# JOBSHEET 9
# ARRAY 1

## 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

## 2. Laboratory

## 2.1 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

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

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

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

5. Display all contents of the elements to the screen

```java
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?

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

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

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

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?

5. Commit and push the changes to GitHub.

## 2.2 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

```java
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**?

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

```java
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 **arrayAverageScoreXX**. (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

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

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

```java
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.

```java
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**

```java
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 **linearSearchXX**. (XX=student ID number)

2. Add the following code

```java
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
```

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.

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
```

## 3.  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

2.  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.