**Introduction To OOPs**

**(JAVA Programming)**

**UET Peshawar**

**Department Of Computer Science**

**2nd Semester (spring)**



**Lab 2 Tasks**

**Submitted By:**

Waqas Khan s/o Azeem Khan

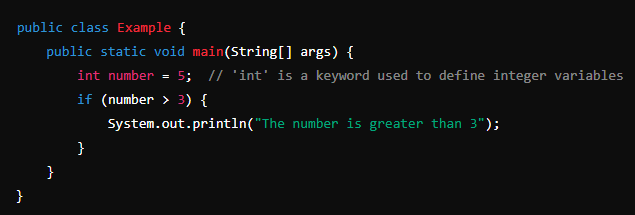
RegNo : 24pwbcs1141

**Submitted To:**

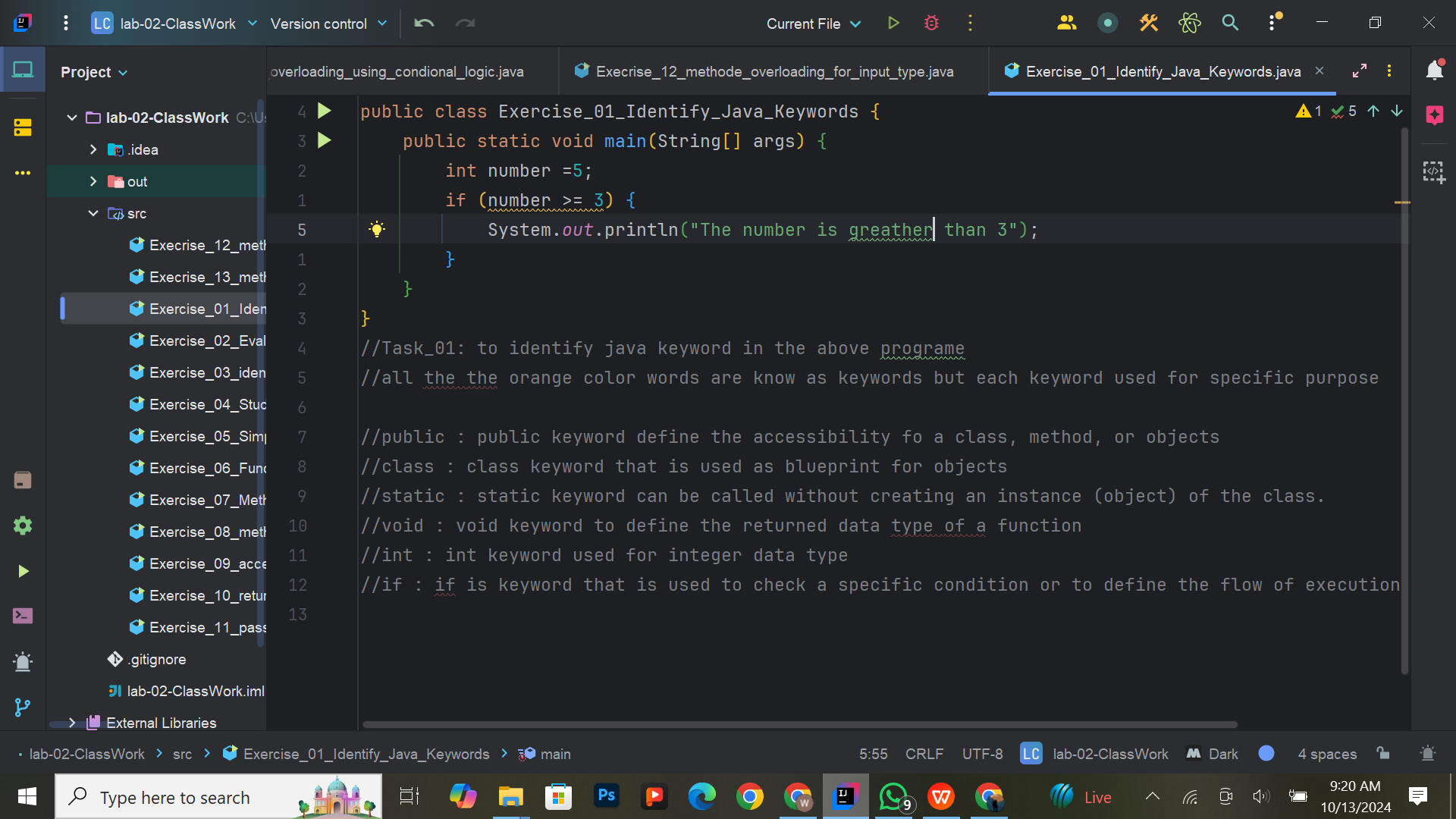
Sir Syed Adeel Ali Shah Associate Professor CS&IT Peshawar

**Exercise 1: Identifying Java Keywords**

**Task:**Identify all the keywords used in the following Java program and explain the purpose of each keyword.



**Expected Output:**  
A list of keywords with explanations of their function.



**Exercise 2: Writing and Evaluating Expressions**

**Task:**  
Write a program that declares four integer variables, performs an arithmetic operation (addition or multiplication), and prints the result.

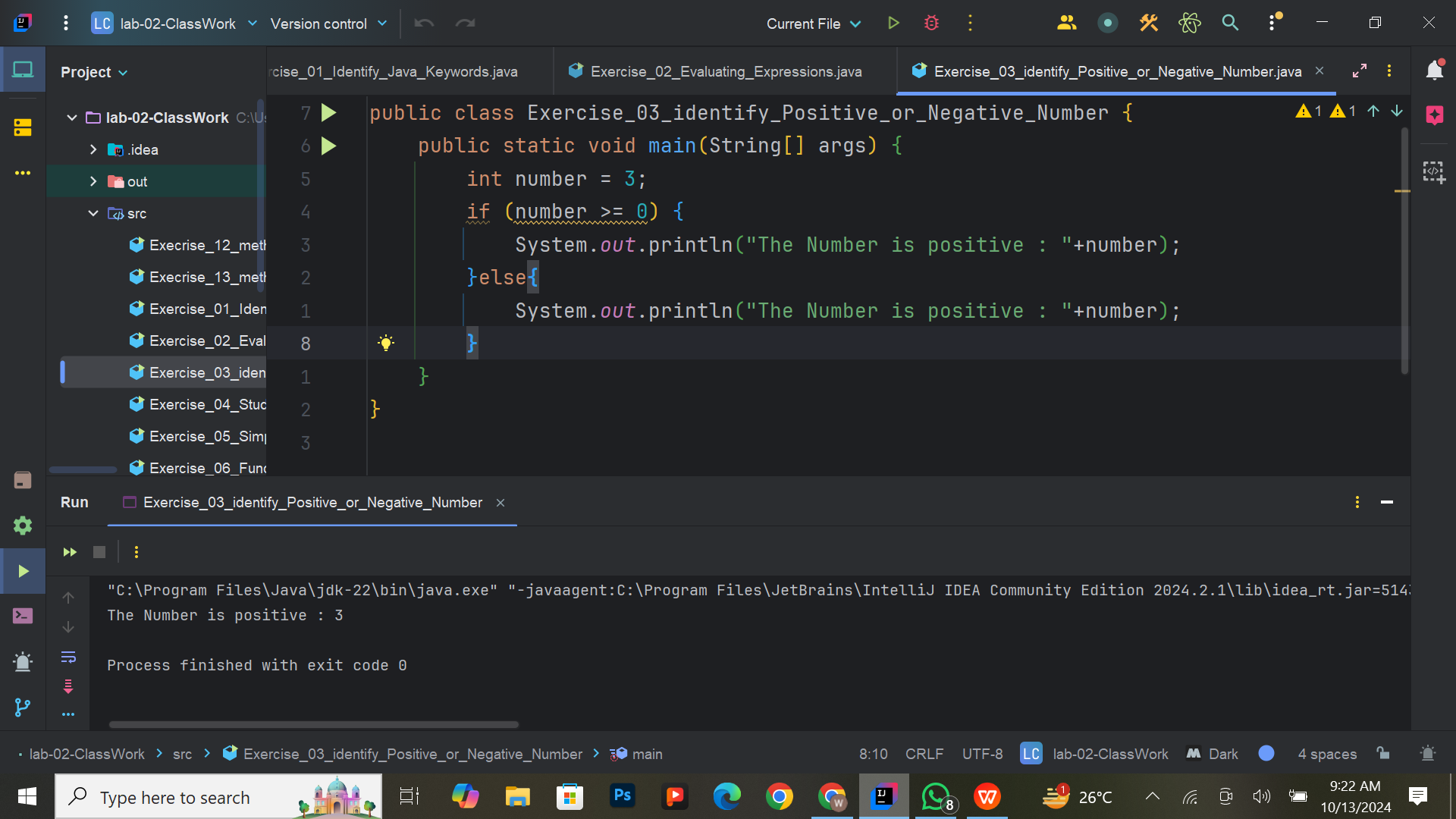
**Expected Output:**  
The result of the arithmetic operation.



**Exercise 3: Understanding Code Blocks and Indentation**

**Task:**  
Write a program where you include a simple if-else block to check whether a number is positive or negative. Ensure proper indentation for clarity.

**Expected Output:**  
The program will print whether the number is positive or negative.

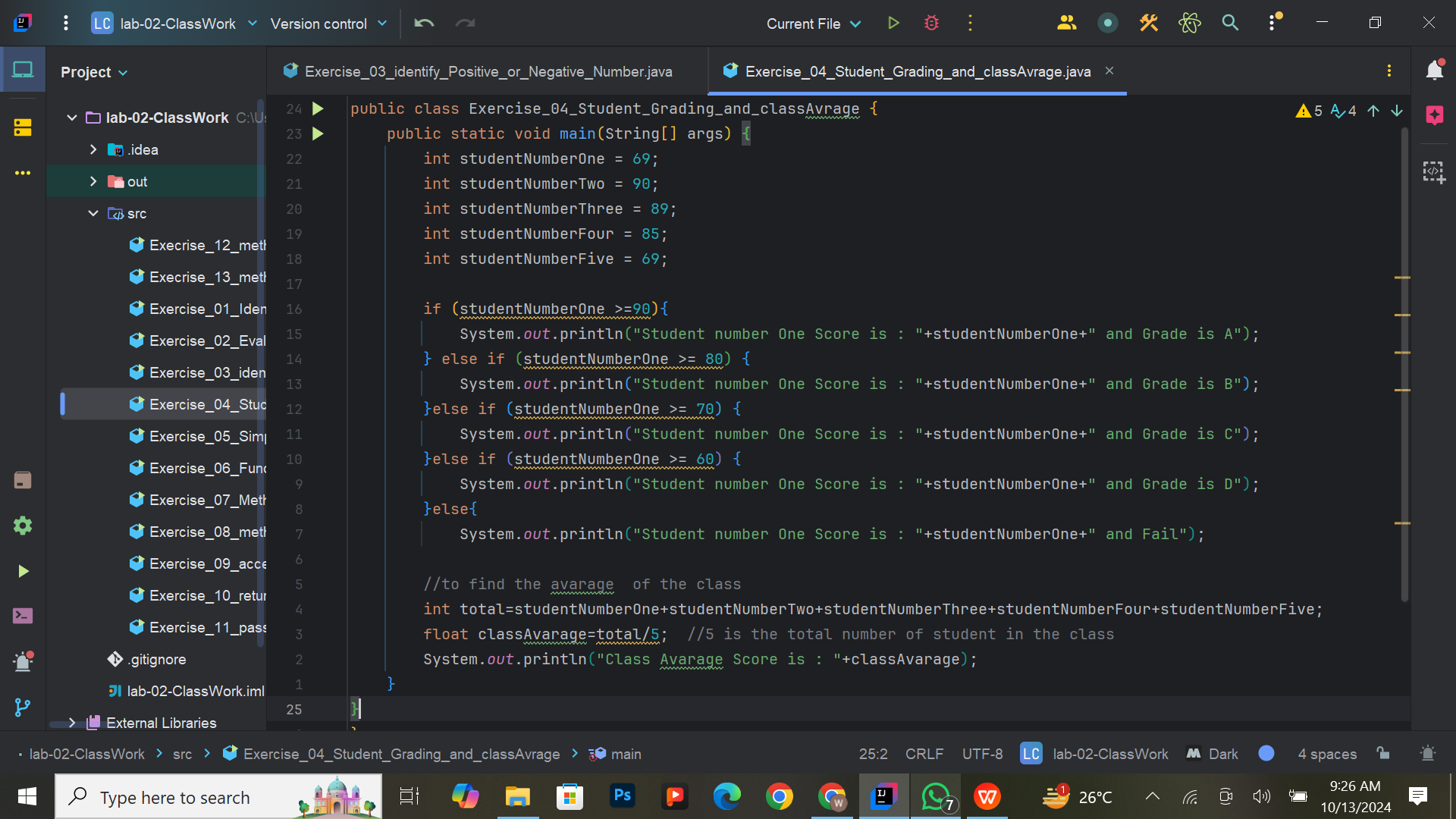


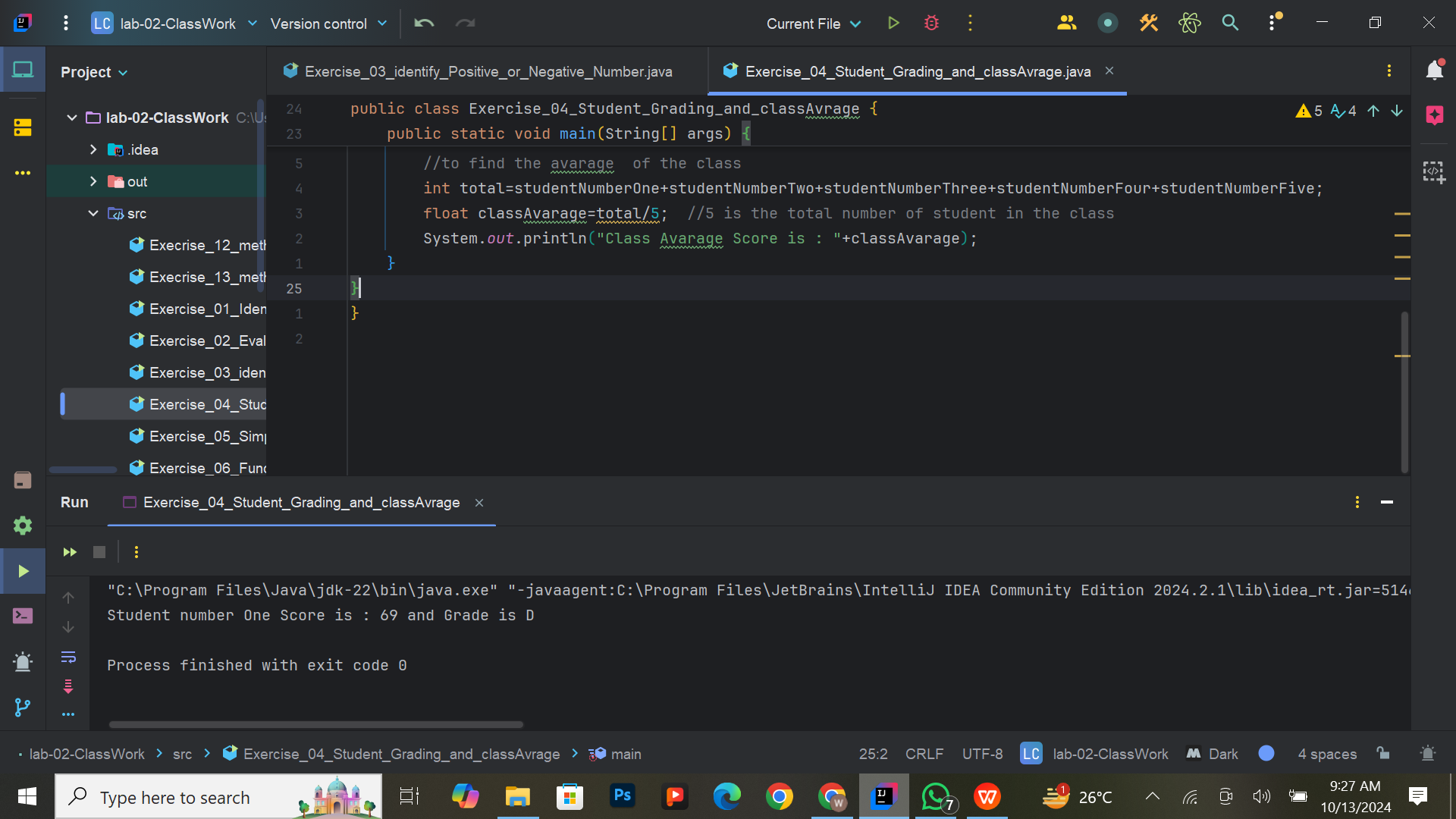
**Modified Exercise 4: Grading Multiple Students and Class Average**

**Task:**  
Write a program that evaluates the scores of multiple students and assigns grades based on the following criteria:

1. 90 and above: Grade A
2. 80 and above: Grade B
3. 70 and above: Grade C
4. 60 and above: Grade D
5. Below 60: Fail

Additionally, after grading each student, the program should calculate the class average and display it.

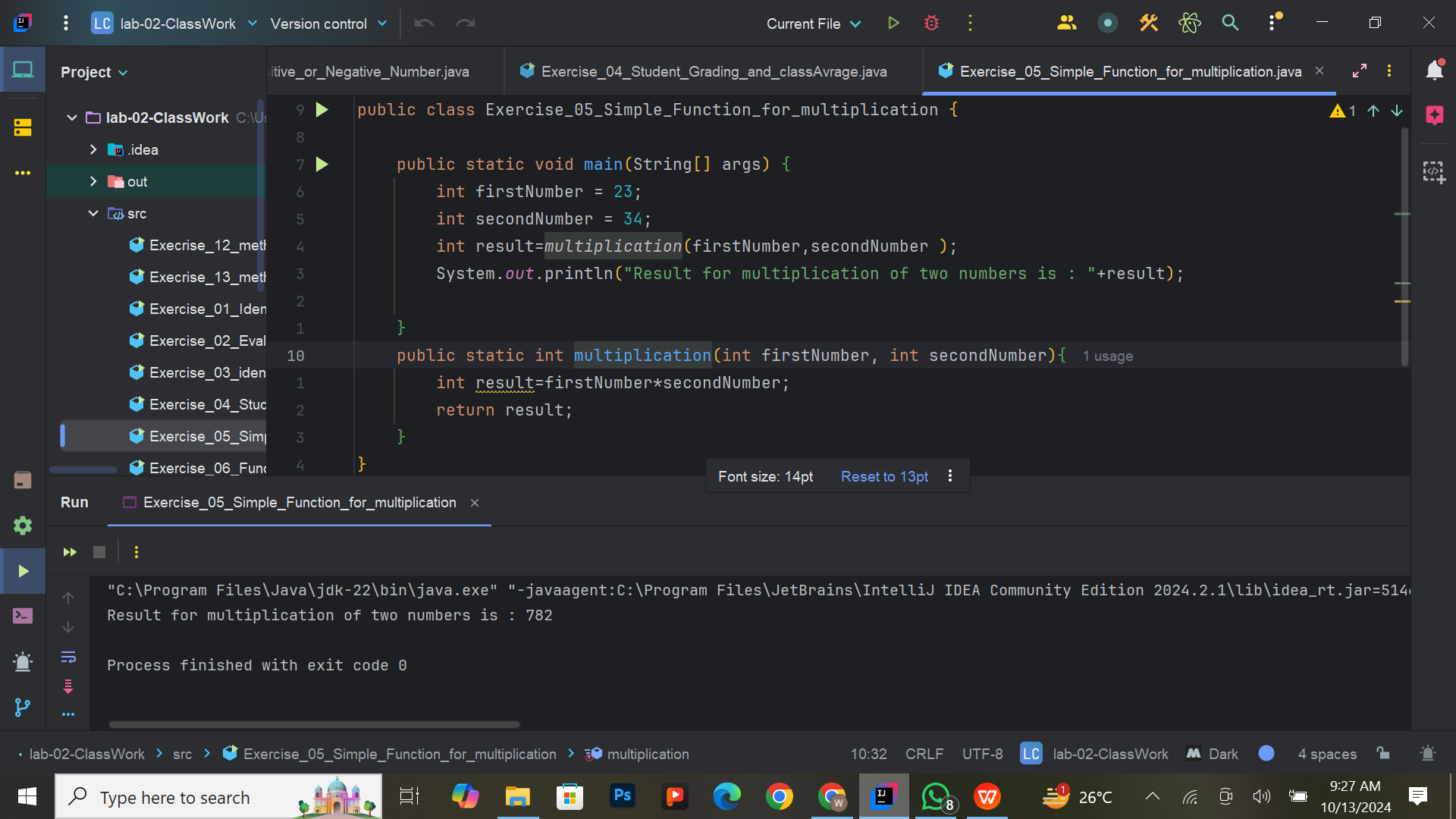




**Exercise 5: Writing a Simple Function**

**Task:**  
Write a function that takes two integer inputs, multiplies them, and returns the result. Call the function from the main method and print the result.

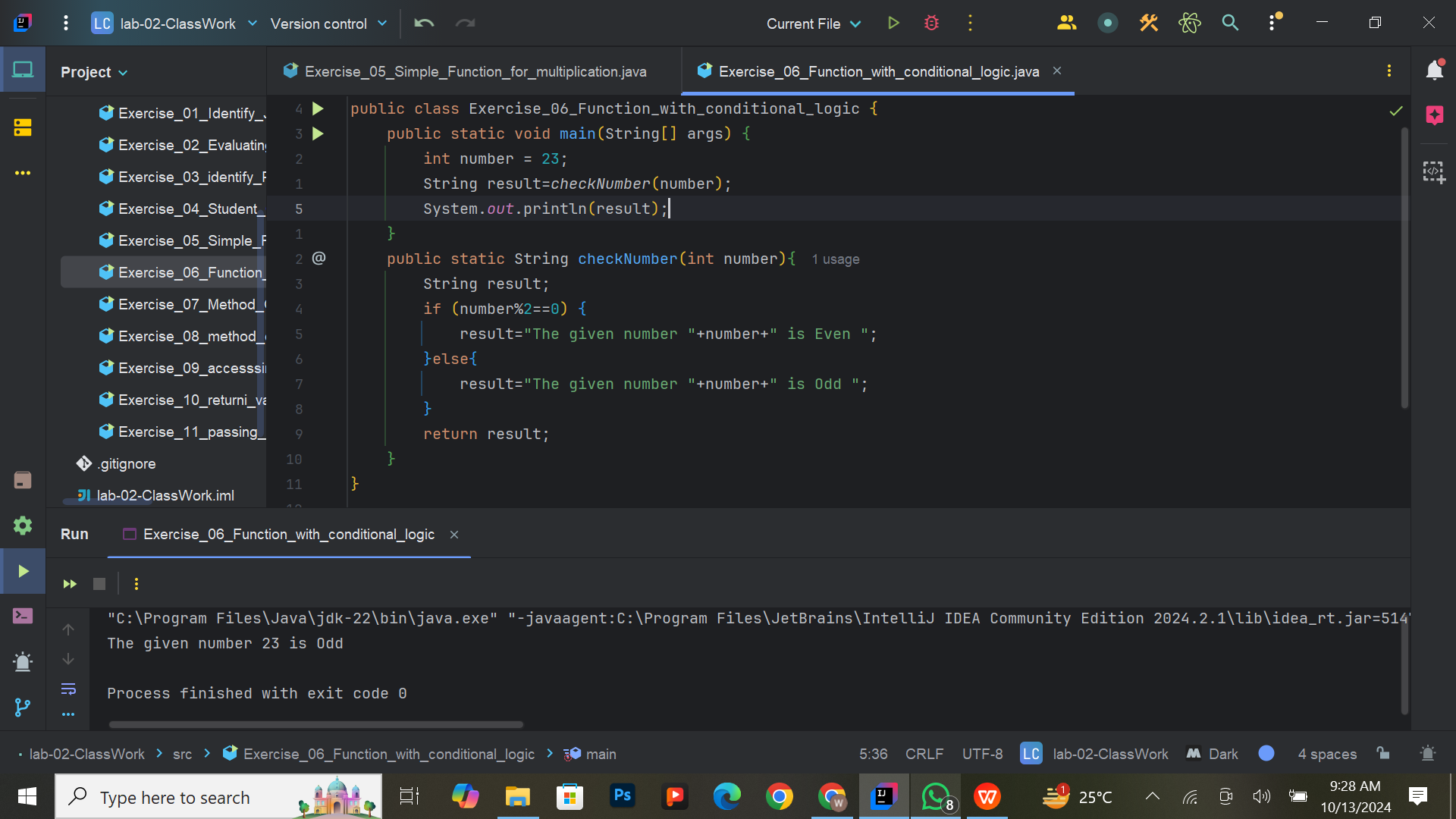
**Expected Output:**  
The product of the two integers.



**Exercise 6: Function with Conditional Logic**

**Task:**  
Write a function that accepts an integer as an argument, checks if it's even or odd, and returns the corresponding message.

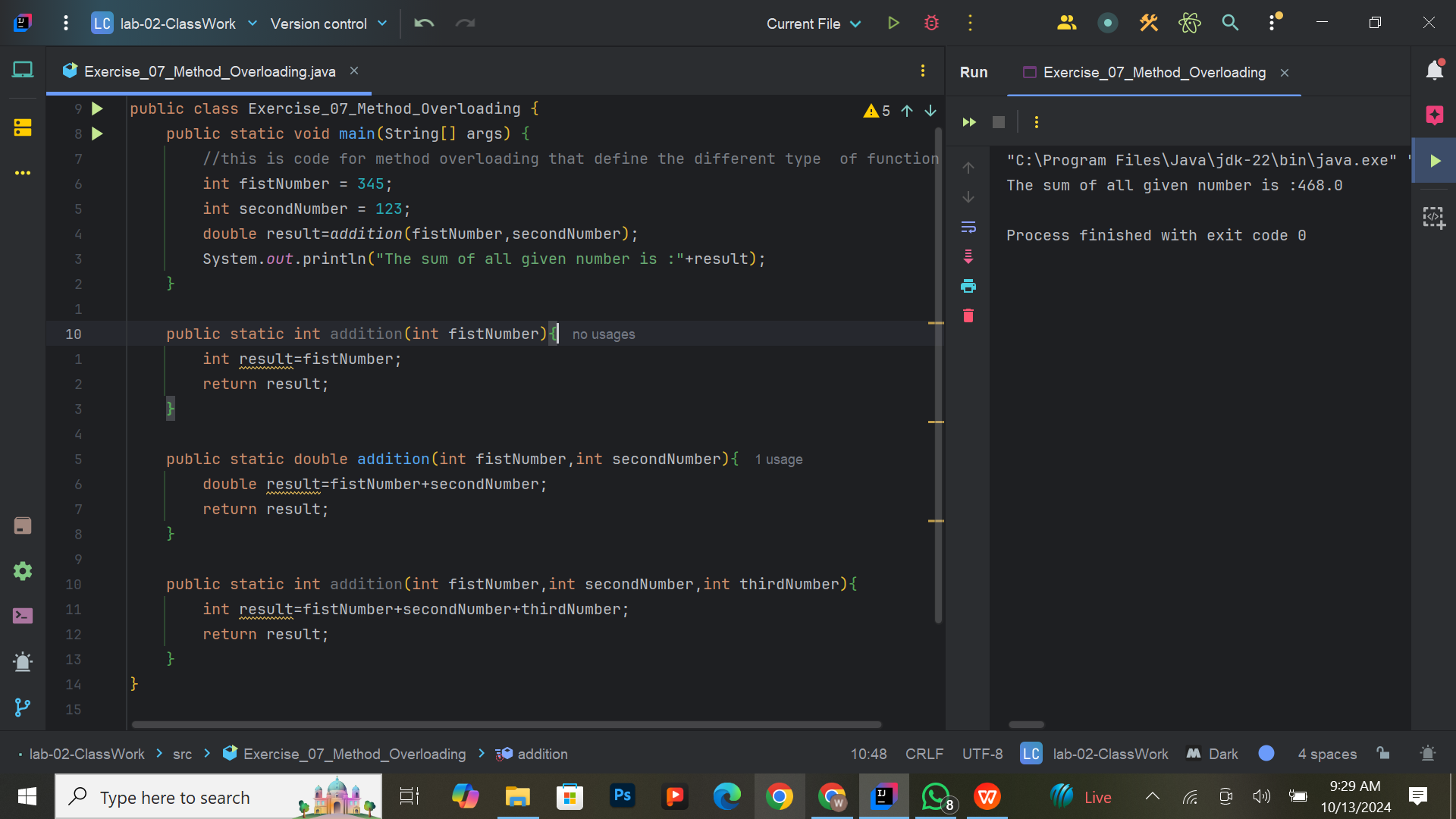
**Expected Output:**  
The function will return a message indicating whether the number is even or odd.



**Exercise 7: Method Overloading – Adding Integers and Doubles**

**Task:**  
Overload a method named addValues that adds two integers in one version and two doubles in another. Call both versions and print the results.

**Expected Output:**  
Two outputs: one with the sum of integers, and the other with the sum of doubles.



**Modified Exercise 8: Calculating Total Inventory Value Using Method Overloading**

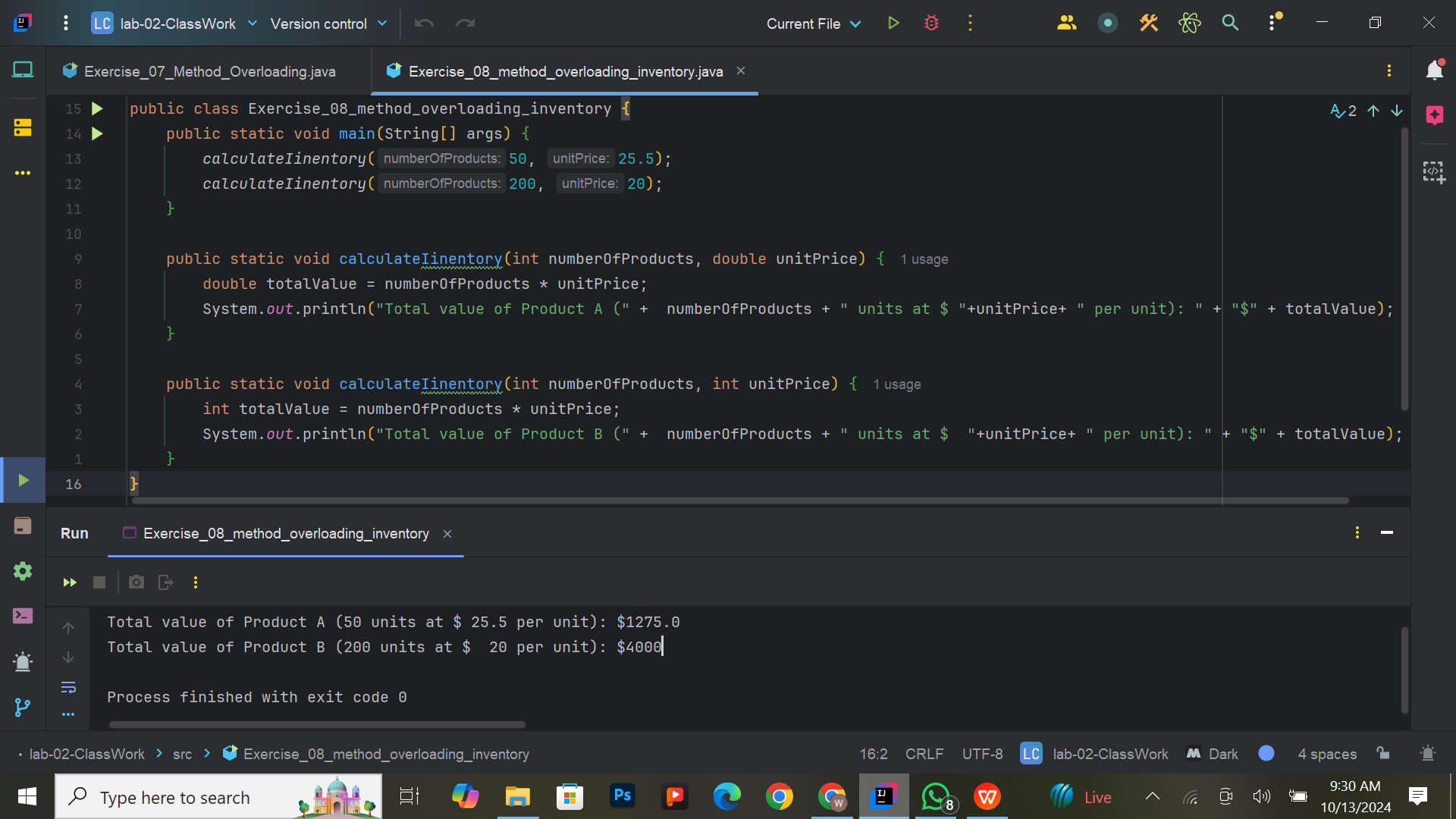
**Task:**  
Write overloaded methods that calculate the total value of an inventory item, where:

1. The first method calculates the total value based on the product count (integer) and price per unit (double).
2. The second method calculates the total value based on the product count (integer) and price per unit (integer), assuming both quantities are whole numbers (e.g., for bulk pricing).

Use method overloading to handle both cases, ensuring that the correct method is called based on the data types passed in. After calculating the total value for each item, print the result.

**Expected Output:**  
Total value of Product A (50 units at $25.5 per unit): $1275.0

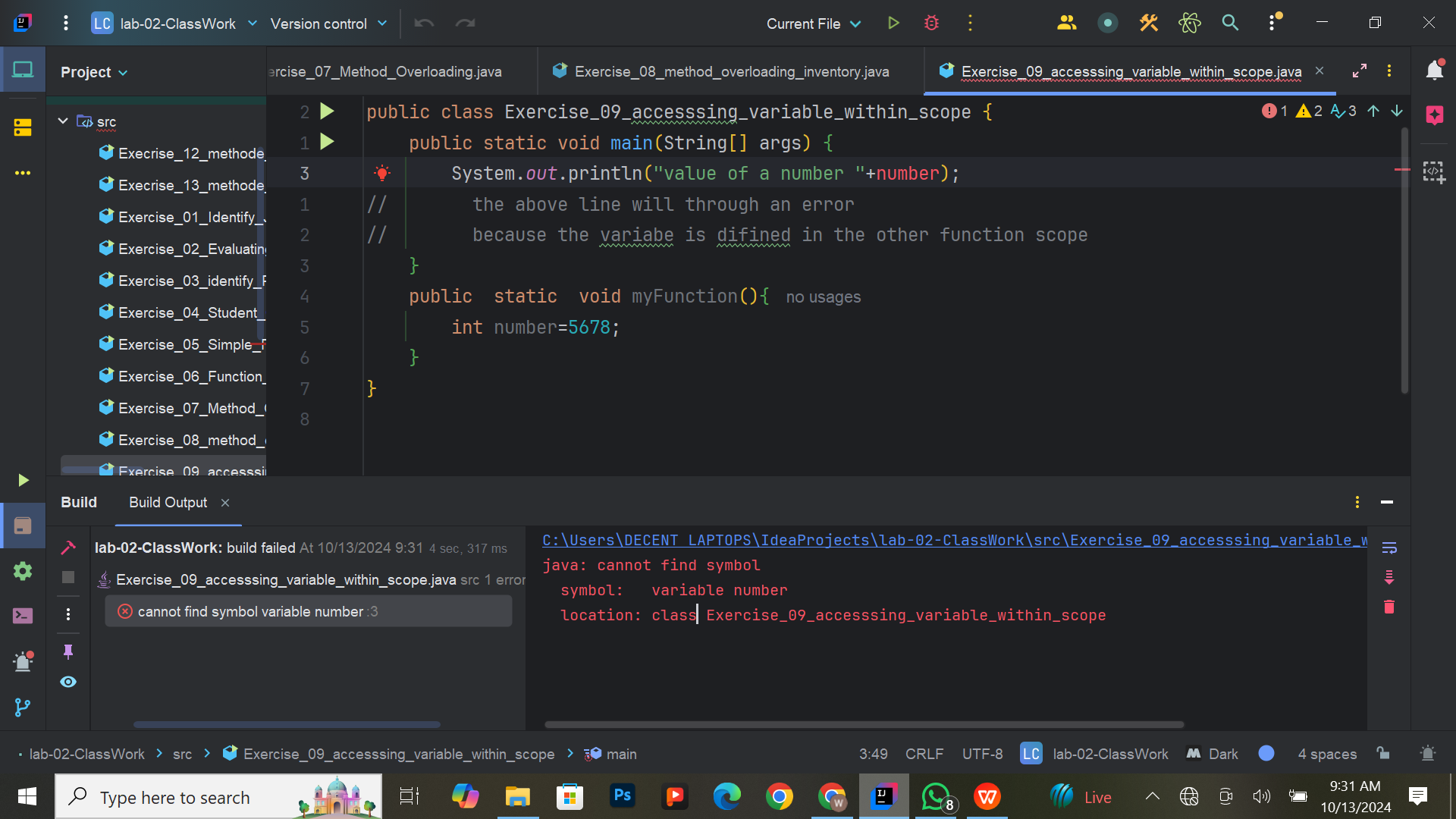
Total value of Product B (200 units at $20 per unit): $4000



**Exercise 9: Accessing Variables Within Code Blocks**

**Task:**  
Write a program that declares a variable inside an if block and attempts to access it outside the block. Analyze the result and explain the scope of the variable.

**Expected Output:**  
The program should explain why the variable is inaccessible outside the block.



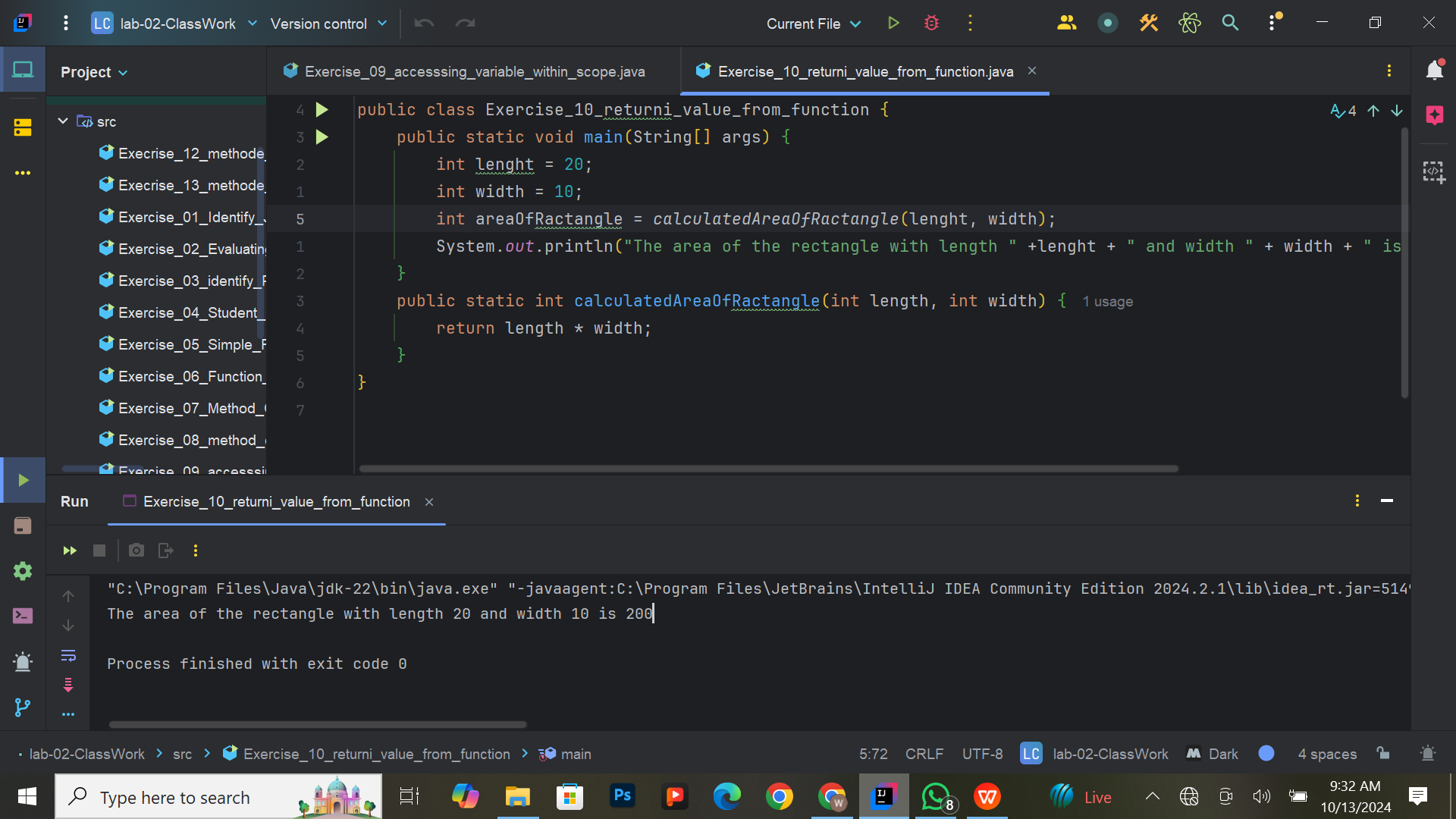
**Exercise 10: Returning Values from Functions**

**Task:**  
Create a function named calculateRectangleArea that takes the length and width of a rectangle as parameters, computes the area, and returns the result. In the main method, call this function using predefined values for the length and width, and display the area in a user-friendly message.

**Steps to Complete:**

1. Define the calculateRectangleArea function.
2. In the main method, set predefined values for the length and width of the rectangle.
3. Call the calculateRectangleArea function with these values and store the returned result.
4. Print a message displaying the area of the rectangle.

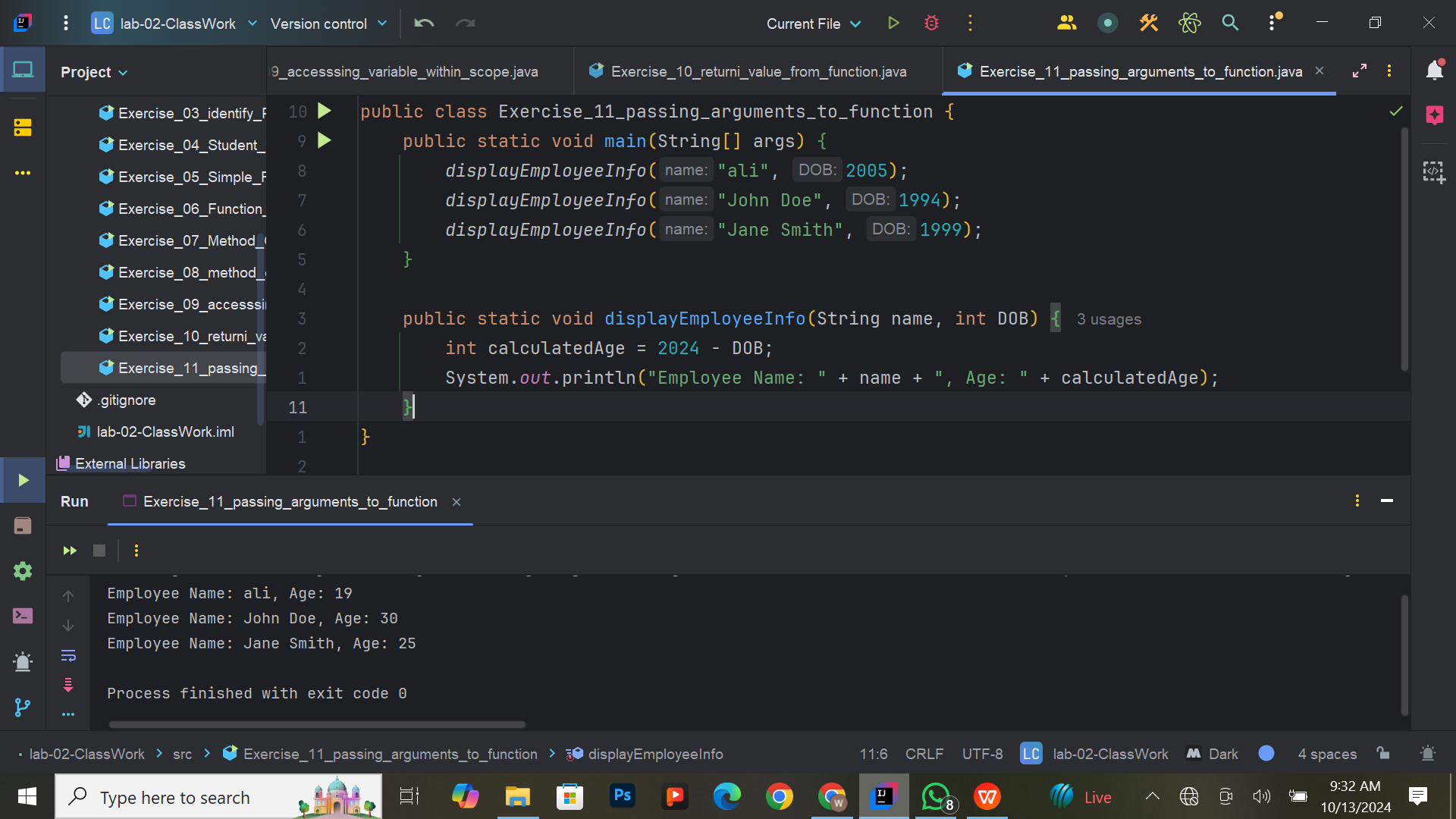
**Expected Output:**  
The program should output a message like:  
*“The area of the rectangle with length [length] and width [width] is [area].”*

**

**Exercise 11: Passing Arguments to Functions**

**Task:**  
Create a function named displayEmployeeInfo that accepts an employee's name (as a string) and their year of birth (as an integer) as input parameters. The function should calculate the employee's age based on the current year and print a formatted message that includes both the name and the calculated age. In the main method, call this function multiple times with different employee names and years of birth to demonstrate its functionality.

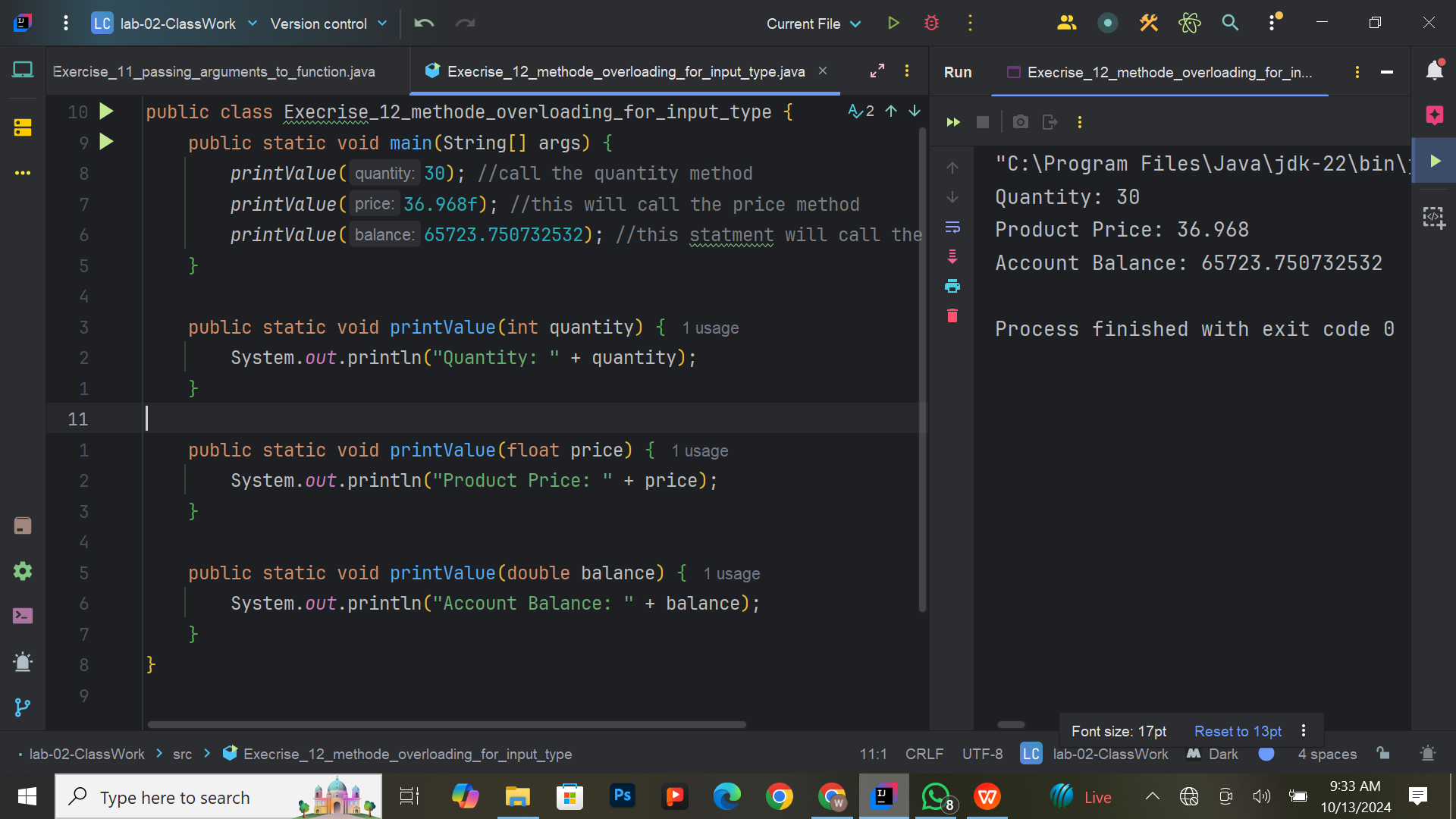
**Expected Output:**  
The program should output messages like:  
*“Employee Name: [name], Age: [calculated age]”*  
For example:  
*“Employee Name: John Doe, Age: 30”* (if the current year is 2024 and the year of birth is 1994)  
*“Employee Name: Jane Smith, Age: 25”* (if the current year is 2024 and the year of birth is 1999)



**Exercise 12: Using Method Overloading for Input Types**

**Task:**  
Create a method named printValue that is overloaded to handle different input types: integer, float, and double. This method should be used to print the details of various financial transactions. In the main method, call the printValue method for each data type with meaningful examples like a product price (float), a quantity (integer), and an account balance (double).

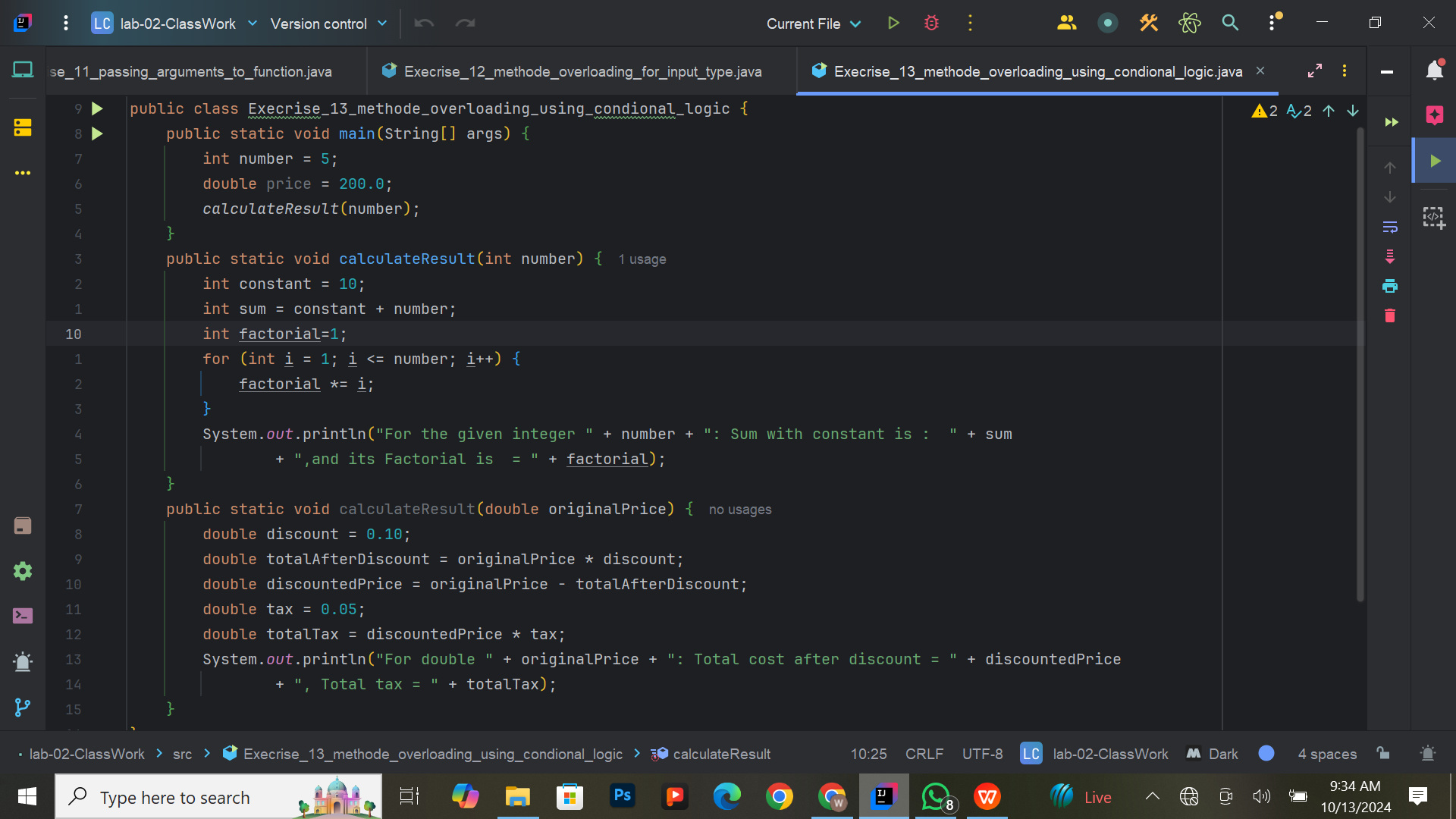
**Expected Output:**  
The program should output messages like:  
*“Quantity: 10”*  
*“Product Price: 19.99”*  
*“Account Balance: 1500.75”*

**

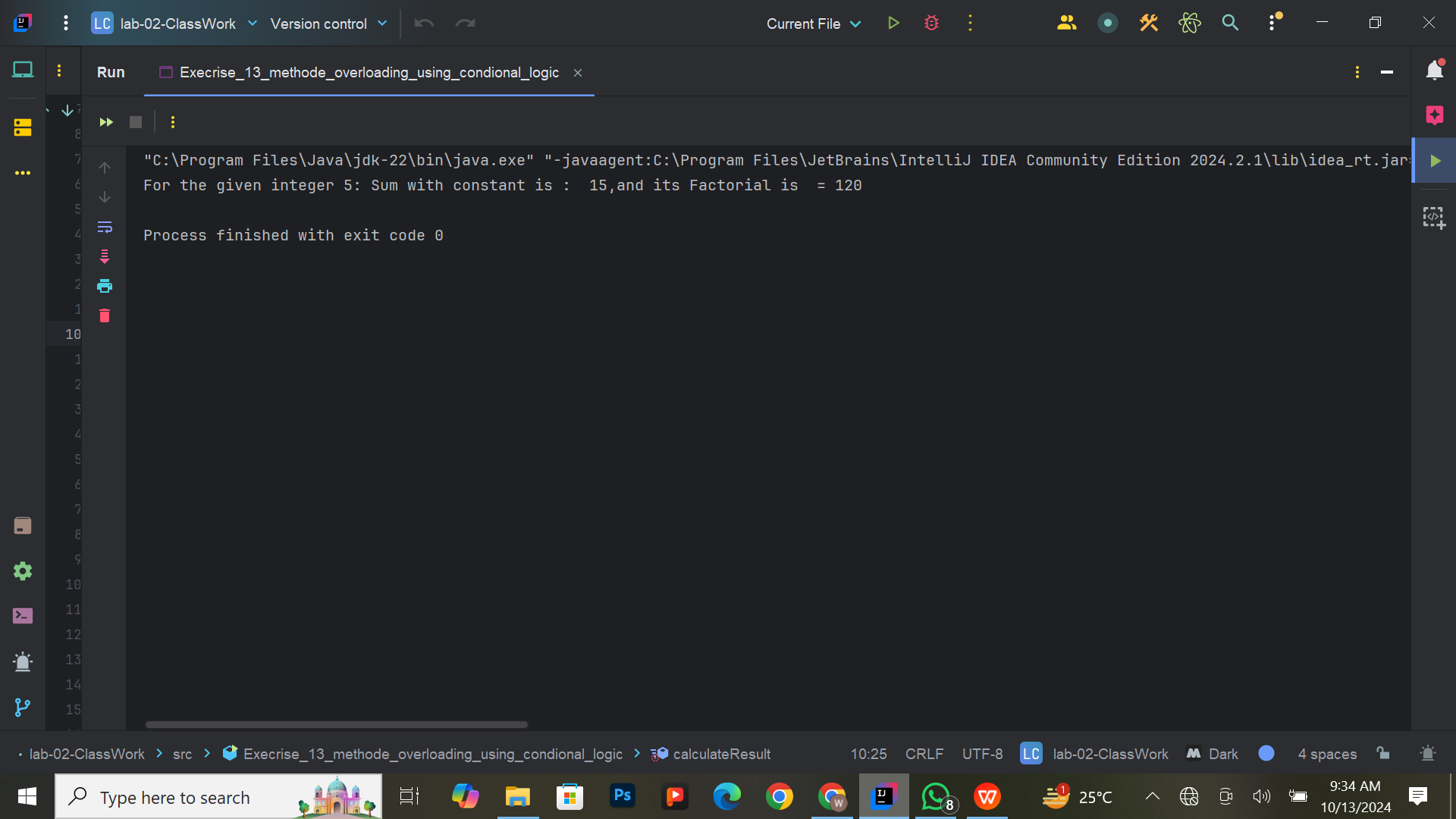
**Exercise 13: Combining Method Overloading with Conditional Logic**

**Task:**  
Create a method named calculateResult that is overloaded to handle both integers and doubles. Each version of the method should perform different calculations based on the input type. For integers, calculate the sum of the integer with a predefined constant and the factorial of that integer. For doubles, calculate the total cost after applying a discount and the total tax on that amount. In the main method, call the calculateResult method for each data type and print the results.

**Expected Output:**  
The program should output messages like:  
*“For integer 5: Sum with constant = 15, Factorial = 120”*  
*“For double 200.0: Total cost after discount = 180.0, Total tax = 9.0”*

**

**Output:**

**