JOBSHEET 9

ARRAY 1

DAVA RIZKY H

254107020246

Objective

- Students are able to understand one-dimensional Array creation and accessing its elements in Java
- Students are able to make programs using the concept of one-dimensional arrays

Laboratory

Experiment 1: Fill in Array Element

Experiment Time: 20 minutes

1. Open a text editor, create a new Java class with the name **arrayNumbersXX**. (XX=student ID number)
2. Write the basic structure of the Java programming language which contains the **main()** function
3. Create an array of integer type named **num** with a capacity of 4 elements

```
int[] num = new int[4];
```

4. Fill each element of the array with numbers 5, 12, 7, 20

```
num[0] = 5;
num[1] = 12;
num[2] = 7;
num[3] = 20;
```

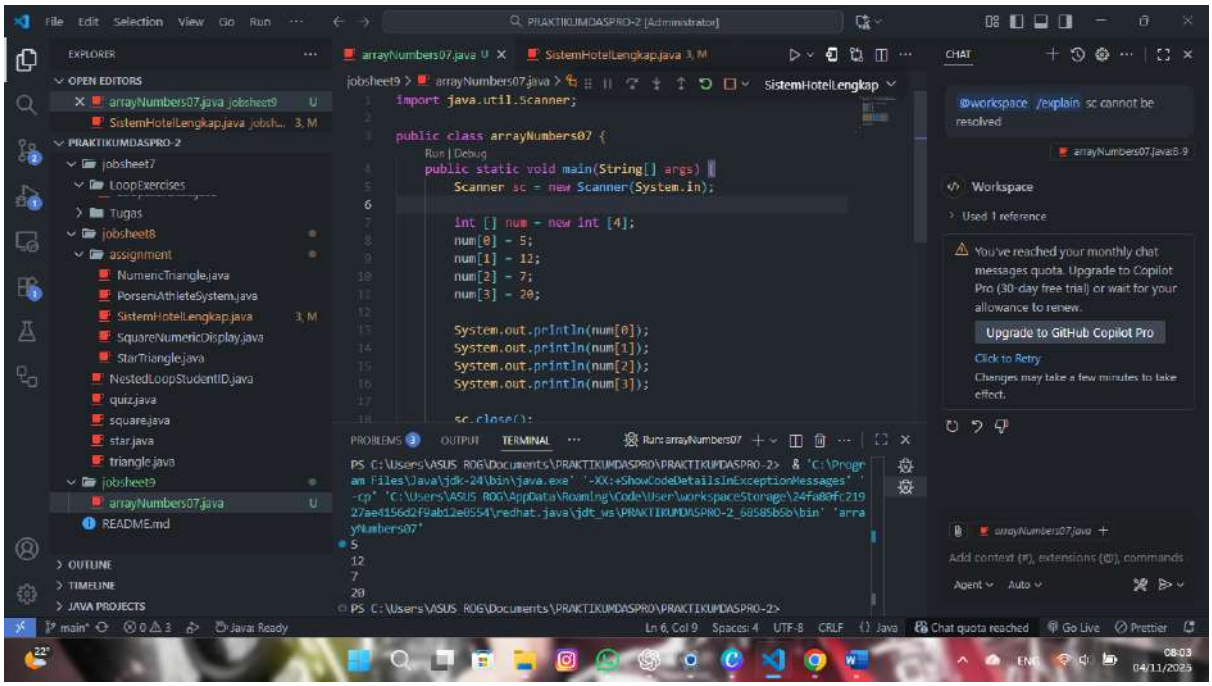
5. Display all contents of the elements to the screen

```
System.out.println(num[0]);
System.out.println(num[1]);
System.out.println(num[2]);
System.out.println(num[3]);
```

6. Compile and run the program. Match the results of the running programs that you have created according to the following display

```
5
12
7
20
```

7. Commit and push the changes to GitHub.



Questions!

1. If the contents of each element of the array **num** are changed with numbers 5.0, 12867,

   7.5, 2000000. What happens? How can it be like that?

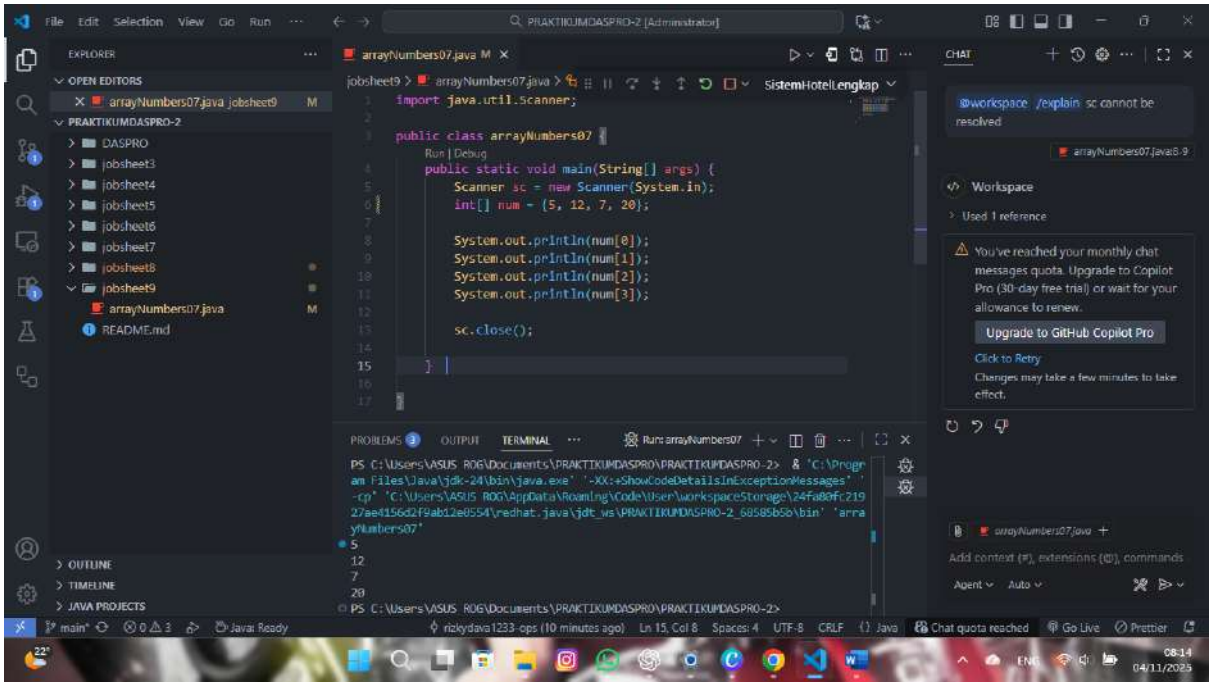
   The program will produce an **error** because the **int** data type cannot store decimal numbers such as 5.0 or 7.5. To use decimal values, the array data type must be changed to **double**.

   Correct example:

   double[] num = {5.0, 12867, 7.5, 2000000};

   Explanation:
   The error occurs due to a data type mismatch. Java cannot automatically convert a double value to int without explicit casting.

2. Modify the program code by initializing the array elements at the same time when declaring the array.

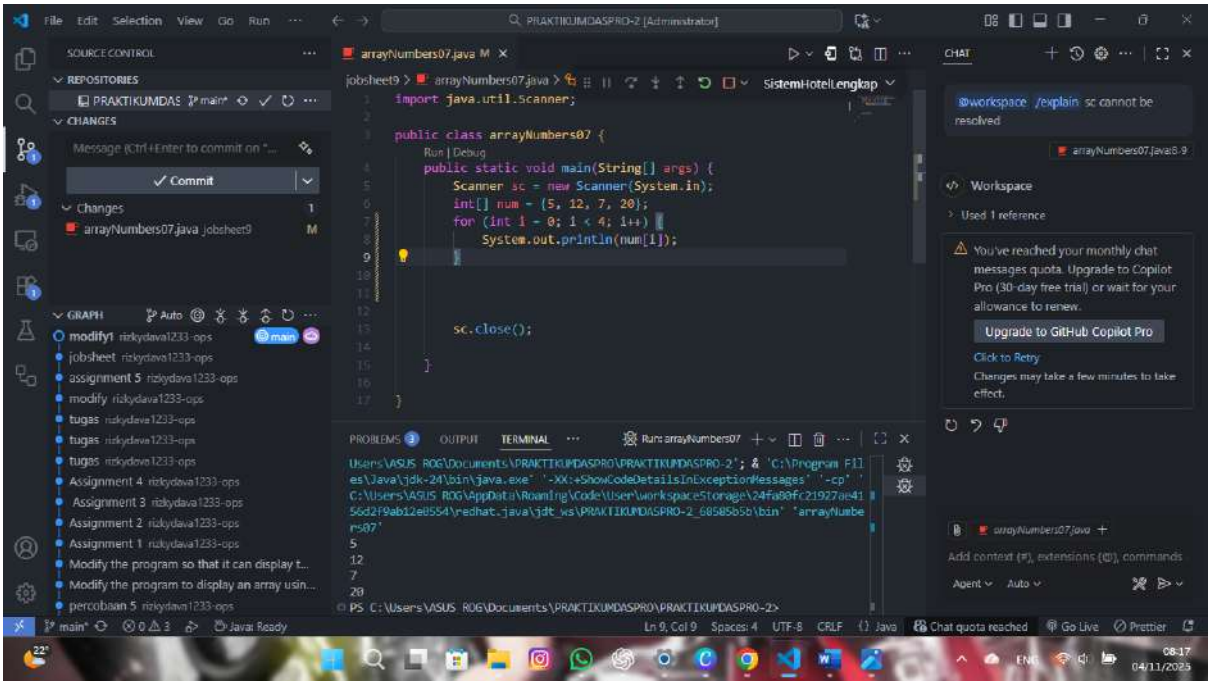


3. Change the statement in step 6 to be like this

```
for (int i = 0; i < 4; i++) {
    System.out.println(num[i]);
}
```

What is the result? How can it be like that?

The **for** loop prints all the elements of the array sequentially from index 0 to 3
The program displays every value stored in the array in order.



4. If the condition in the for-loop statement is changed to i <= 4, what is the output of the program? Why is the result like that?

Answer:
The program will cause a **runtime error** called **`ArrayIndexOutOfBoundsException`.**

Explanation:
The array only has indexes from 0 to 3 (four elements in total). When the loop condition is changed to `i <= 4`, the program tries to access `num[4]`, which does not exist, resulting in an exception.

5. Commit and push the changes to GitHub.

Experiment 2: Requesting User Input to Fill in an Array Element

Experiment Time: 40 minutes

1. Open a text editor, create a Java file then save it with the name **arrayValueXX**. (XX=student ID number)
2. Write the basic structure of the Java programming language which contains the **main()** function

3. Add the Scanner library
4. Create an array of integer type with the name **finalScore**, with a capacity of 10 elements

```
int[] finalScore = new int[10];
```

5. Using a loop, create an input to fill in the **finalScore** array element

```
for (int i = 0; i < 10; i++) {
    System.out.print("Enter the final score " + i + ": ");
    finalScore[i] = sc.nextInt();
}
```

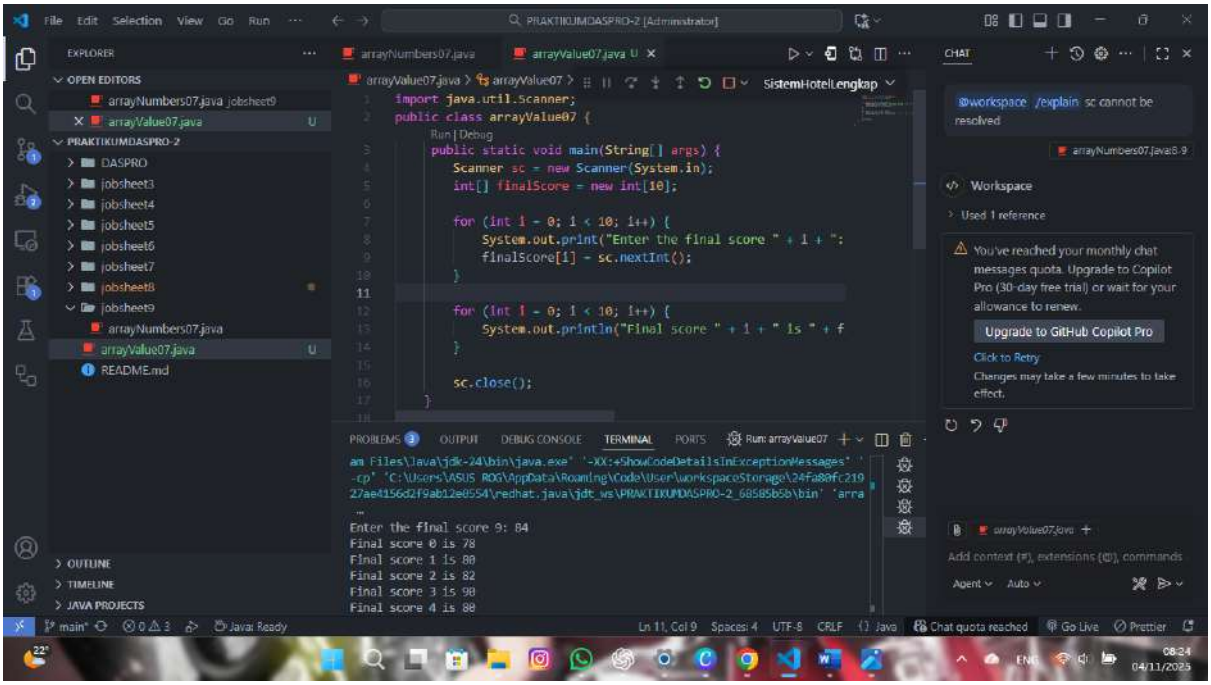
6. Using a loop, display all the contents of the elements from the **finalScore** array

```
for (int i = 0; i < 10; i++) {
    System.out.println("Final score " + i + " is " + finalScore[i]);
}
```

7. Compile and run the program. Match the results of the running programs that you have created according to the following display

```
Enter the final score 0: 78
Enter the final score 1: 89
Enter the final score 2: 94
Enter the final score 3: 85
Enter the final score 4: 79
Enter the final score 5: 87
Enter the final score 6: 93
Enter the final score 7: 72
Enter the final score 8: 86
Enter the final score 9: 91
Final score 0 is 78
Final score 1 is 89
Final score 2 is 94
Final score 3 is 85
Final score 4 is 79
Final score 5 is 87
Final score 6 is 93
Final score 7 is 72
Final score 8 is 86
Final score 9 is 91
```

8. Commit and push the changes to GitHub.

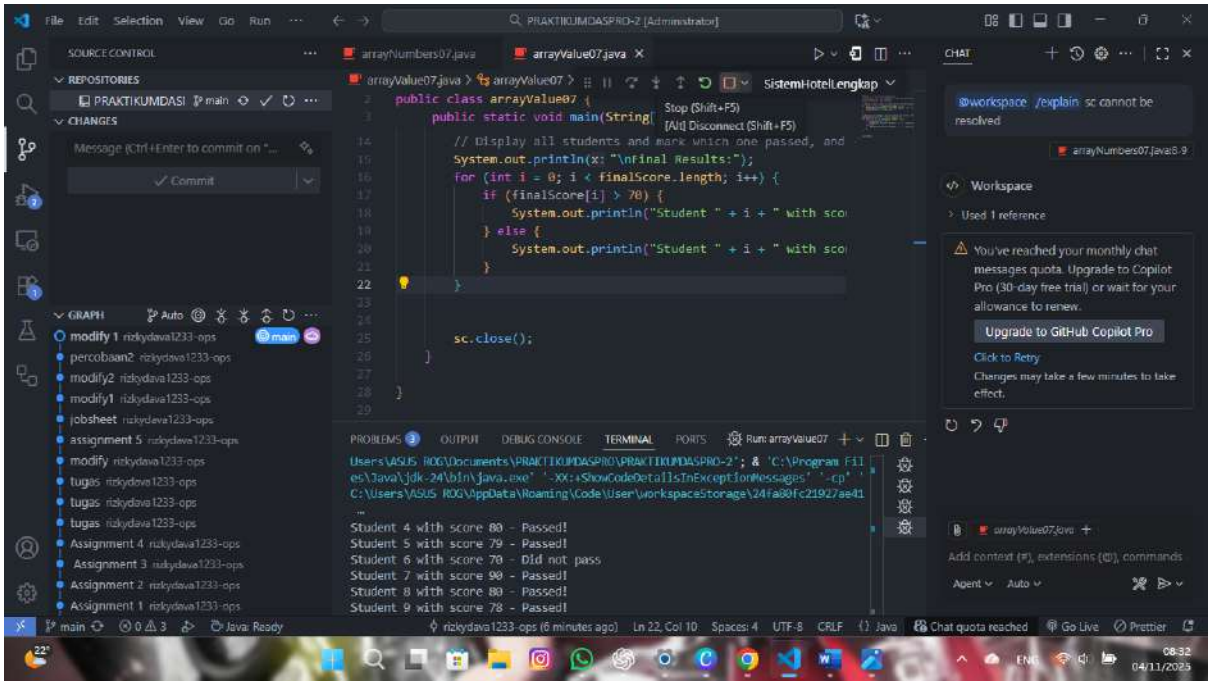


Questions!

1. Change the statement in step 5 to be like this

```
for (int i = 0; i < finalScore.length; i++) {
    System.out.print("Enter the final score " + i + ": ");
    finalScore[i] = sc.nextInt();
}
```

Run the program. Have there been any changes? How can it be like that?



2. Apa yang dimaksud dengan kondisi **i < finalScore.length**?

   finalScore.length is a property that returns the number of elements in the array

   For our array with 10 elements, finalScore.length equals 10

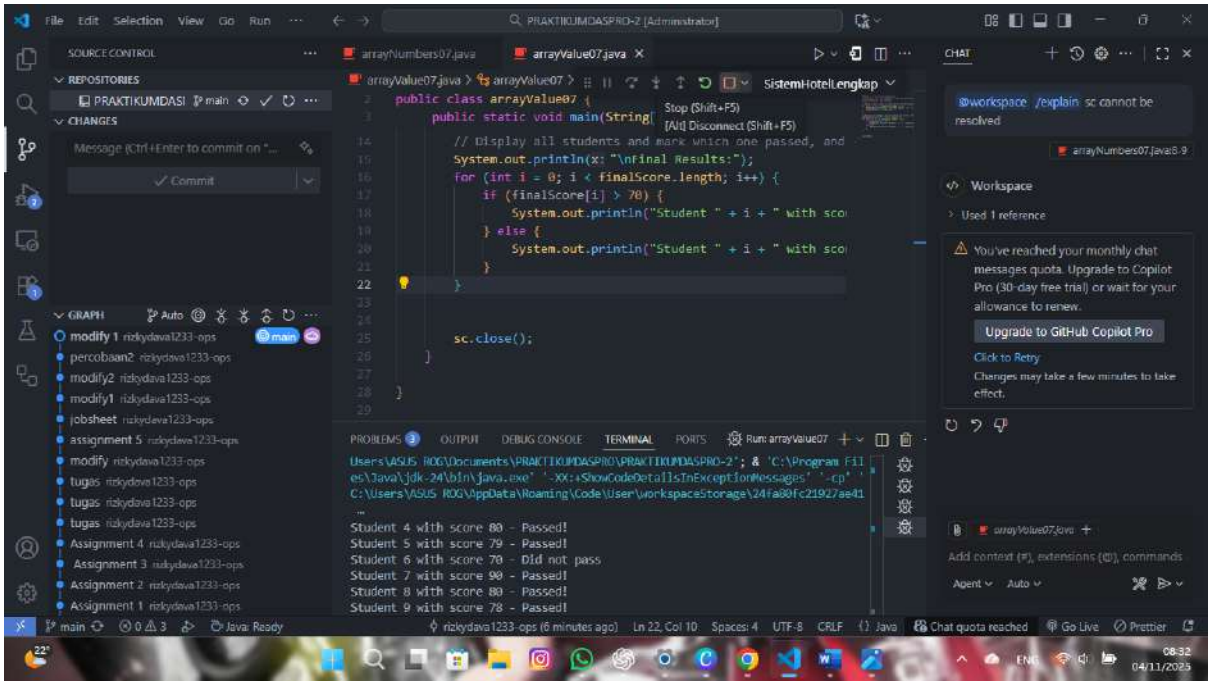
   The condition i < finalScore.length means "continue the loop while i is less than the array size"

   This ensures we access only valid indices: 0, 1, 2, ..., 9 (never 10, which would cause an error)

3. Change the statement in step 6 to be like this, so that the program only displays the grades of students who passed, students who have a score > 70

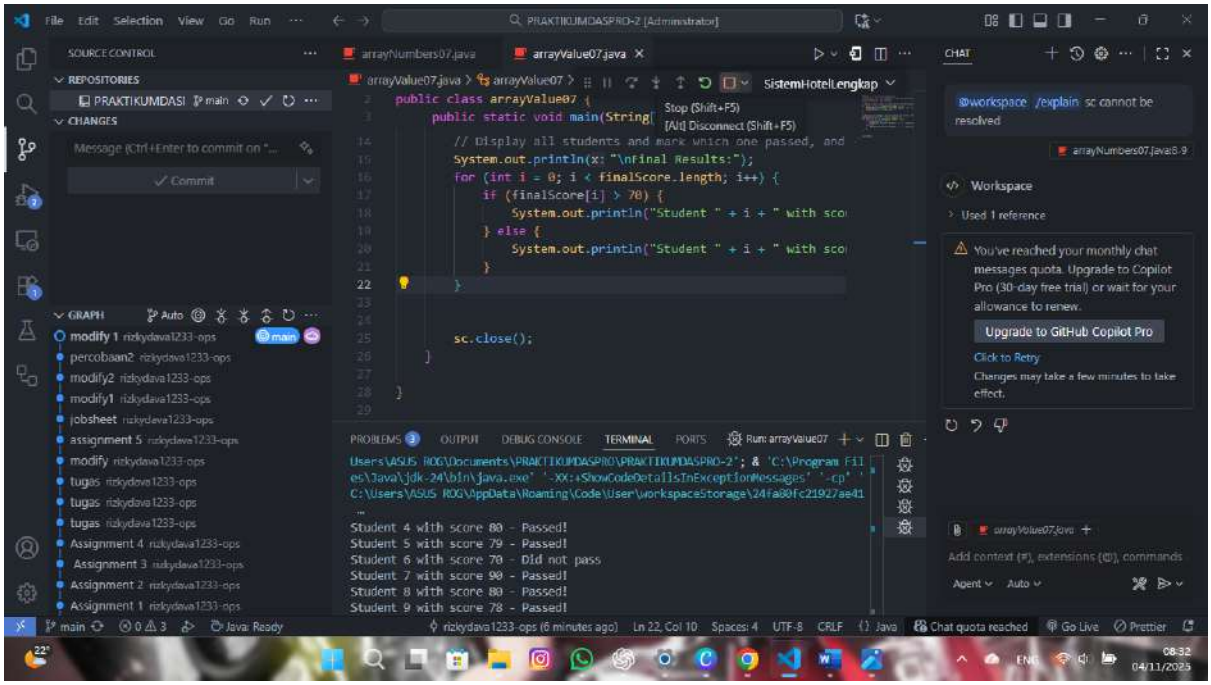
```
for (int i = 0; i < finalScore.length; i++) {
    if (finalScore[i] > 70) {
        System.out.println("Student " + i + " Passed!");
    }
}
```

   Run the program and describe the flow of the program!



4. Modify the program so that it displays all students, and mark which one passed, and which did not!

```
Enter the final score 0: 87
Enter the final score 1: 65
Enter the final score 2: 78
Enter the final score 3: 95
Enter the final score 4: 92
Enter the final score 5: 58
Enter the final score 6: 89
Enter the final score 7: 67
Enter the final score 8: 85
Enter the final score 9: 78
Student 0 Passed!
Student 1 Failed!
Student 2 Passed!
Student 3 Passed!
Student 4 Passed!
Student 5 Failed!
Student 6 Passed!
Student 7 Failed!
Student 8 Passed!
Student 9 Passed!
```



5. Commit and push the changes to GitHub.

2.3 Experiment 3: Perform Arithmetic Operations on Array Elements

Experiment Time: 75 minutes

This experiment is done to add array elements. The program will accept input of 10 student scores. Then the program will display the average score of 10 students.

1. Open a text editor, create a Java file then save it with the name **arrayAverageScore**XX.

   (XX=student ID number)

2. Write the basic structure of the Java programming language which contains the **main()** function
3. Add the Scanner library and make a **Scanner** declaration for input purposes
4. Create an array of integer types with the name **score** with a capacity of 10. Then declare the variables total and average

```
int[] score = new int[10];
double total = 0;
double average;
```

5. Using a loop, create an input to fill in the **score** array element

```
for (int i = 0; i < score.length; i++) {
    System.out.print("Enter student score " + (i + 1) + ": ");
    score[i] = sc.nextInt();
}
```

6. Using a loop, calculate the total number of scores.

```
for (int i = 0; i < score.length; i++) {
    total += score[i];
}
```

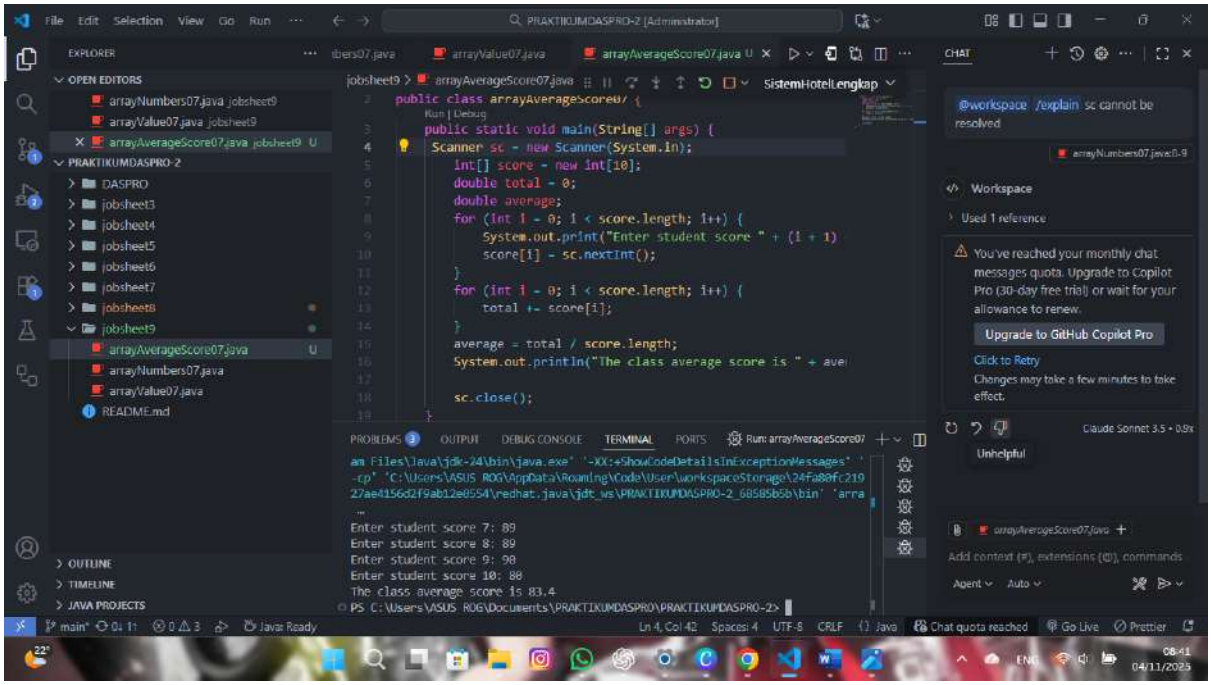
7. Calculate the average value by dividing **total** by the number of elements of **score**

```
average = total / score.length;
System.out.println("The class average score is " + average);
```

8. Compile and run the program. Match the results of the running programs that you have created according to the following display

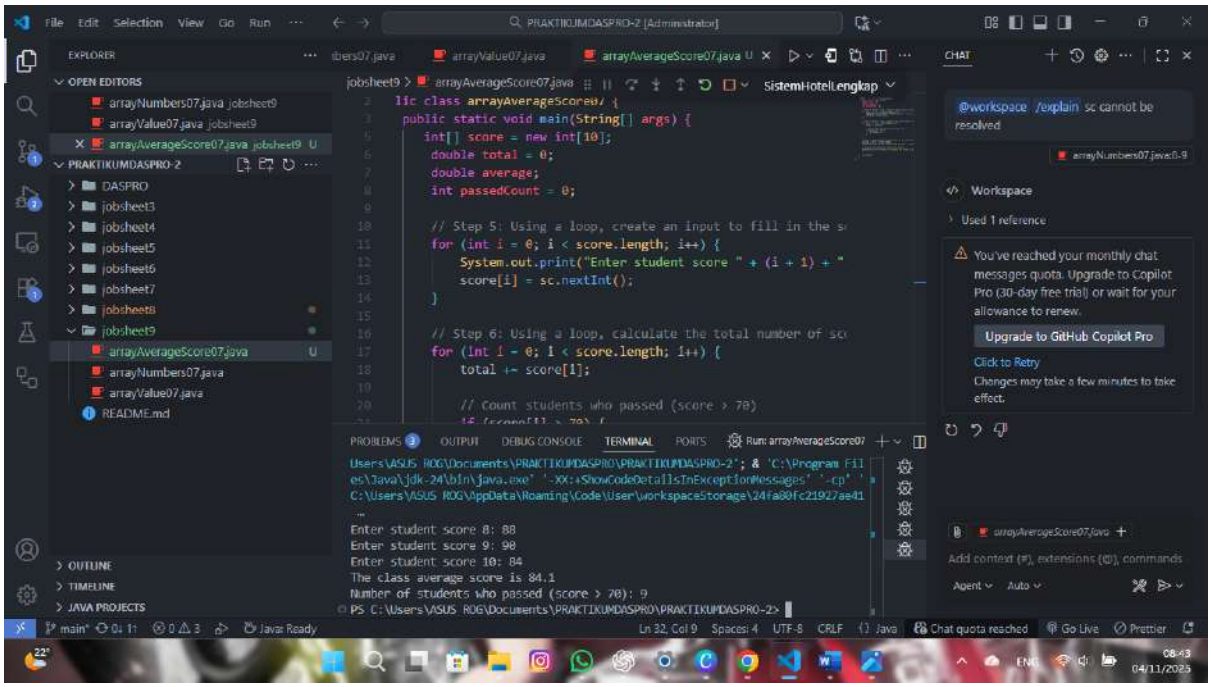
```
Enter student score 1: 98
Enter student score 2: 73
Enter student score 3: 86
Enter student score 4: 82
Enter student score 5: 95
Enter student score 6: 68
Enter student score 7: 90
Enter student score 8: 71
Enter student score 9: 78
Enter student score 10: 84
The class average score is 82.5
```

9. Commit and push the changes to GitHub



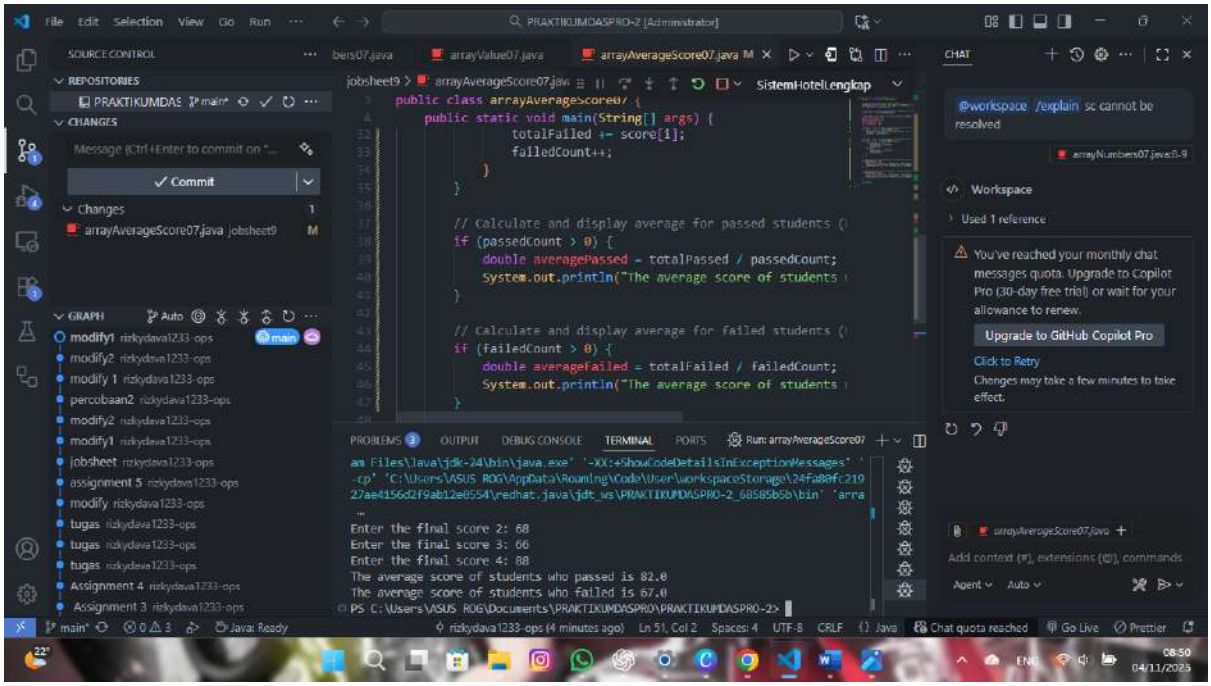
Questions!

1. Modify the program in Experiment 3 so that the program can display the number of students who passed, students who have a score greater than 70 (>70)



2. Modify the program in Experiment 3 so that it can produce output like the following display

```
Enter the number of students: 5
Enter the final score 0: 81
Enter the final score 1: 76
Enter the final score 2: 90
Enter the final score 3: 68
Enter the final score 4: 63
The average score of students who passed is 82.33333333333333
The average score of students who failed is 65.5
```



3. Commit and push the changes to GitHub

2.4 Experiment 4: Searching

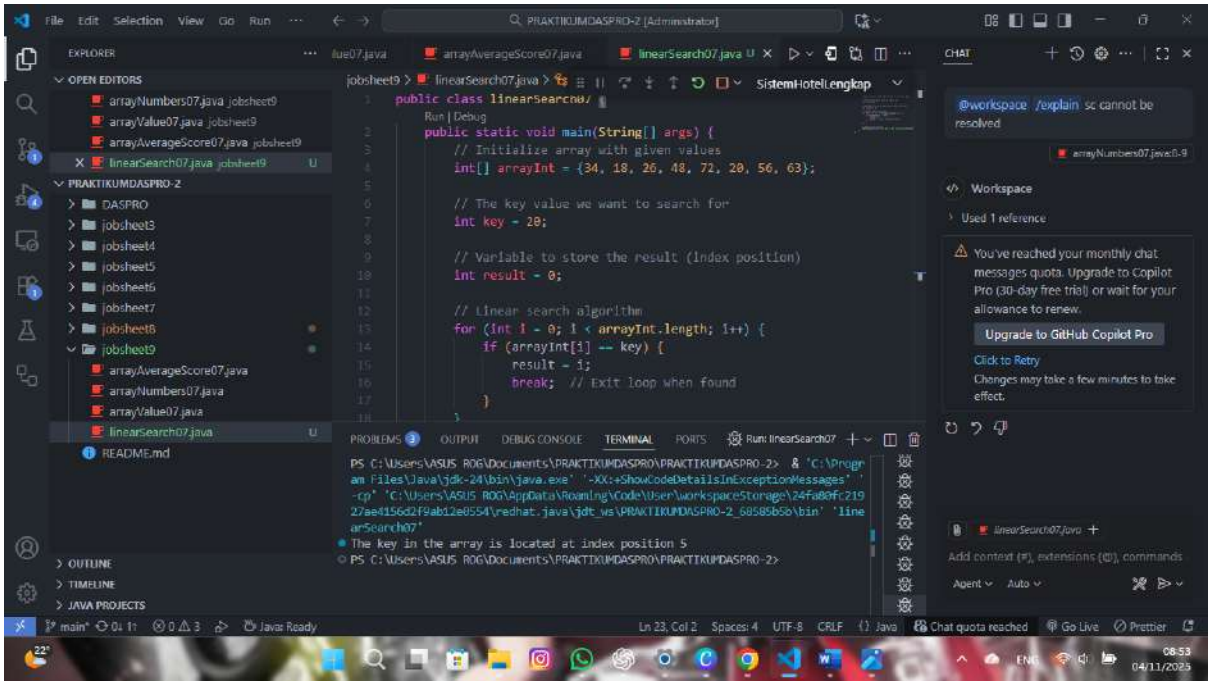
Experiment Time: 45 minutes

1. Open a text editor, create a Java file then save it with the name **linearSearch**XX. (XX=student ID number)
2. Add the following code

```java
    public static void main(String[] args) {
        int[] arrayInt = { 34, 18, 26, 48, 72, 20, 56, 63 };
        int key = 20;
        int result = 0;
        for (int i = 0; i < arrayInt.length; i++) {
            if (arrayInt[i] == key) {
                result = i;
                break;
            }
        }
        System.out.println("The key in the array is located at index position " + result);
    }
}
```

3. Compile and run the program. Match the results of the running programs that you have created according to the following display

```
The key in the array is located at index position 5
```



4. Commit and push the changes to GitHub

Questions!

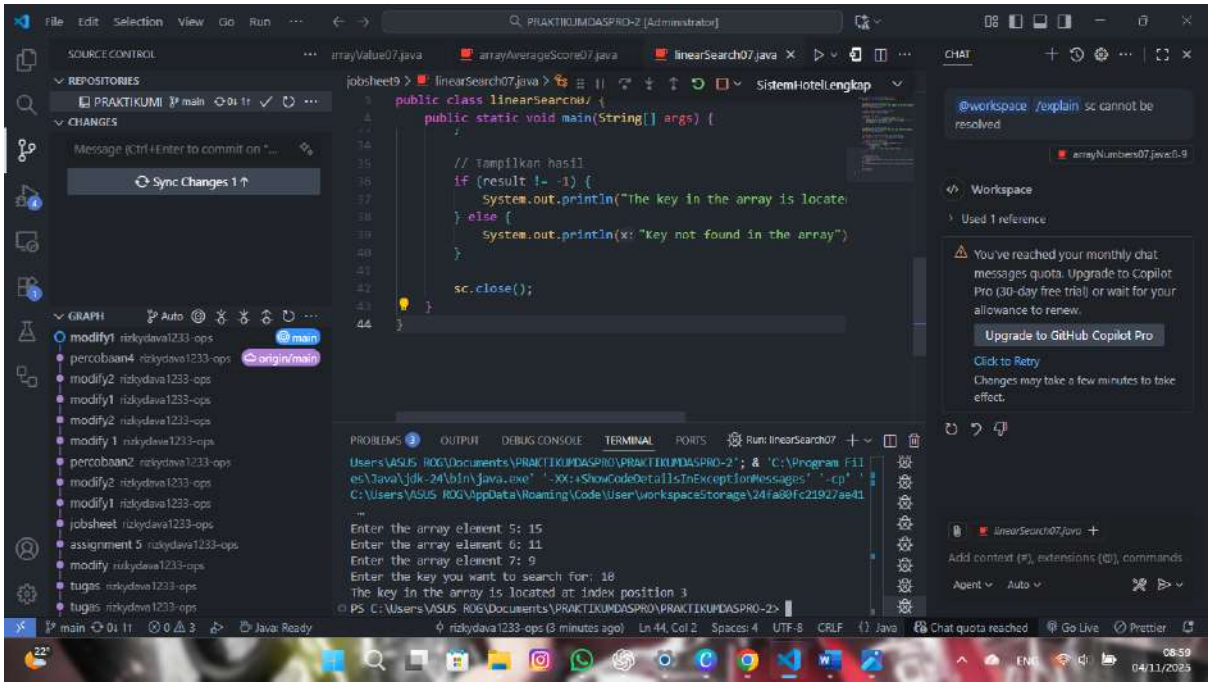
1. Explain the meaning of the **break;** statement on line 13 of the program code in Experiment 4.

The break; statement makes the linear search algorithm efficient by stopping the search as soon as the target value is found, rather than continuing through the entire array unnecessarily.

Think of it like looking for your keys: once you find them on the table, you don't keep searching in the drawers!

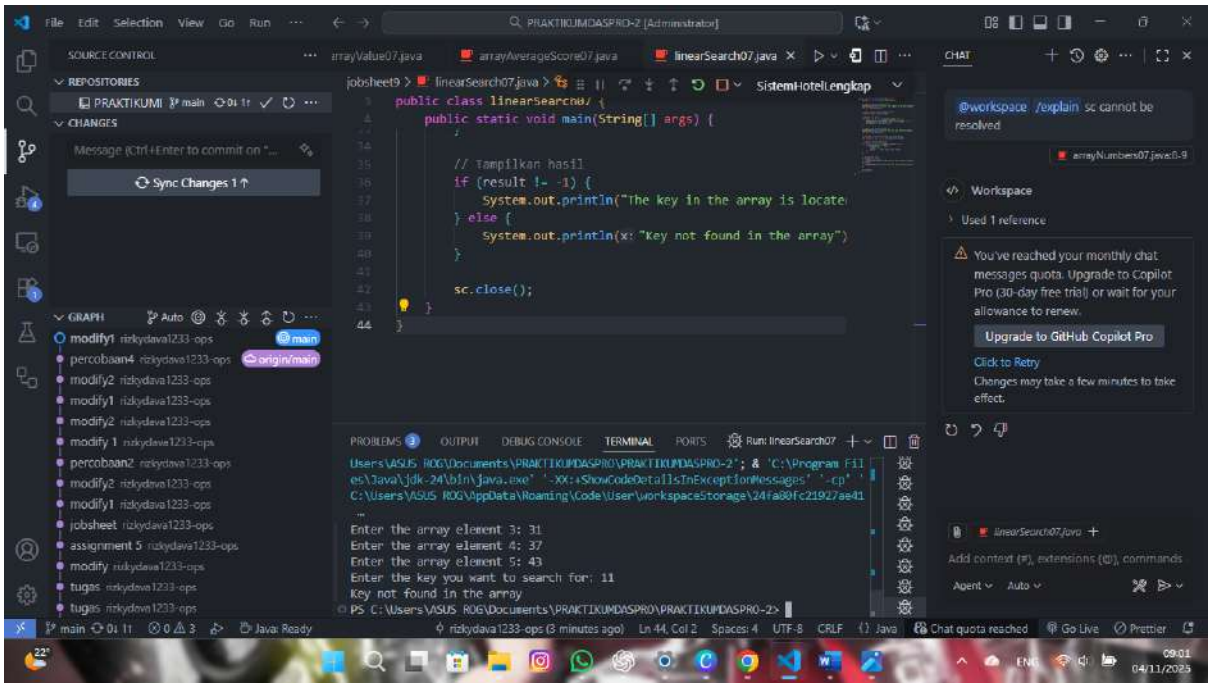
2. Modify the program code in experiment 4 so that the program can receive input in the form of the number of array elements, the contents of the array, and the key you want to search for. Then, print to the screen the index of the element positions of the searched key. Example of program results:

```
Enter the number of array elements: 8
Enter the array element 0: 12
Enter the array element 1: 18
Enter the array element 2: -6
Enter the array element 3: 10
Enter the array element 4: 6
Enter the array element 5: 15
Enter the array element 6: 11
Enter the array element 7: 9
Enter the key you want to search for: 10
The key in the array is located at index position 3
```



3. Modify the program in experiment 4 so that the program will give the message "key not found" if the key is not in the array. Example of program results:

```
Enter the number of array elements: 6
Enter the array element 0: 19
Enter the array element 1: 23
Enter the array element 2: 29
Enter the array element 3: 31
Enter the array element 4: 37
Enter the array element 5: 43
Enter the key you want to search for: 11
Key not found
```

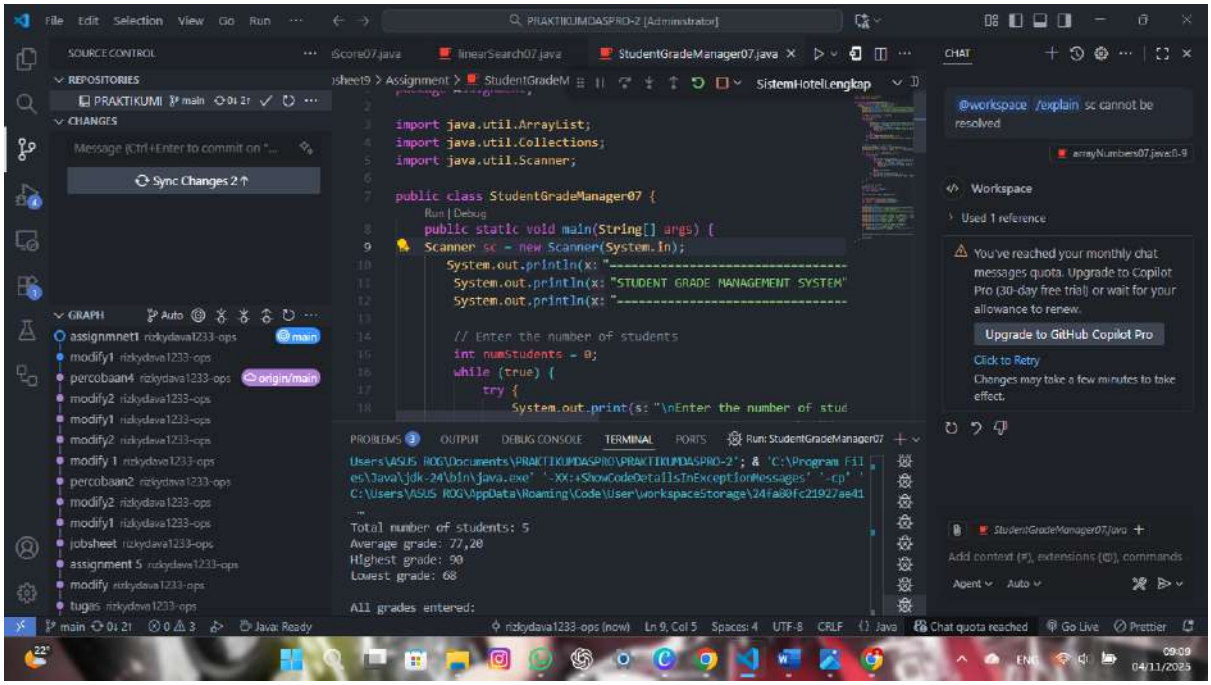


Assignment

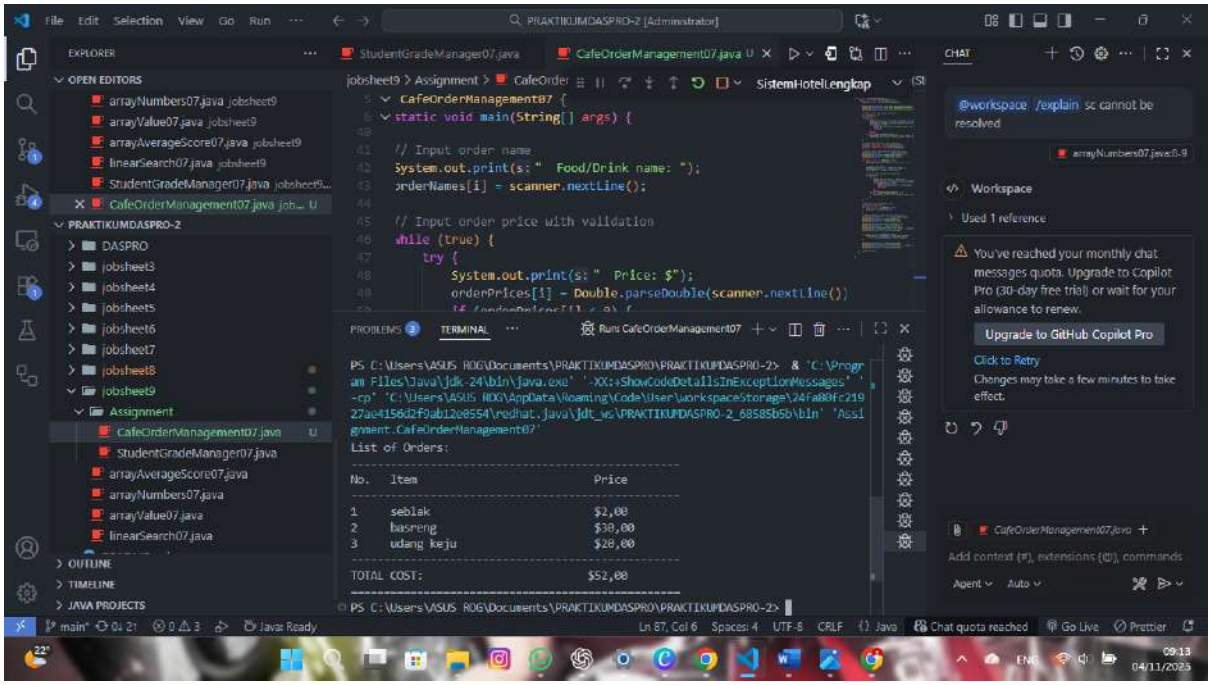
1. You are asked to create a program that can store and manage student grades.

   The grades are integers. The program must provide features for:

   - entering the number of student grades to be entered,
   - entering each student's grade,
   - calculating the average grade,
   - displaying the highest and lowest grades, and
   - displaying all grades entered.



2. Create a program that can manage food and beverage orders at a cafe. The program will allow users to enter orders, calculate the total cost of the order, and display a list of the orders that have been placed.
   - Input:
   - number of orders (input by the user).
   - name of the food/drink and the price for each order (input by the user).
   - Process:
   - store the order data in a one-dimensional array for the order names; and a separate one-dimensional array for the prices.
   - calculate the total cost of all orders entered.
   - display a list of orders that have been entered along with the total cost.
   - Output:
   - list of orders and the total cost of all orders.



3. Continuing with the example of ordering food at a cafe, create a program that allows users to order food from the cafe's menu. The program must store a list of food items in an array and provide an option to search for the desired item using a linear search method.

- Input:

- a predefined menu item in array form. The item names are initialized during the array declaration. For example:

```
String[] menu = {"Fried Rice", "Fried Noodles", "Toasted Bread", "Fried Potatoes", "Teh Tarik", "Cappuccino", "Chocolate Ice"};
```

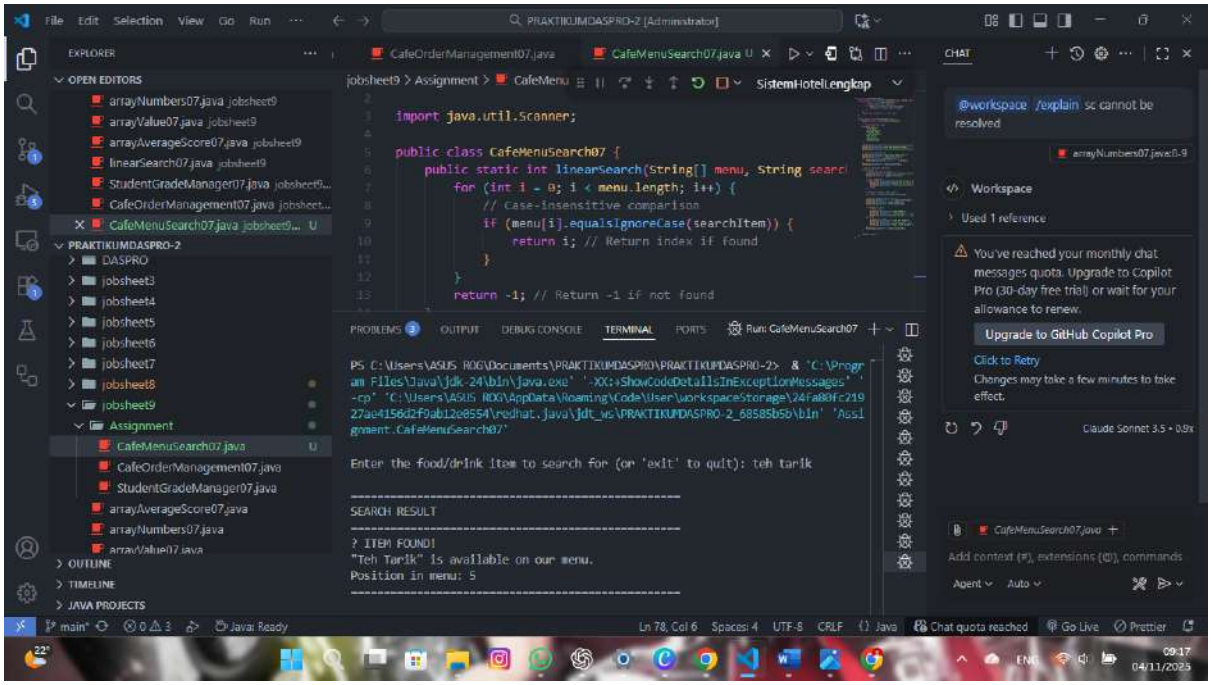
- the name of the item to be searched for (user input).

- Process:

- the program searches for the item entered by the user using a linear search algorithm.
- if the item is found, the program informs the user that it is available. If not, the program informs the user that the item is not on the menu.

- Output:

- display the search results to the user.


File Edit Selection View Go Run
PRAKTIKUMDASPRO-2 [Administrator]
EXPLORER
OPEN EDITORS
arrayNumbers07.java jobsheet9
arrayValue07.java jobsheet9
arrayAverageScore07.java jobsheet9
linearSearch07.java jobsheet9
StudentGradeManager07.java jobsheet9
CafeOrderManagement07.java jobsheet...
CafeMenuSearch07.java jobsheet9... U
PRAKTIKUMDASPRO-2
DASPRO
jobsheet3
jobsheet4
jobsheet5
jobsheet6
jobsheet7
jobsheet8
jobsheet9
Assignment
CafeMenuSearch07.java U
CafeOrderManagement07.java
StudentGradeManager07.java
arrayAverageScore07.java
arrayNumbers07.java
arrayValue07.java
OUTLINE
TIMELINE
JAVA PROJECTS
CafeOrderManagement07.java
CafeMenuSearch07.java U
jobsheet9 > Assignment > CafeMenu
SistemHotelLengkap
import java.util.Scanner;
public class CafeMenuSearch07 {
public static int linearSearch(String[] menu, String searc
for (int i = 0; i < menu.length; i++) {
// Case-insensitive comparison
if (menu[i].equalsIgnoreCase(searchItem)) {
return i; // Return index if found
}
}
return -1; // Return -1 if not found
PROBLEMS 3
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Run: CafeMenuSearch07
PS C:\Users\ASUS ROG\Documents\PRAKTIKUMDASPRO\PRAKTIKUMDASPRO-2> & 'C:\Progr
am Files\Java\jdk-24\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '
-cp' 'C:\Users\ASUS ROG\AppData\Roaming\Code\User\workspaceStorage\24fa80fc219
27ae4156d2f9ab12e0554\redhat.java\jdt_ws\PRAKTIKUMDASPRO-2_68585b5b\bin' 'Assi
gnment.CafeMenuSearch07'
Enter the food/drink item to search for (or 'exit' to quit): teh tarik
SEARCH RESULT
? ITEM FOUND!
"Teh Tarik" is available on our menu.
Position in menu: 5
CHAT
@workspace /explain sc cannot be resolved
Workspace
Used 1 reference
You've reached your monthly chat messages quota. Upgrade to Copilot Pro (30-day free trial) or wait for your allowance to renew.
Upgrade to GitHub Copilot Pro
Click to Retry
Changes may take a few minutes to take effect.
Claude Sonnet 3.5 • 0.9x
Add context (#), extensions (@), commands
Agent
Auto
main*
Java: Ready
Ln 78, Col 6
Spaces: 4
UTF-8
CRLF
Java
Chat quota reached
Go Live
Prettier
04/11/2025