

51. double monthlyPayment = loanCalculator.calculateMonthlyPayment();

54. loanCalculator.printRecord(monthlyPayment);

57. }
58. }



**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 com {

    private double principal;

    private double annualInterestRate;

    private int numberOfCompounds;

    private int years;

    public void acceptRecord() {

        Scanner scanner = new Scanner(System.in);

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

        principal = scanner.nextDouble();

        System.out.print("Enter the interest: ");

        annualInterestRate = scanner.nextDouble() / 100;

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

        numberOfCompounds = scanner.nextInt();

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

        years = scanner.nextInt();

    }

    public double calculateFutureValue() {

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

    }

    public void printRecord() {

        double futureValue = calculateFutureValue();

        double totalInterest = futureValue - principal;

        System.out.printf("Future Value of the investment: %.2f\n", futureValue);

        System.out.printf("Total Interest Earned: %.2f\n", totalInterest);

    }

    public static void main(String[] args) {

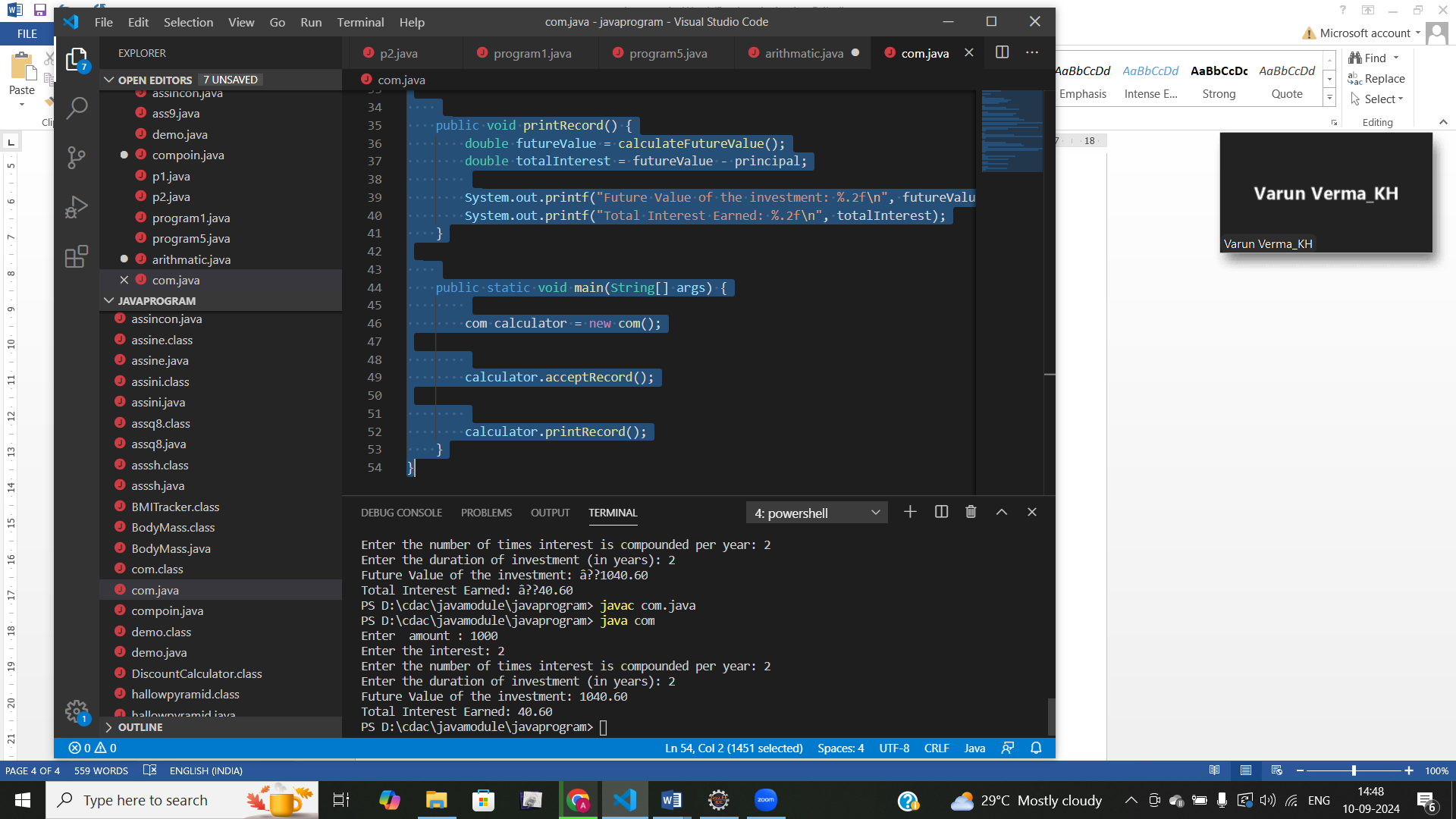
        com calculator = new com();

        calculator.acceptRecord();

        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{

     int weight;

     double height;

     double BMI;

     String classification;

    public void acceptRecord(){

        Scanner sc=new Scanner(System.in);

        System.out.println("enter weight kg");

         weight=sc.nextInt();

        System.out.println("enter height in meter");

         height=sc.nextDouble();

        sc.close();

    }

        public void calculateBMI(){

            BMI = weight / (height \* height);

        }

     public void classifyBMI(){

         if(BMI<18.5)

         {

             classification="underwieght";

         }

        else if (BMI>= 18.5 && BMI< 24.9) {

            classification = "Normal weight";

        } else if (BMI>= 25 && BMI < 29.9) {

            classification = "Overweight";

        } else if (BMI >= 30) {

            classification = "Obese";

        }

}

public void printRecord() {

        System.out.printf("BMI: %.2f%n", BMI);

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

    }

}

class program5{

    public static void main(String args[]){

         BMITracker bmi=new  BMITracker();

         bmi.acceptRecord();

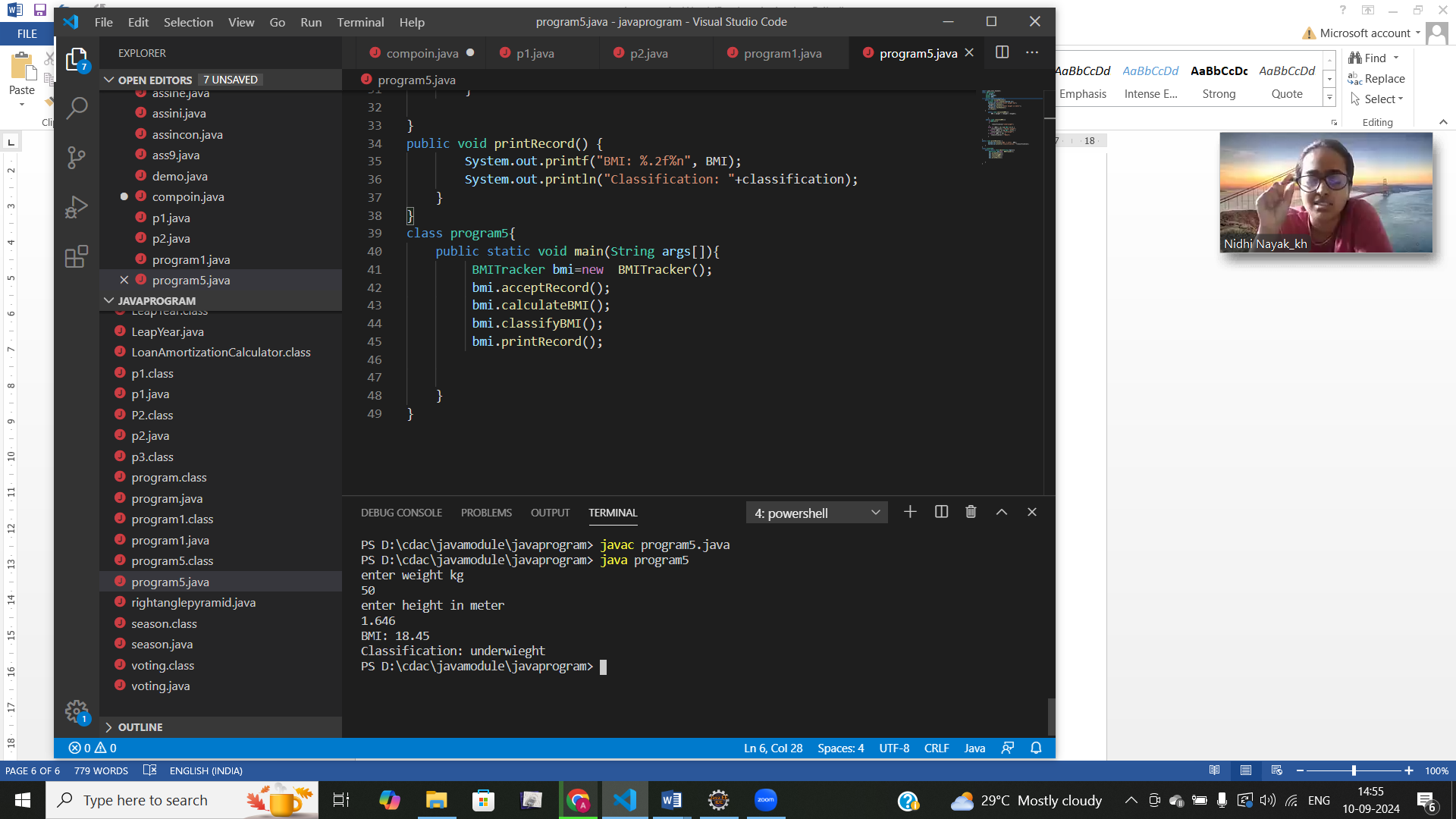
         bmi.calculateBMI();

         bmi.classifyBMI();

         bmi.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 {

    private double originalPrice;

    private double discountRate;

    public void acceptRecord() {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the original price of the item");

        originalPrice = sc.nextDouble();

        System.out.print("Enter the discount percentage ");

        discountRate = sc.nextDouble();

    }

    public double calculateDiscountAmount() {

        return originalPrice \* (discountRate / 100);

    }

    public double calculateFinalPrice() {

        return originalPrice - calculateDiscountAmount() ;

    }

    public void printRecord() {

          double discountAmount = calculateDiscountAmount();

        double finalPrice = calculateFinalPrice();

        System.out.println("\nOriginal Price: "+originalPrice);

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

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

    }

}

public class ass34{

    public static void main(String[] args) {

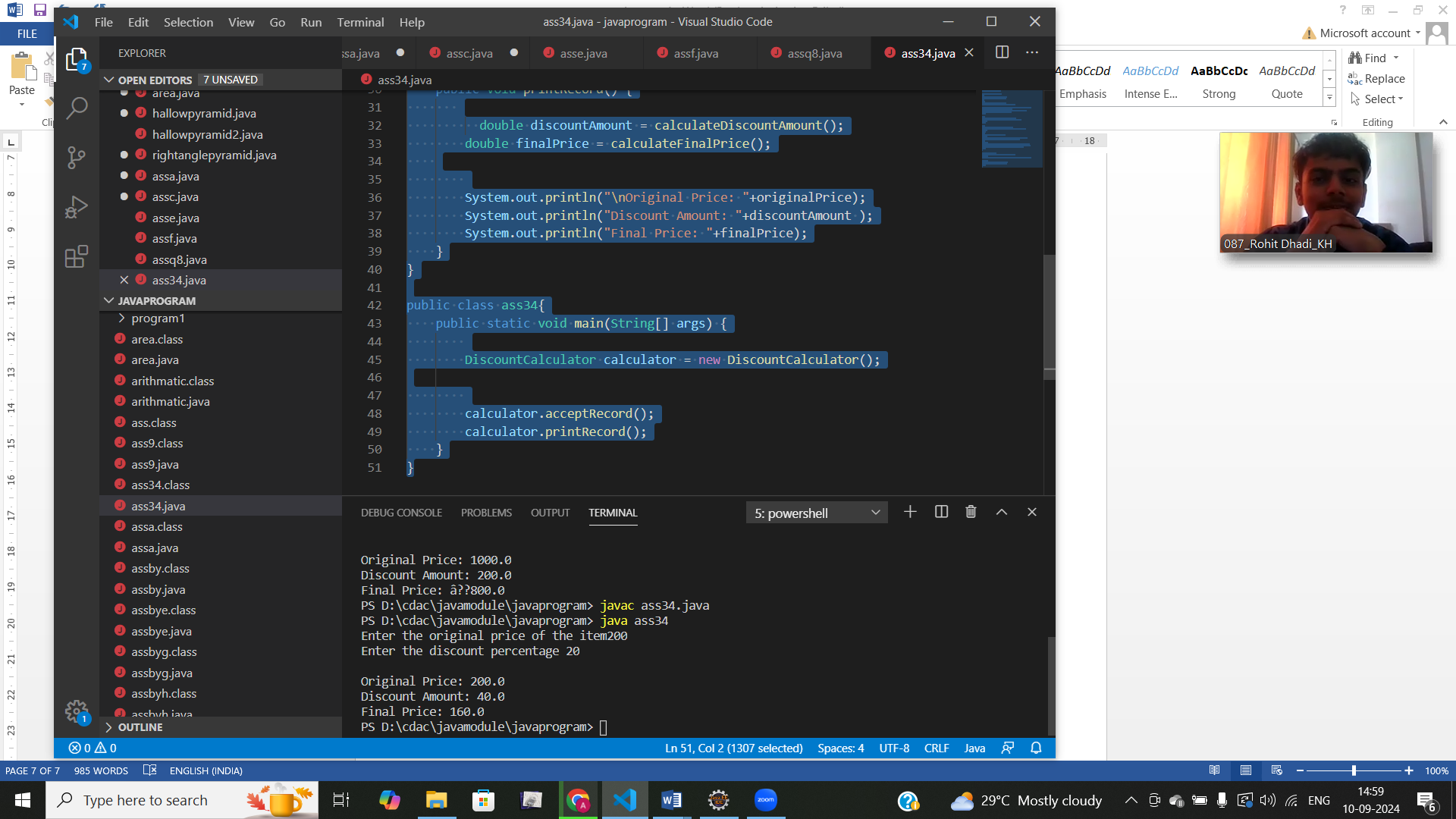
        DiscountCalculator calculator = new DiscountCalculator();

        calculator.acceptRecord();

        calculator.printRecord();

    }

}



**5. Toll Booth Revenue Management**

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

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 carTollRate;

    double truckTollRate;

    double motorcycleTollRate;

    int numCars;

    int numTrucks;

    int numMotorcycles;

    public void setTollRates() {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter the toll rate for Car : ");

        carTollRate = sc.nextDouble();

        System.out.print("Enter the toll rate for Truck : ");

        truckTollRate = sc.nextDouble();

        System.out.print("Enter the toll rate for Motorcycle : ");

        motorcycleTollRate = sc.nextDouble();

    }

    public void acceptRecord() {

        Scanner sc = new Scanner(System.in);

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

        numCars = sc.nextInt();

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

        numTrucks = sc.nextInt();

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

        numMotorcycles = sc.nextInt();

    }

    public double calculateRevenue() {

        double totalRevenue = (numCars \* carTollRate) + (numTrucks \* truckTollRate) + (numMotorcycles \* motorcycleTollRate);

        return totalRevenue;

    }

    public void printRecord() {

        int totalVehicles = numCars + numTrucks + numMotorcycles;

        double totalRevenue = calculateRevenue();

        System.out.println("\nTotal number of vehicles: " + totalVehicles);

        System.out.println("Total revenue collected: " + totalRevenue);

    }

}

public class toll {

    public static void main(String[] args) {

        TollBoothRevenueManager manager = new TollBoothRevenueManager();

        manager.setTollRates();

        manager.acceptRecord();

        manager.printRecord();

    }

}

