**JOBSHEET 9 ARRAY 1**

# 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

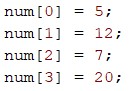
# Laboratory

## Experiment 1: Fill in Array Element Experiment Time: 20 minutes

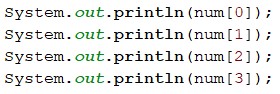
1. Open a text editor, create a new Java class with the name **arrayNumbersXX**. (XX=student ID number)
2. Write the basic structure of the Java programming language which contains the **main()** function
3. Create an array of integer type named **num** with a capacity of 4 elements



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



1. Display all contents of the elements to the screen



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



*/\*\**

*\* arrayNumbers22*

*\*/*

public class arrayNumbers22 {

    public static void main(String[] args) {

        int [] num22 = new int[5];

            num22[0] = 10;

            num22[1] = 22;

            num22[2] = 7;

            num22[3] = 20;

            num22[4] = 10;

*// num22[0] = 10;*

*// num22[0] = 10;*

            System.out.println(num22[0]);

            System.out.println(num22[1]);

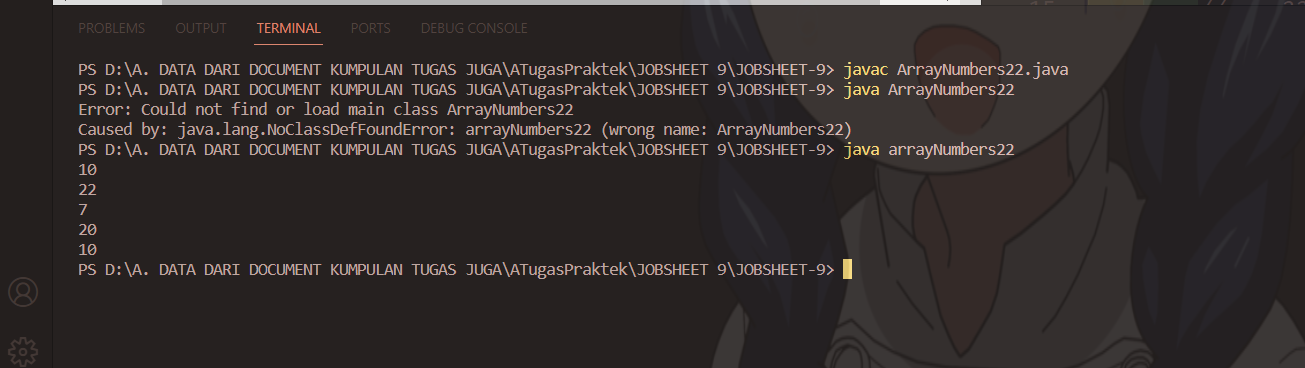
            System.out.println(num22[2]);

            System.out.println(num22[3]);

            System.out.println(num22[4]);

    }

}



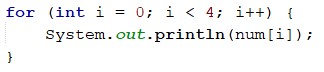
1. Commit and push the changes to GitHub.

**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?

1. Modify the program code by initializing the array elements at the same time when declaring the array.
2. Change the statement in step 6 to be like this



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

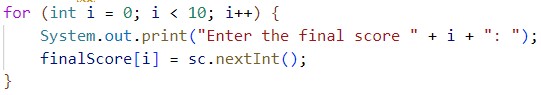
1. If the condition in the for-loop statement is changed to i <= 4, what is the output of the program? Why is the result like that?
2. Commit and push the changes to GitHub.

## Experiment 2: Requesting User Input to Fill in an Array Element Experiment Time: 40 minutes

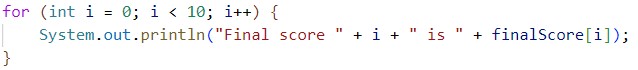
1. Open a text editor, create a Java file then save it with the name **arrayValueXX**. (XX=student ID number)
2. Write the basic structure of the Java programming language which contains the **main()** function
3. Add the Scanner library
4. Create an array of integer type with the name **finalScore**, with a capacity of 10 elements



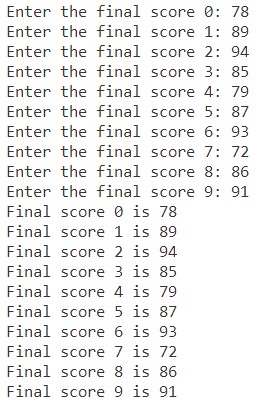
1. Using a loop, create an input to fill in the **finalScore** array element



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



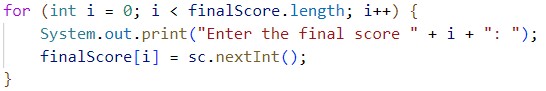
1. Compile and run the program. Match the results of the running programs that you have created according to the following display



1. Commit and push the changes to GitHub.

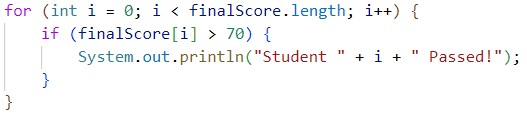
**Questions!**

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



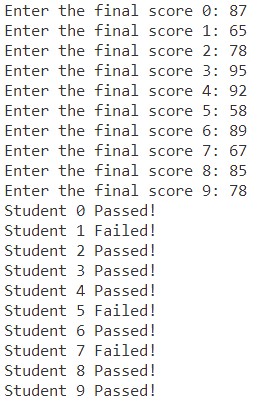
Run the program. Have there been any changes? How can it be like that?

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



Run the program and describe the flow of the program!

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



1. Commit and push the changes to GitHub.

**2.3 Experiment 3: Perform Arithmetic Operations on Array Elements**

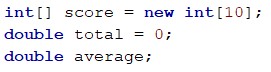
# Experiment Time: 75 minutes

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.

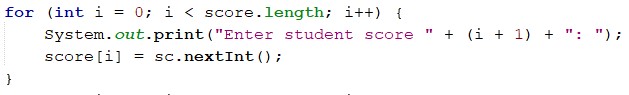
1. Open a text editor, create a Java file then save it with the name **arrayAverageScoreXX**.

(XX=student ID number)

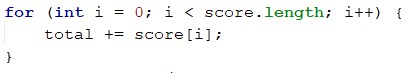
1. Write the basic structure of the Java programming language which contains the **main()** function
2. Add the Scanner library and make a **Scanner** declaration for input purposes
3. Create an array of integer types with the name **score** with a capacity of 10. Then declare the variables total and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



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



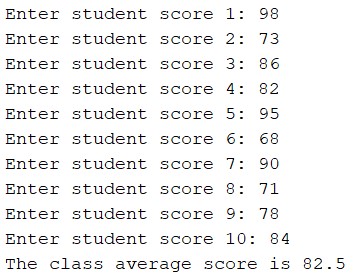
1. Using a loop, calculate the total number of scores.



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



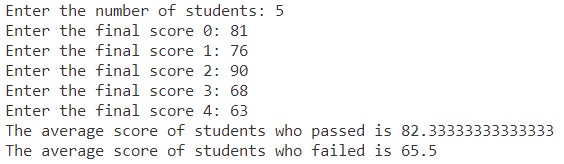
1. Compile and run the program. Match the results of the running programs that you have created according to the following display



1. 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

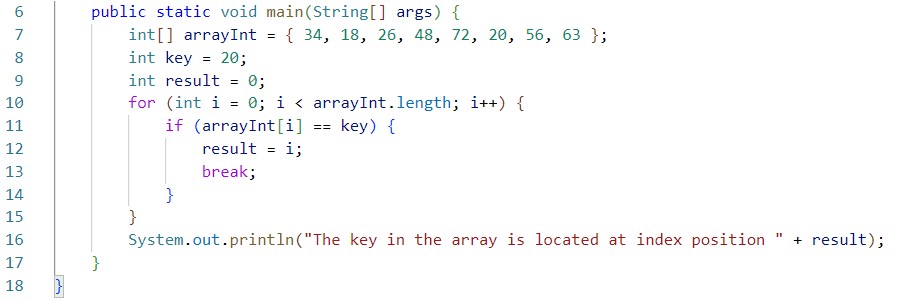


* + 1. Commit and push the changes to GitHub

**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



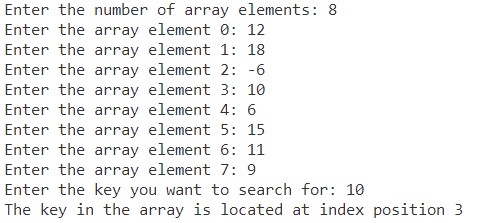
1. Compile and run the program. Match the results of the running programs that you have created according to the following display



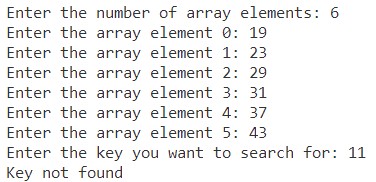
1. Commit and push the changes to GitHub

**Questions!**

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



1. 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:



# Assignment

1. Create a program to produce the highest value, lowest value, and average from an array containing integer type numbers.

Terms:

* + Input: Number of elements, value of each element
  + Output: Highest value, lowest value, average value

1. Implement the flowchart that was created in the assignment for Week 9 of the Programming Fundamentals course related to the group project into Java program code.

Commit and push the results of your program code to your project's GitHub repository.

Note: assignments may only apply material from Week 1 to Week 9.