Task1) Explanation:

The provided Java code performs multiplication of two numbers without using the multiplication operator (\*). It uses a Scanner to obtain input for two integers, initializes variables to store the numbers and the result, and prompts the user to input the first (A) and second (B) numbers. After closing the scanner, the program initializes the result to 0 and employs a for loop to simulate multiplication by repeatedly adding the value of A to the result for B iterations.

Finally, the program prints the result of the multiplication, incorporating the entered values for A and B into the output message.This code structure ensures user input, multiplication logic without using the multiplication operator, and clear output display of the result. The use of a loop to simulate multiplication highlights an alternative approach to arithmetic operations.Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

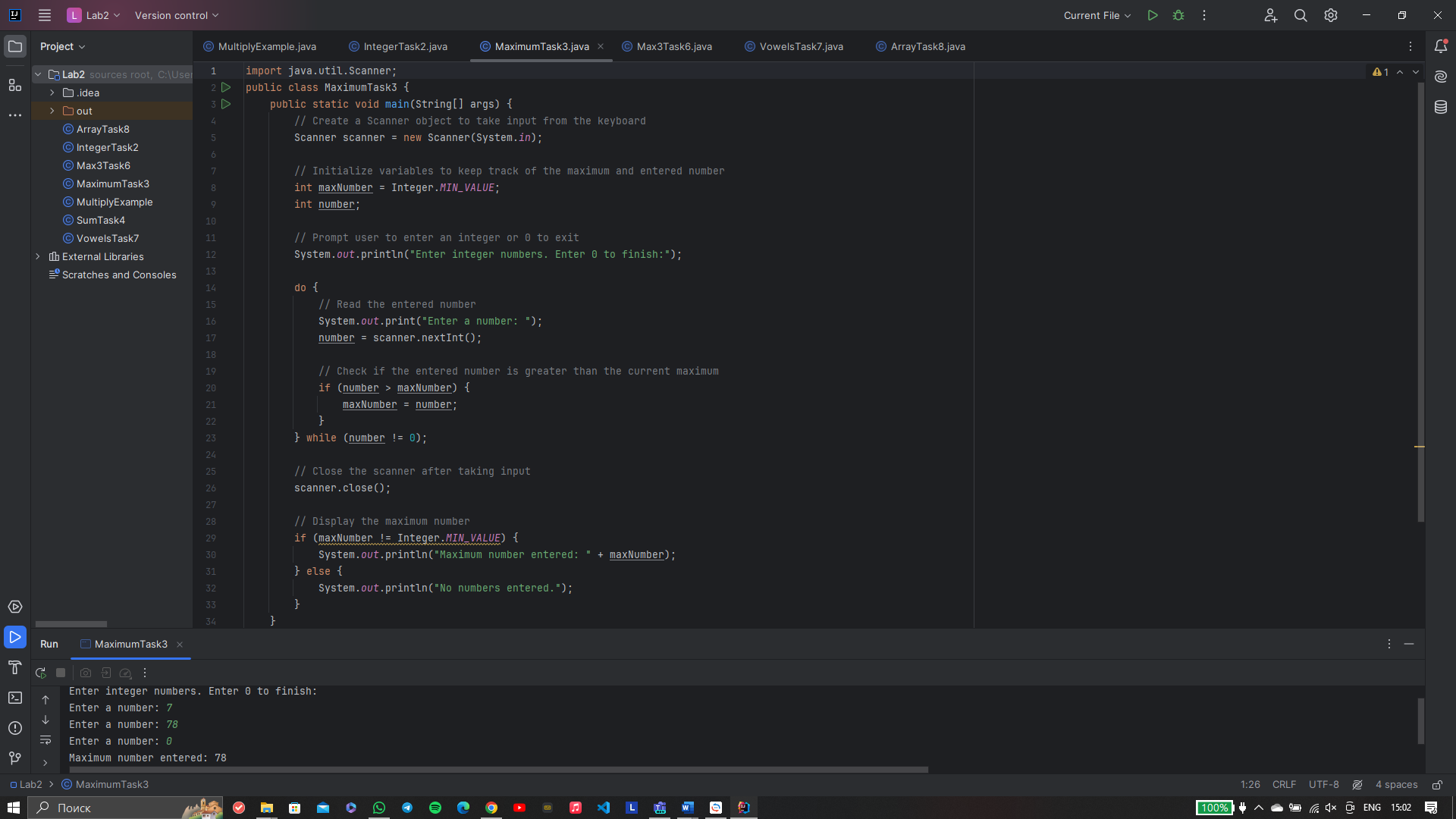
Task 2) Explanation:

Code calculates the sum and average of a series of user-entered integer numbers until 0 is input. It uses a Scanner to take input, initializes variables to keep track of the sum, count, and the entered number, and prompts the user to input integers until 0 is entered. The do-while loop continuously reads and processes input, updating the sum and count if the entered number is not 0.

After the loop, the Scanner is closed, and the program displays the overall amount of entered numbers. It then calculates and displays the average if at least one number was entered. If no numbers were entered, it outputs a message indicating so.This code structure ensures continuous input processing and provides clear information about the overall amount of entered numbers and their average. The handling of the count variable helps avoid division by zero when calculating the average.

Task 3) Explanation:

Task determines the maximum among a series of user-entered integer numbers until 0 is input. It uses a Scanner to take input, initializes variables to keep track of the maximum and the entered number, and prompts the user to input integers until 0 is entered. The do-while loop continuously reads and processes input, updating the maximum if the entered number is greater.

After the loop, the Scanner is closed, and the program displays the maximum number entered. If no numbers were entered (indicated by the initialized value of `maxNumber`), it outputs a message indicating so.This code structure ensures continuous input processing and accurately tracks the maximum number entered by the user. The handling of the initialized value of `maxNumber` accounts for the case when no valid numbers are provided by the user.

Task 4)Explanation:

The code calculates the sum of the first N odd numbers in a sequence. It begins by prompting the user to input a positive integer N, using a Scanner object for input. After closing the scanner, it checks if N is a positive integer; if not, it prompts the user to enter a positive integer. If N is positive, the program proceeds to calculate the sum of the first N odd numbers using a method called `computeSum`.

In the `computeSum` method, a variable `sum` is initialized to 0, and a variable `currentTerm` is set to 1. It then iterates N times, adding the current term to the sum and updating the current term by adding 2 in each iteration, simulating the generation of odd numbers. Finally, the program displays the computed sum for the user.The code structure ensures input validation and provides a clear method (`computeSum`) for the core logic of summing the first N odd numbers.

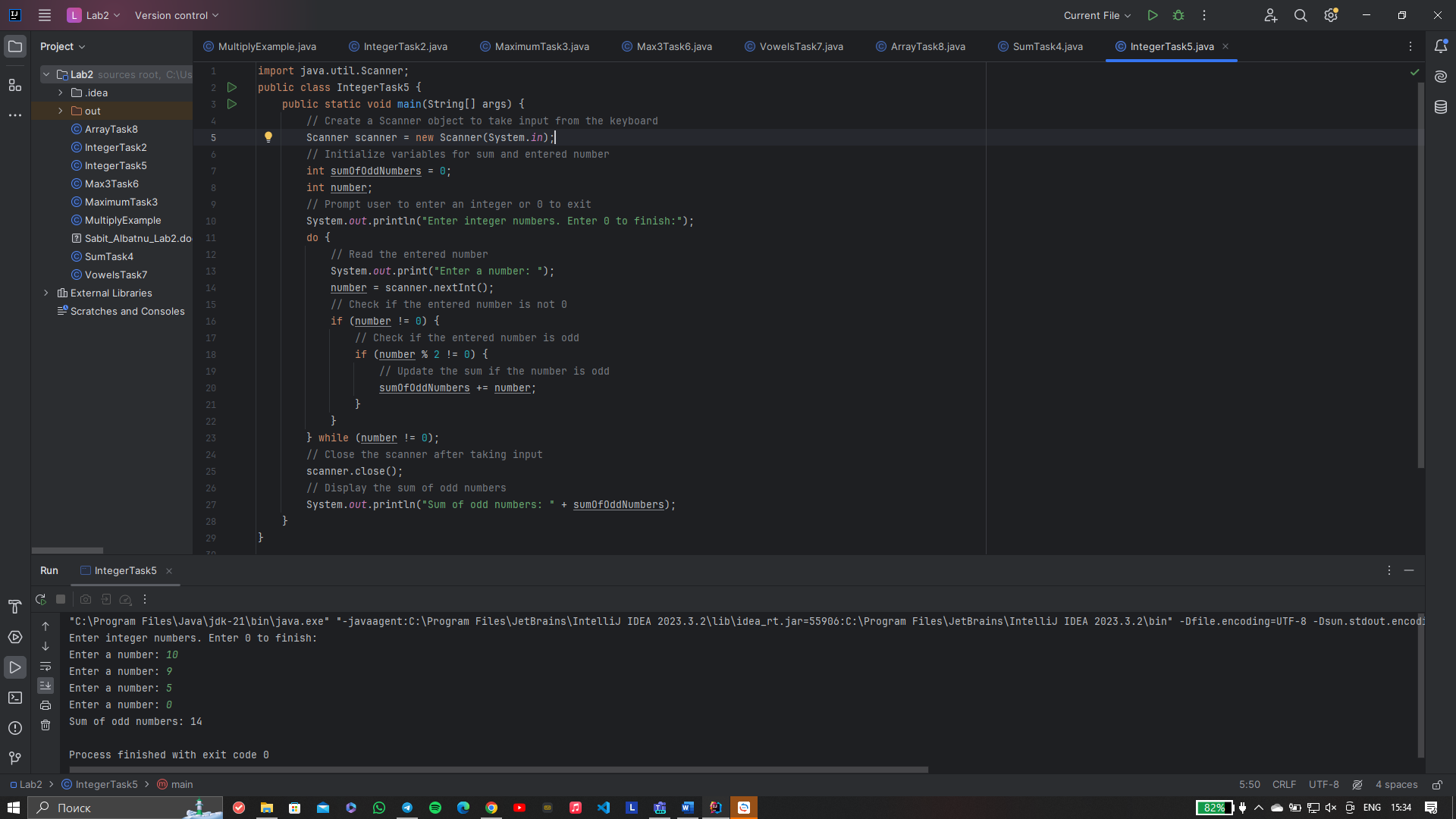
Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание

Task 5) Justification:

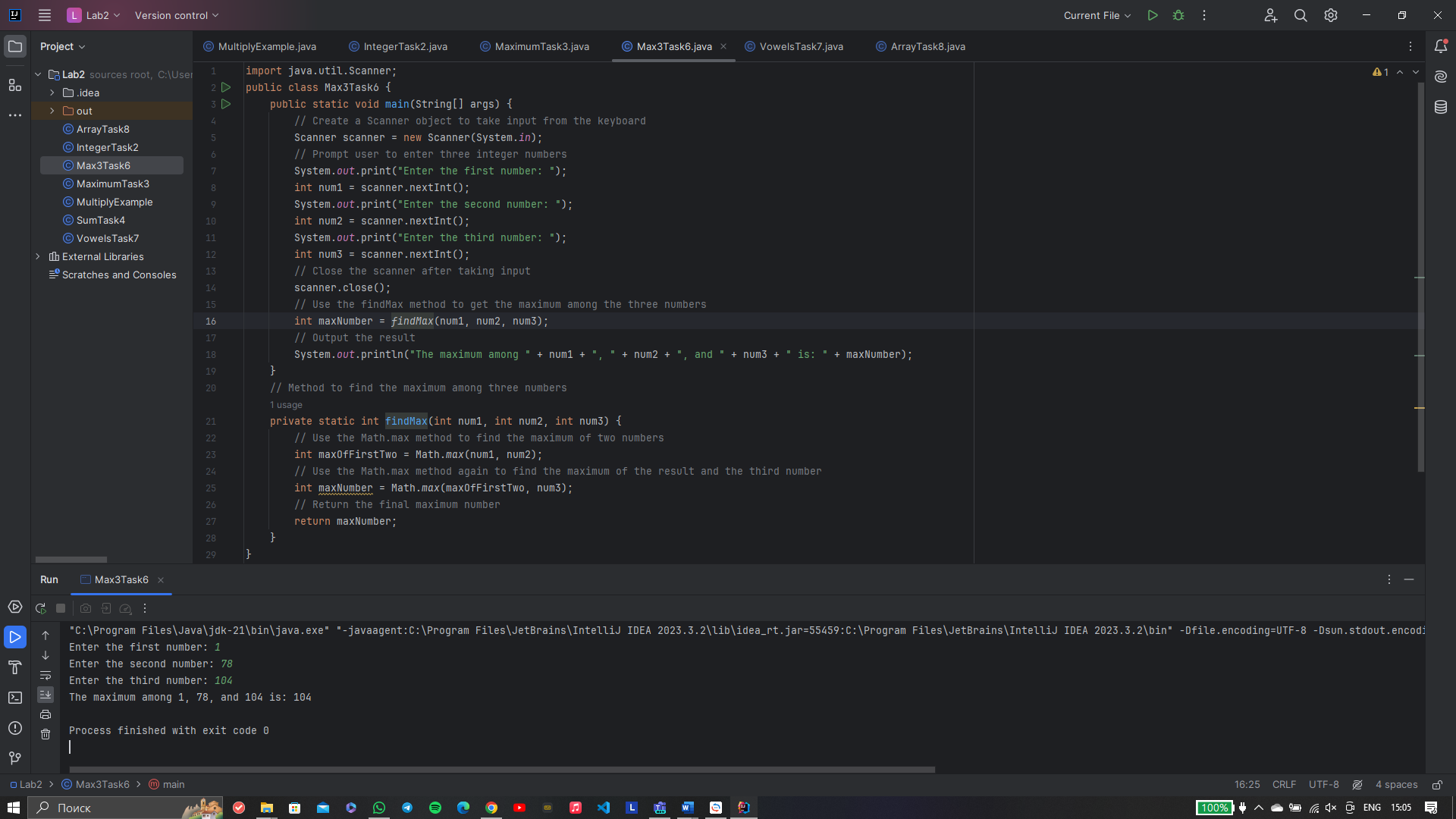
In this task, code calculates the sum of odd numbers entered by the user until 0 is input. It uses a Scanner to take input, initializes variables for the sum and the entered number, and prompts the user to input integers until 0 is entered. The do-while loop continuously reads and processes input, checking if the entered number is odd and updating the sum accordingly.

If the entered number is not 0 and is odd, it gets added to the `sumOfOddNumbers`. The loop continues until the user enters 0. After the loop, the Scanner is closed, and the program displays the sum of the entered odd numbers.This code ensures continuous input processing, updates the sum for odd numbers, and provides a clear user interface for entering integers and calculating their sum.



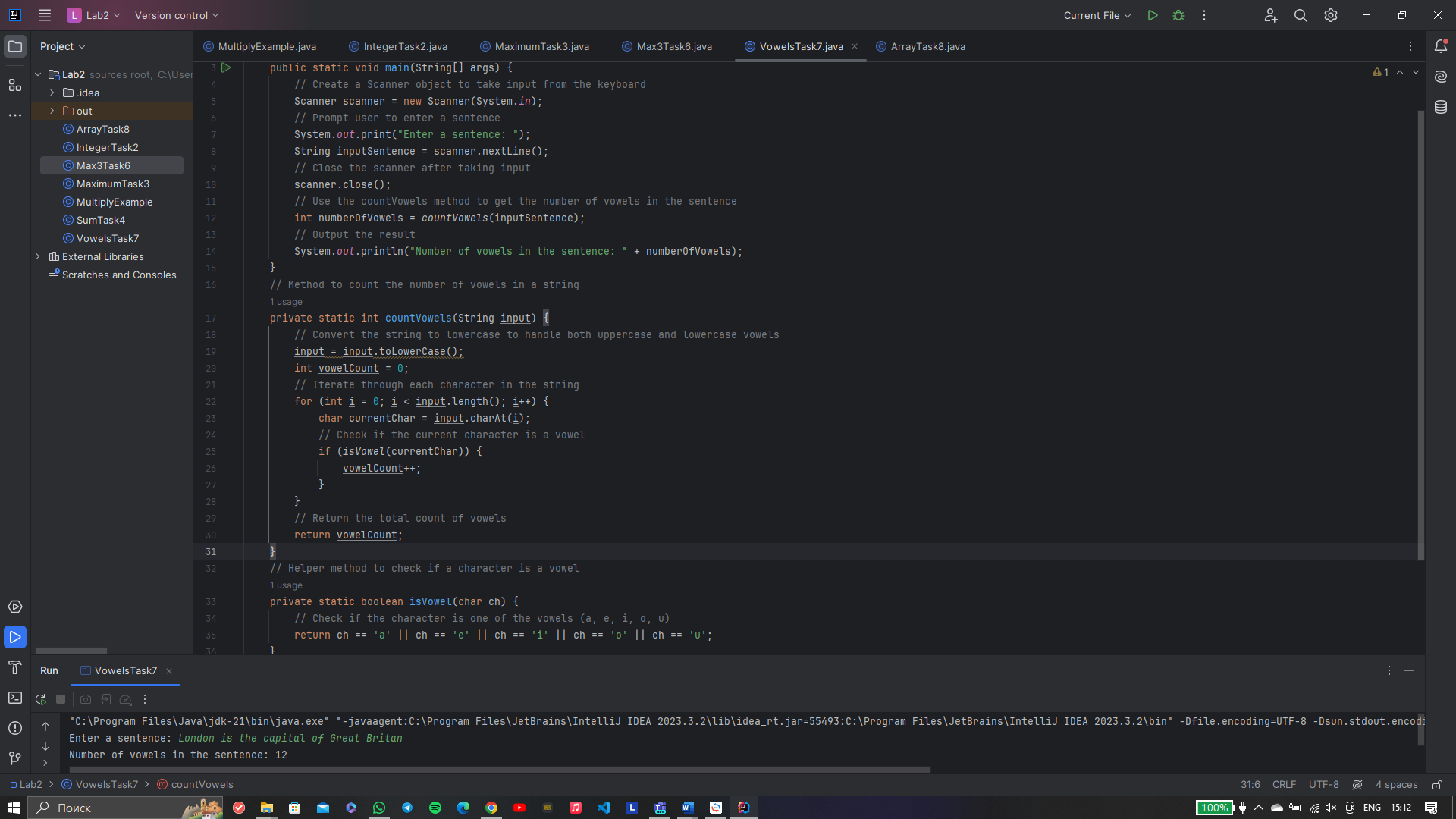
Task 6) Explanation:

It determines the maximum among three user-entered integer numbers. It uses a Scanner to obtain input for three integers, prompting the user accordingly. After closing the scanner, the program employs a method named `findMax` to calculate the maximum among the three input numbers.

The `findMax` method utilizes the `Math.max` function to find the maximum of two numbers. It first calculates the maximum of the first two input numbers and then uses the same method to find the maximum of that result and the third input number. The final maximum number is returned.The main method then outputs the result, displaying the maximum among the three input numbers. The code provides a clear structure for user input, method abstraction for maximum calculation, and concise output of the result.

Task 7) Explanation:

The code determines the number of vowels in a user-entered sentence. It utilizes a Scanner to obtain input for the sentence, prompting the user accordingly. After closing the scanner, the program employs a method named `countVowels` to calculate the number of vowels in the input sentence.

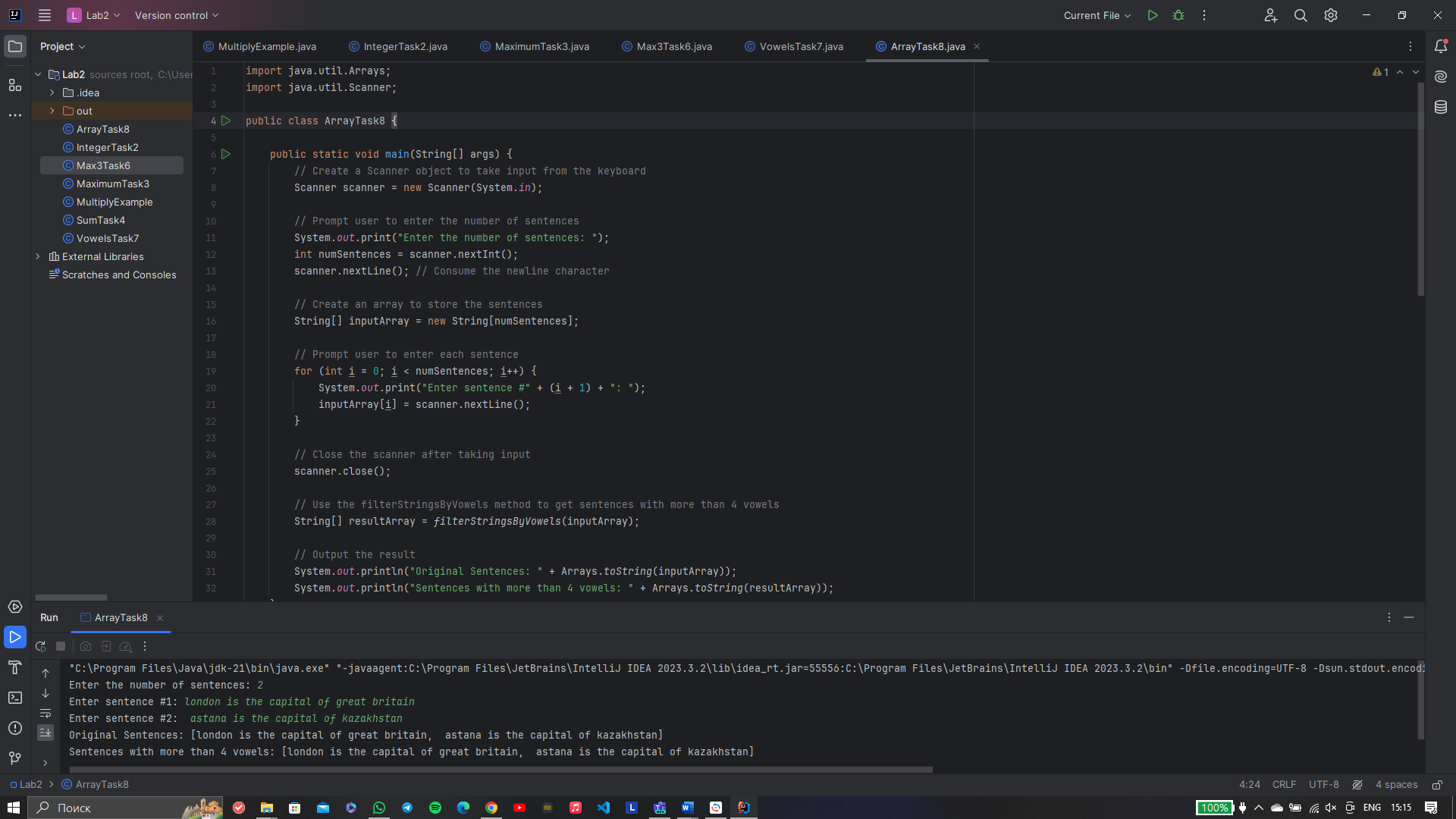
The `countVowels` method first converts the input sentence to lowercase to handle both uppercase and lowercase vowels consistently. It then iterates through each character in the string, utilizing a helper method `isVowel` to check if the current character is a vowel (a, e, i, o, or u). The count of vowels is incremented accordingly, and the total count is returned.The main method then outputs the result, displaying the number of vowels in the entered sentence. This code structure ensures a clear separation of concerns, with distinct methods for input, processing, and output.

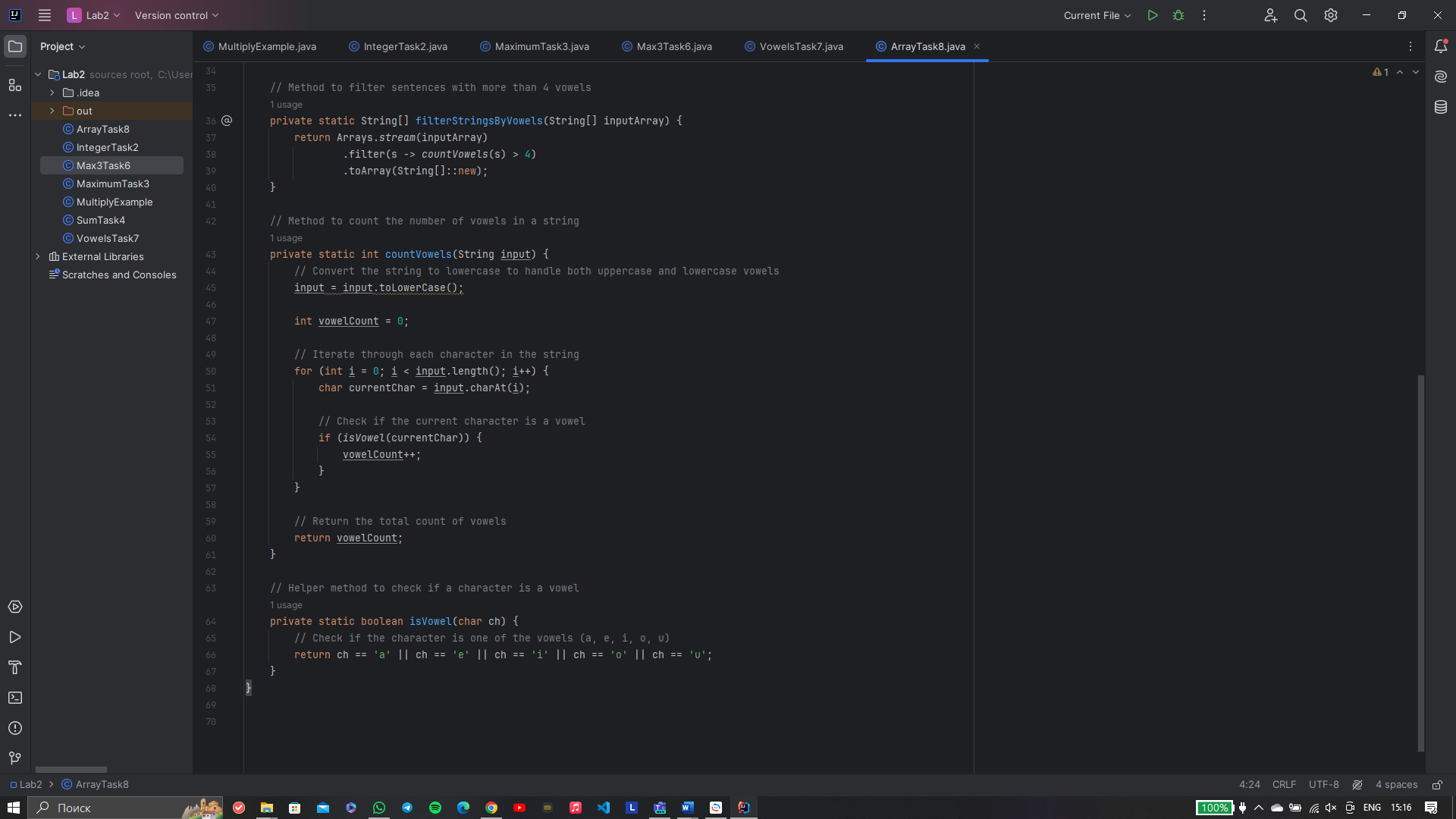
Task 8) Justification:

Task processes an array of sentences entered by the user. It uses a Scanner to obtain the number of sentences and each sentence itself. After closing the scanner, it creates an array to store the sentences, prompts the user to input each sentence, and then uses a method named `filterStringsByVowels` to extract sentences with more than 4 vowels.

The `filterStringsByVowels` method utilizes the Java Stream API to filter the input array based on the count of vowels in each sentence. The result array is then displayed, showing both the original sentences and the ones with more than 4 vowels.

The code structure ensures modularization with separate methods for input, processing, and output. It leverages the Stream API for efficient filtering and utilizes helper methods to handle vowel counting and checking.



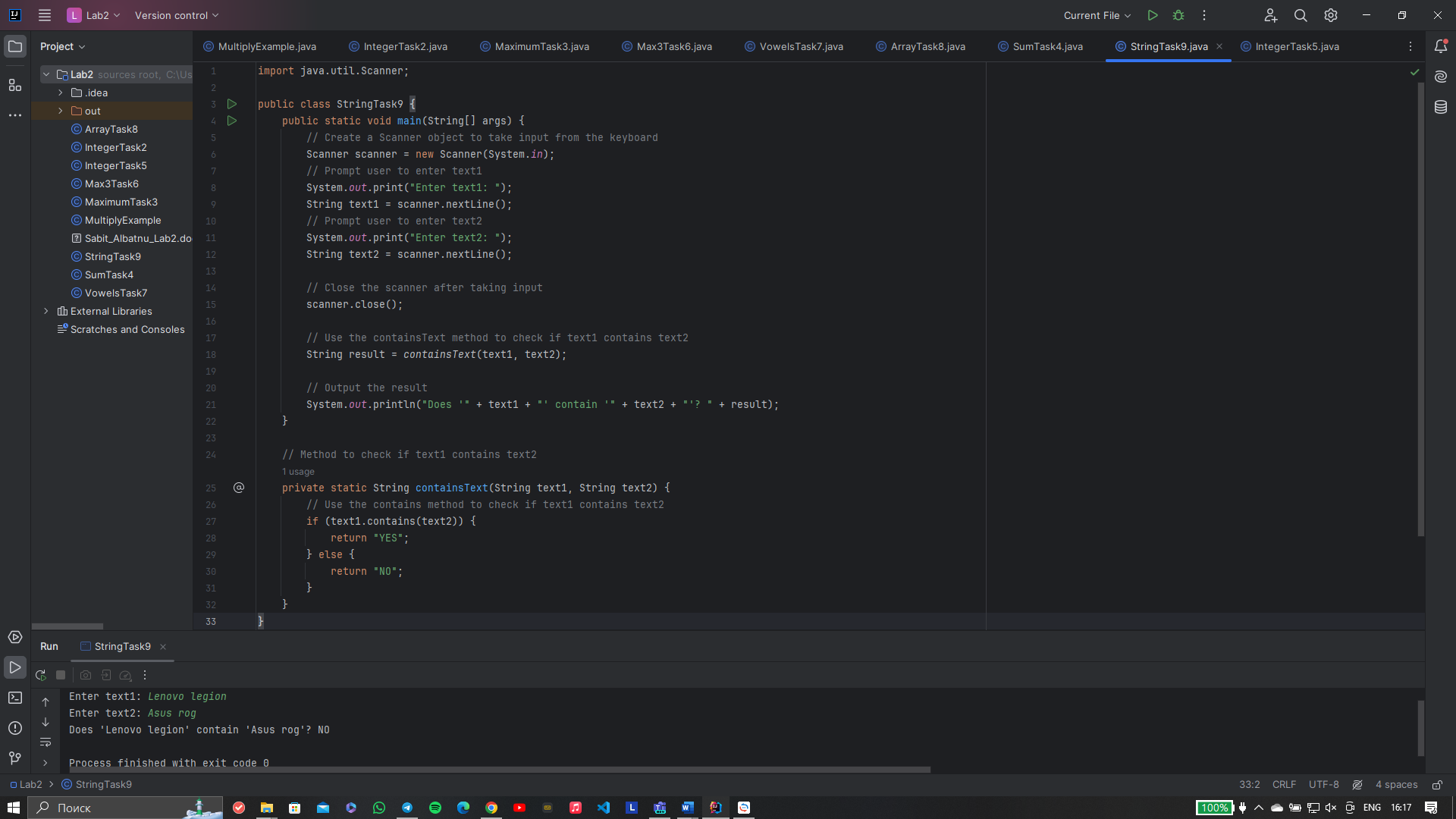


Task 9)Explanation:

The provided Java code checks whether the first entered text (`text1`) contains the second entered text (`text2`). It utilizes a Scanner to obtain input for both texts, prompts the user accordingly, and then uses a method named `containsText` to perform the containment check.

The `containsText` method uses the `contains` method of the String class to check if `text1` contains `text2`. The result is then displayed, indicating whether `text1` contains `text2` with a "YES" or "NO" response.

This code structure ensures a clear separation of concerns, with distinct methods for input, processing, and output. The `containsText` method abstracts the containment check, making the main method concise and easy to understand.



Task 10)Justification:

Code checks whether the entered string is a palindrome. It uses a Scanner to obtain input for the string, prompts the user accordingly, and then uses a method named `checkPalindrome` to perform the palindrome check.

The `checkPalindrome` method first removes spaces and converts the input string to lowercase to ensure a case-insensitive comparison. It then uses a `StringBuilder` to reverse the string and compares the original string with the reversed one. The result is displayed, indicating whether the entered string is a palindrome with a "YES" or "NO" response.

This code structure ensures a clear separation of concerns, with distinct methods for input, processing, and output. The `checkPalindrome` method abstracts the palindrome check logic, making the main method concise and easy to understand. The code efficiently handles case-insensitivity and space removal for accurate palindrome detection.

