



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.

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "PRAKTIKUMDASPRO-2". The "jobsheet9" folder contains "arrayNumbers07.java" and "SistemHotelLengkap.java".
- Code Editor:** Displays the Java code for "arrayNumbers07.java". The code initializes an array `num` with values 5, 12, 7, and 20, then prints each element.
- Terminal:** Shows the command run in the terminal: `PS C:\Users\VASUS\ROG\Documents\PRAKTIKUMDASPRO\PRAKTIKUMDASPRO-2> & "C:\Program Files\Java\jdk-24\bin\java.exe" -XX:+ShowCodeDetailsInExceptionMessages -cp "C:\Users\VASUS\ROG\AppData\Roaming\Code\User\workspaceStorage\24fa80fc21927ae4156d219ab02e0554\redhat\java\jdt_ws\PRAKTIKUMDASPRO-2_685bb0c\bin" arrayNumbers07`.
- Output:** Shows the output of the program: 5, 12, 7, 20.
- Problems:** Shows a warning about the use of the `sc` variable.
- Right Panel:** Shows GitHub integration with a message about reaching monthly chat limits and options to upgrade or retry.

Questions!

- 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.

The screenshot shows a Java application named "arrayNumbers07.java" open in the editor. The code prints four integer values from an array. The GitHub Copilot feature is active, with a message about reaching a quota and an upgrade option.

```
import java.util.Scanner;

public class arrayNumbers07 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[] num = {5, 12, 7, 20};

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

        sc.close();
    }
}
```

PROBLEMS

```
PS C:\Users\VSUS ROG\Documents\PRAKTIKUMDASPRO\PRAKTIKUMDASPRO-2> & 'C:\Program Files\Java\jdk-14\bin\java.exe' -XX:+ShowCodeDetailsInExceptionMessages -cp 'C:\Users\VSUS ROG\AppData\Roaming\Code\User\workspaceStorage\%2f4e80f-2192-7ae4156d2f9ab2e0554\redhat\java\jdt_vs\PRAKTIKUMDASPRO-2_69585b5\bin' 'arrayNumbers07'
● 5
● 12
● 7
● 20
```

OUTPUT

```
PS C:\Users\VSUS ROG\Documents\PRAKTIKUMDASPRO\PRAKTIKUMDASPRO-2>
```

CHAT

@workspace /explain sc cannot be resolved

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.

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.



```

1 import java.util.Scanner;
2
3 public class arrayNumbers07 {
4     Run | Debug
5     public static void main(String[] args) {
6         Scanner sc = new Scanner(System.in);
7         int[] num = {5, 12, 7, 20};
8         for (int i = 0; i < 4; i++) {
9             System.out.println(num[i]);
10        }
11    }
12    sc.close();
13 }
14
15
16
17

```

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.



```

    arrayNumbers07.java  arrayValue07.java X
    import java.util.Scanner;
    public class arrayValue07 {
        public static void main(String[] args) {
            Scanner sc = new Scanner(System.in);
            int[] finalScore = new int[10];
            for (int i = 0; i < 10; i++) {
                System.out.print("Enter the final score " + i + ": ");
                finalScore[i] = sc.nextInt();
            }
            for (int i = 0; i < 10; i++) {
                System.out.println("Final score " + i + " is " + finalScore[i]);
            }
            sc.close();
        }
    }

```

The screenshot shows the IntelliJ IDEA interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Run, ...
- Toolbar:** Standard toolbar icons.
- Explorer:** Shows the project structure under 'PRAKTIKUMDASPRO-2'. It includes files like 'arrayNumbers07.java', 'arrayValue07.java', 'README.md', and several 'jobsheet' files (jobsheet1 through jobsheet9).
- Editor:** Displays the Java code for 'arrayValue07.java'.
- Output:** Shows the terminal output of the program execution, which reads 10 scores from the user and prints them back.
- Bottom Bar:** Includes tabs for 'main*', 'Java Ready', and system status indicators like battery level, signal strength, and date/time (04/11/2025).

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?



```

public class arrayValue07 {
    public static void main(String[] args) {
        // Display all students and mark which one passed, and
        System.out.println("Final Results:");
        for (int i = 0; i < finalScore.length; i++) {
            if (finalScore[i] > 70) {
                System.out.println("Student " + i + " with score " + finalScore[i] + " Passed!");
            } else {
                System.out.println("Student " + i + " with score " + finalScore[i] + " Did not pass");
            }
        }
        sc.close();
    }
}

```

The screenshot shows the Visual Studio Code interface with the Java file `arrayValue07.java` open. The code prints the final results of student scores. The output window shows the results for 10 students, with Student 4, 5, 6, 7, 8, and 9 being marked as 'Passed!' and Student 0, 1, 2, 3, and 10 as 'Did not pass'. A GitHub Copilot Pro message is visible on the right side of the interface.

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!



```

1  public class arrayValue07 {
2      public static void main(String[] args) {
3          // Display all students and mark which one passed, and
4          System.out.println("Final Results:");
5          for (int i = 0; i < finalScore.length; i++) {
6              if (finalScore[i] > 70) {
7                  System.out.println("Student " + i + " with score " +
8                      finalScore[i] + " - Passed!");
9              } else {
10                  System.out.println("Student " + i + " with score " +
11                      finalScore[i] + " - Failed!");
12              }
13          }
14          sc.close();
15      }
16  }

```

The screenshot shows the Microsoft Visual Studio Code interface. On the left, there's a Source Control sidebar with repositories like 'PRAKTIKUMDASI' and changes. The main editor window contains Java code for reading student scores from a file and printing them with their status ('Passed!' or 'Failed!'). The terminal at the bottom shows the output of running the program. A GitHub Copilot Pro banner is visible on the right.

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!

```



```

1 public class arrayValue07 {
2     public static void main(String[] args) {
3         // Display all students and mark which one passed, and
4         System.out.println("Final Results:");
5         for (int i = 0; i < finalScore.length; i++) {
6             if (finalScore[i] > 70) {
7                 System.out.println("Student " + i + " with score " +
8                     finalScore[i] + " - Passed!");
9             } else {
10                 System.out.println("Student " + i + " with score " +
11                     finalScore[i] + " - Did not pass!");
12             }
13         }
14         sc.close();
15     }
16 }

```

- 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.

- Open a text editor, create a Java file then save it with the name **arrayAverageScoreXX**.
(XX=student ID number)
- Write the basic structure of the Java programming language which contains the **main()** function
- Add the Scanner library and make a **Scanner** declaration for input purposes
- 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



```

public class arrayAveragescore07 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[] score = new int[10];
        double total = 0;
        double average;
        for (int i = 0; i < score.length; i++) {
            System.out.print("Enter student score " + (i + 1) + ": ");
            score[i] = sc.nextInt();
        }
        for (int i = 0; i < score.length; i++) {
            total += score[i];
        }
        average = total / score.length;
        System.out.println("The class average score is: " + average);
        sc.close();
    }
}

```

The output window shows the following interaction:

```

Enter student score 7: 89
Enter student score 8: 89
Enter student score 9: 98
Enter student score 10: 88
The class average score is: 83.4

```

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)

```

public class arrayAveragescore07 {
    public static void main(String[] args) {
        int[] score = new int[10];
        double total = 0;
        double average;
        int passedCount = 0;

        // Step 5: Using a loop, create an input to fill in the scores
        for (int i = 0; i < score.length; i++) {
            System.out.print("Enter student score " + (i + 1) + ": ");
            score[i] = sc.nextInt();
        }

        // Step 6: Using a loop, calculate the total number of scores
        for (int i = 0; i < score.length; i++) {
            total += score[i];
        }

        // Count students who passed (score > 70)
        for (int i = 0; i < score.length; i++) {
            if (score[i] > 70) {
                passedCount++;
            }
        }
    }
}

```

The output window shows the following interaction:

```

Enter student score 8: 88
Enter student score 9: 98
Enter student score 10: 84
The class average score is: 84.1
Number of students who passed (score > 70): 9

```



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
```

The screenshot shows a GitHub Copilot Pro interface with a Java code editor. The code calculates the average score of students based on user input. It defines a method `arrayAverageScore07` that takes an array of integers and prints the average for passed and failed students. The GitHub interface shows a commit message and a graph of previous commits.

```
public class arrayAverageScore07 {
    public static void main(String[] args) {
        totalPassed += score[i];
        failedCount++;
    }

    // Calculate and display average for passed students
    if (passedCount > 0) {
        double averagePassed = totalPassed / passedCount;
        System.out.println("The average score of students who passed is " + averagePassed);
    }

    // Calculate and display average for failed students
    if (failedCount > 0) {
        double averageFailed = totalFailed / failedCount;
        System.out.println("The average score of students who failed is " + averageFailed);
    }
}
```

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

linearSearchXX. (XX=student ID number)

2. Add the following code



```

6   public static void main(String[] args) {
7       int[] arrayInt = { 34, 18, 26, 48, 72, 20, 56, 63 };
8       int key = 20;
9       int result = 0;
10      for (int i = 0; i < arrayInt.length; i++) {
11          if (arrayInt[i] == key) {
12              result = i;
13              break;
14          }
15      }
16      System.out.println("The key in the array is located at index position " + result);
17  }
18

```

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

The screenshot shows the GitHub Copilot interface within a code editor. The code editor displays the Java file `linearSearch07.java` from a workspace named `PRAKTIKUMDASPRO-2`. The code implements a linear search algorithm to find the index of a key value in an array. The output terminal shows the command `PS C:\Users\ASUS\ROG\Documents\PRAKTIKUMDASPRO\PRAKTIKUMDASPRO-2> & 'C:\Program Files\Java\jdk-24\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\ASUS\ROG\AppData\Roaming\Code\User\workspaceStorage\24fb08fc21927ae156d2f9ab1e0554\redhat\java\jdt-ws\PRAKTIKUMDASPRO-2_68580b5e\bin' 'linearSearch07'` followed by the result: `The key in the array is located at index position 5`.

4. Commit and push the changes to GitHub

Questions!

- 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
```

The screenshot shows the GitHub Copilot interface within a code editor. The code is a Java program named `linearSearch07.java` which implements a linear search algorithm. The user has entered the key value 10 and the program output shows that the key was found at index position 3.

```
public class linearSearch07 {
    public static void main(String[] args) {
        // Tampilkan hasil
        if (result != -1) {
            System.out.println("The key in the array is located at index position " + result);
        } else {
            System.out.println("Key not found in the array");
        }
        sc.close();
    }
}
```

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
```

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface with the following details:

- File Explorer:** Shows a project structure with files like `arrayValue07.java`, `arrayAverageScore07.java`, `linearSearch07.java`, and `SistemHotel lengkap`. A graph view shows multiple branches of the file `linearSearch07.java`.
- Code Editor:** Displays the `linearSearch07.java` file. The code implements a linear search algorithm. A yellow dot indicates the current cursor position.
- Terminal:** Shows the output of running the program. The user enters array elements (3, 4, 5) and a key to search for (11). The program outputs "Key not found in the array".
- Status Bar:** Shows the file name as `linearSearch07.java`, the status as `Java Ready`, and other system information.

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.



```

import java.util.ArrayList;
import java.util.Collections;
import java.util.Scanner;

public class StudentGradeManager07 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("-----");
        System.out.println("STUDENT GRADE MANAGEMENT SYSTEM");
        System.out.println("-----");

        // Enter the number of students
        int numStudents = 0;
        while (true) {
            try {
                System.out.print("Enter the number of students: ");
                numStudents = Integer.parseInt(sc.nextLine());
                if (numStudents > 0) {
                    break;
                }
            } catch (Exception e) {
                System.out.println("Please enter a valid integer greater than 0.");
            }
        }

        int totalGradeSum = 0;
        int highestGrade = 0;
        int lowestGrade = 100;
        for (int i = 1; i <= numStudents; i++) {
            System.out.print("Enter grade " + i + ": ");
            int grade = Integer.parseInt(sc.nextLine());
            totalGradeSum += grade;
            if (grade > highestGrade) {
                highestGrade = grade;
            }
            if (grade < lowestGrade) {
                lowestGrade = grade;
            }
        }

        System.out.println("Total number of students: " + numStudents);
        System.out.println("Average grade: " + (totalGradeSum / numStudents));
        System.out.println("Highest grade: " + highestGrade);
        System.out.println("Lowest grade: " + lowestGrade);

        System.out.println("All grades entered:");
        for (int i = 1; i <= numStudents; i++) {
            System.out.println(i + ": " + grade);
        }
    }
}

```

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.



```

CafeOrderManagement07.java
1 package SistemHotelLengkap;
2
3 public class CafeOrderManagement07 {
4     static void main(String[] args) {
5         // Input order name
6         System.out.print("Food/Drink name: ");
7         orderNames[i] = scanner.nextLine();
8
9         // Input order price with validation
10        while (true) {
11            try {
12                System.out.print("Price: $");
13                orderPrices[i] = Double.parseDouble(scanner.nextLine());
14            } catch (NumberFormatException e) {
15                System.out.println("Please enter a valid price.");
16            }
17        }
18    }
19 }

```

PROBLEMS TERMINAL

```

PS C:\Users\ASUS\ROG\Documents\PRAKTIKUMDASPRO\PRAKTIKUMDASPRO-2> & 'C:\Program Files\Java\jdk-24\bin\java.exe' '-Xss1M' -Xms1M -Xmx1G -XX:+ShowCodeDetailsInExceptionMessages' -cp "C:\Users\ASUS\ROG\Applata\Mooring\Code\User\workspaceStorage\24fa89fc21927ae158d2f0a12e6554\redhat\java\jdt_ws\PRAKTIKUMDASPRO-2_68585b50\bin" Assignment.CafeOrderManagement07
List of Orders:
No. Item Price
1 seblak $1,00
2 basreng $10,00
3 udang keju $28,00
TOTAL COST: $53,00

```

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.



The screenshot shows a Java development environment with the following details:

- File Explorer:** Shows several Java files including `arrayNumbers07.java`, `arrayValue07.java`, `arrayAverageScore07.java`, `linearSearch07.java`, `StudentGradeManager07.java`, `CafeOrderManagement07.java`, and `CafeMenuSearch07.java`.
- Code Editor:** Displays the `CafeMenuSearch07.java` file with the following code:

```
import java.util.Scanner;

public class CafeMenuSearch07 {
    public static int linearSearch(String[] menu, String search) {
        for (int i = 0; i < menu.length; i++) {
            // Case-insensitive comparison
            if (menu[i].equalsIgnoreCase(search)) {
                return i; // Return index if found
            }
        }
        return -1; // Return -1 if not found
    }
}
```
- Terminal:** Shows the command line output of the Java application:

```
PS C:\Users\ASUS\ROG\Documents\PRAKTIKUMDASPRO\PRAKTIKUMDASPRO-2> & 'C:\Program Files\Java\jdk-24\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' -cp "C:\Users\ASUS\ROG\AppData\Roaming\Code\User\workspaceStorage\24f489fc21927ae158d2f0a12e6554\redhat\java\jdt-vs\PRAKTIKUMDASPRO-2_68585b50\bin" 'Assignment.CafeMenuSearch07'
```
- Output:** Displays the search result for "Teh Tarik":

```
Enter the food/drink item to search for (or 'exit' to quit): teh tarik
-----
? ITEM FOUND!
"Teh Tarik" is available on our menu.
Position in menu: 5
```
- Right Panel:** Includes a GitHub Copilot message, workspace status, and a GitHub Copilot Pro upgrade offer.