Array Manipulation and For Loops

Estimated time needed: 20 minutes

After completing this lab, you will be able to use arrays and for loops in Java.

You are currently viewing this lab in a Cloud-based Integrated Development Environment (Cloud IDE). It is a fully-online integrated development environment that is pre-installed with JDK 21, allowing you to code, develop, and learn in one location.

Learning Objectives

After completing this lab, you will be able to:

- Create an integer array with values
- Access an element at a specified index in the array
- Display all elements in the array
- Determine the length of an array
- Initialize an array and assign values
- Use loops to iterate through the array
- Add functionality that locates the maximum and minimum values in the array using loops

Array creation and elements access

An array is a set of elements of a particular data type. In this section of the lab, you will create an array with the years from 2020 to 2025 stored in it.

1. Create a project directory by running the following command.

   ```
   mkdir my_array_proj
   ```

2. Run the following code to create the directory structure.

   ```
   mkdir -p my_array_proj/src
   mkdir -p my_array_proj/classes
   mkdir -p my_array_proj/test
   cd my_array_proj
   ```

3. Now create a file named `ArrayAccess.java` inside the src directory.

   ```
   touch /home/project/my_array_proj/src/ArrayAccess.java
   ```

4. Click the **Open ArrayAccess.java in IDE** button to open the file for editing.

Open **ArrayAccess.java** in IDE

5. Read each statement and the accompanying explanations in the following program to understand how to create the array and how to access each element in the array. Paste the following content in `ArrayAccess.java`.

   ```
   public class ArrayAccess {
       public static void main(String s[]) {
           int years[] = {2020,2021,2022,2023,2024,2025};
           System.out.println(years[0]);
           System.out.println(years[1]);
           System.out.println(years[2]);
           System.out.println(years[3]);
           System.out.println(years[4]);
           System.out.println(years[5]);
       }
   }
   ```

`int years[] = {2020,2021,2022,2023,2024,2025};` - An array named years of type int is declared and it is initialized and assigned with some numbers.
`System.out.println(years[n]);` - Prints the element in nth position, where `n` starts from `0`. The first element of the array is accessed with index 0. If the length of the array is `m`, the index position last element of the array is `m-1`. In the case of the example used here, the length of the array is 6 and the index of the last element is 5.

6. In the IDE, compile the Java program. specifying the destination directory as the `classes` directory that you created. See the following code example.

   ```
   javac -d classes src/ArrayAccess.java
   ```

7. Set the `CLASSPATH` variable.

   ```
   export CLASSPATH=$CLASSPATH:/home/project/my_array_proj/classes
   ```

8. Now, when you run the Java program, the Java program will run seamlessly as expected.

   ```
   java ArrayAccess
   ```

You will see the following output:

```
2020
2021
2022
2023
2024
2025
```

Initialize and create the array

You must first initialize an array before assigning values to its various index positions. Let's learn how to do these tasks.

1. Click the **Open ArrayAccess.java in IDE** button to open the file for editing if the IDE is not already open.

Open **ArrayAccess.java** in IDE

2. Read each statement in the following program to understand how the program initializes the array and how each the program assigns each element later. Paste the following content in `ArrayAccess.java`.

```
public class ArrayAccess {
    public static void main(String s[]) {
        int years[] = new int[6];
        years[1] = 2021;
        years[3] = 2023;
        System.out.println(years[0]);
        System.out.println(years[1]);
        System.out.println(years[2]);
        System.out.println(years[3]);
        System.out.println(years[4]);
        System.out.println(years[5]);
    }
}
```

Heres an explanation of the code:

`int years[] = new int[6];` - An array named `years` of type int is declared The array is initialized to hold six (6) elements. When an array initializes, depending on the type of array, the array has default values. An `int` array by default, has valuee the value of zero (0).

`years[n]` - Allocates an element at the specified index `n`. In this example at `1` and `3`.

`System.out.println(years[n]);` - Prints the element in nth position, where `n` starts from `0`. In this example, as you are only assigning values to index, you will see what values are allocated to the rest of the elements when you compile and run the code.

3. Compile the Java program, specifying the destination directory as the `classes` directory that you created.

```
javac -d classes src/ArrayAccess.java
```

4. Now run the Java program.

```
java ArrayAccess
```

You will see the output as below:

```
0
2021
0
2023
0
0
```

You can see that the index positions that were not assigned values have the values of zero (0) by default as the array was initialized and then assigned.

Using 'for' loops to traverse through an array

When you work with an array, you might not always know the length of the array especially if you did not create it. In programming, the arrays are either created or passed on from other sources. In this part of the lab, you will be using a `length` array variable and `for` loop to traverse through the array.

1. Click on the button below to open the file for editing if it is not already open.

Open **ArrayAccess.java** in IDE

2. Read each statement in the following program and understand how the length of the array is determined and the loop is used to traverse through. Paste the following content in `ArrayAccess.java`.

```
public class ArrayAccess {
    public static void main(String s[]) {
        int years[] = new int[6];
        years[0] = 2020;
        years[1] = 2021;
        years[2] = 2022;
        years[3] = 2023;
        years[4] = 2024;
        years[5] = 2025;
        int count_years = years.length;
        System.out.println("the length of the array is " + count_years);
        for (int i=0; i<count_years; i++) {
            System.out.println(years[i]);
        }
    }
}
```

`int count_years = years.length;` - Creates a variable named `count_years` and stores the value of the length of the array in the variable.

`for (int i=0; i<count_years; i++)` - Creates a `for` loop. The `for` has 3 parts. The initial value, the condition, and the incremental value. In this exercise, you are going to use variable `i` to traverse through the index positions of the array.

The code begin with the initial value of `i` equal to zero(0). The code checks the condition to see if value of `i` is less than the length of the array and then increments the value of `i` after one loop is executed. The code checks the condition every time the loop increments. The code block enclosed within the brackets after creation of the `for` loop continues to run until the condition is met.

3. Compile the Java program, specifying the destination directory using the `classes` directory that you previously created.

   ```
   javac -d classes src/ArrayAccess.java
   ```

4. Now run the Java program.

   ```
   java ArrayAccess
   ```

You will see the following output:

```
2020
2021
2022
2023
2024
2025
```

As you can see, the loop iterated through the array for as long as the array had an element remaining to print and then printed that element.

Using 'for' loops to traverse through the command line arguments

First, command line arguments are code that you provide after `java <classname>` when you run a Java program. Command line arguements are useful ways you can provide user input to the program.

1. Click the **Open ArrayAcess.java in IDE** button to open the file for editing if the IDE environment is not already open.

Open **ArrayAccess.java** in IDE

2. Read each statement in the following program to understand how the method considers the main method's input parameters. Paste the following content in `ArrayAccess.java`.

```
public class ArrayAccess {
    public static void main(String s[]) {
        int num_args = s.length;
        System.out.println("the length of the array is " + num_args);
        for (int i=0; i<num_args; i++) {
            System.out.println(s[i]);
        }
    }
}
```

`int num_args = s.length;` - Creates a variable named `num_args` and stores the value of the length of the array of command line arguments in it.

3. Compile the Java program, specifying the destination directory using the `classes` directory that you previously created.

```
javac -d classes src/ArrayAccess.java
```

4. Now run the Java program.

```
java ArrayAccess
```

You will not see any output as the code did not pass command line arguments.

4. Run the Java program again. However, with this run include the following command line arguments when running the program.

```
java ArrayAccess first second third fourth fifth
```

Here `first`,`second`,`third`,`fourth` and `fifth` are command line arguments.

You will see the following output:

```
the length of the array is 5
first
second
third
fourth
fifth
```

You can see that the loop iterated through the command line arguments array for as long as there was an element remaining to process. The code also printed the element details.

Practice Exercise

Exercise 1

1. Create and initialize an int array with 10 elements.

2. Print each element of the array without using the 'for' loop.

3. Print each element of the array using the 'for' loop.

Exercise 2

1. Create and initialize a string.

2. Convert the string to a char array using the `toCharArray()` method.

3. Print each element of the array using the 'for' loop.

▶ Click here for complete exercise solution.

Conclusion

In this lab, you learned that:

- You can use the `int`code to create, or initialize arrays.
- You can create a variable and store the value of the length of the array in the variable.
- The `for` loop has three parts. The initial value, the condition, and the incremental value. You can use a variable to traverse through the index positions of the array.

Author(s)

Lavanya