Week 08 - Arrays

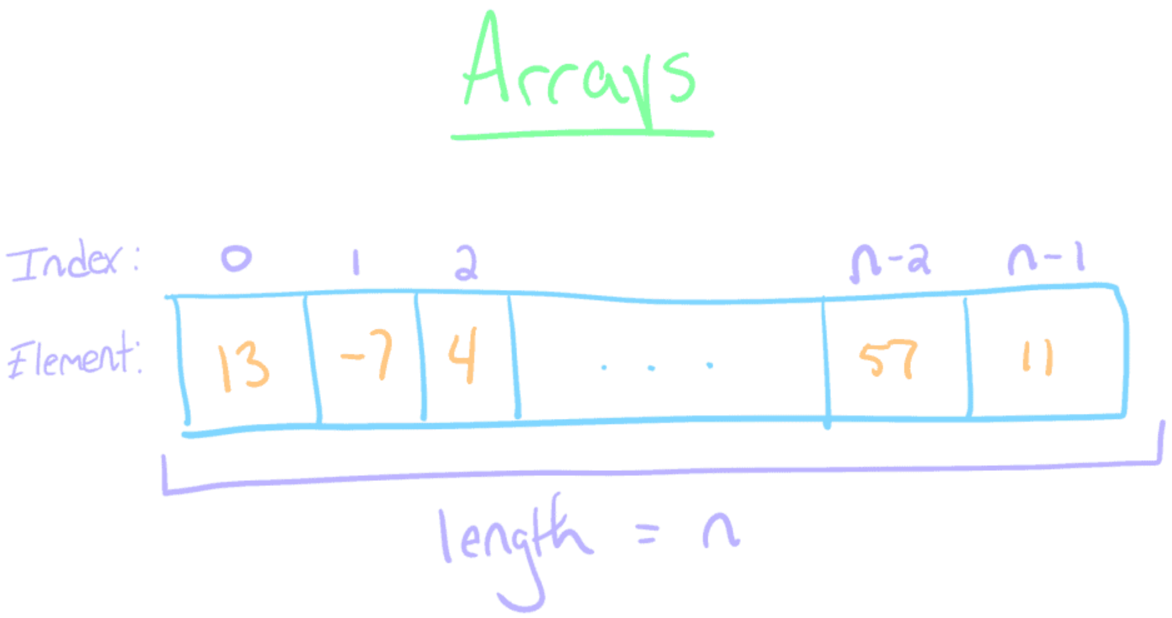
Louis Botha, `louis.botha@tuni.fi`

Java array is an object which contains elements of a similar data type. Additionally, an array is a linear data structure representing that can be accessed by index.

The size of an array must be provided before storing data. Listed below are some properties of an array:

- Each element in an array is of the **same** data type and has the **same** size
- Elements of the array are stored at contiguous memory locations with the first element is starting at the smallest memory location
- Elements of the array can be randomly accessed
- Elements in an array can be of any data type, including primitive types (such as `int`, `double`, and `boolean`) or object references (such as `String`, `Date`, and other custom objects).

Array in Java is index-based, the first element of the array is stored at the 0th index, 2nd element is stored on 1st index and so on.



Array Diagram

- Index refers to an element's position within an array. The index of an array starts from 0.
- Element is an item in an array and each element is accessed by its numerical *index*
- length is a final variable applicable for arrays from which you can obtain the size of the array.

Oracle array documentation

Types of Arrays in Java

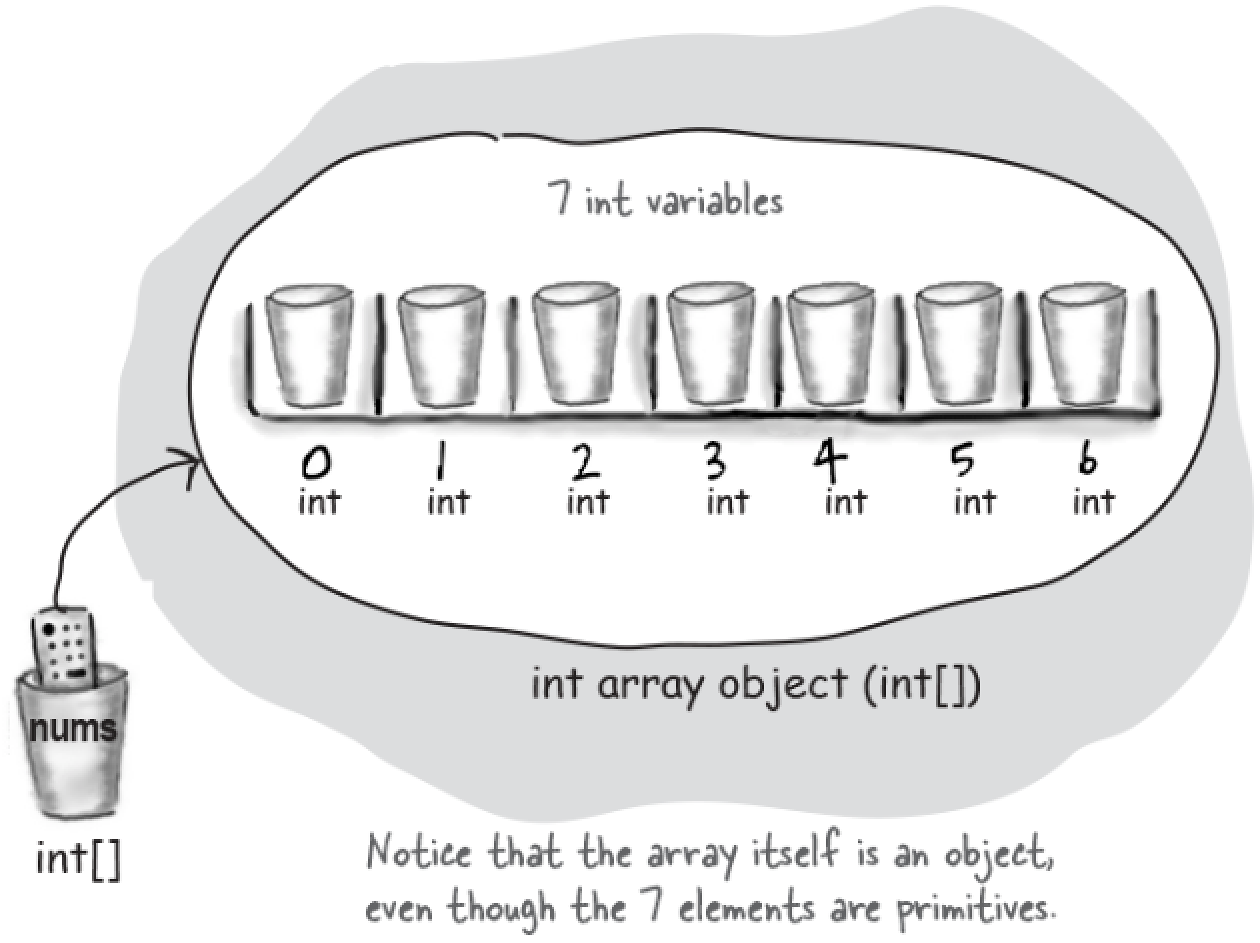
There are two types of arrays.

- Single Dimensional Array
- Two Dimensional Array

One Dimensional Array

Arrays in Java are declared with a specific size and can be accessed by index. The first element in an array has an index of 0, and the last element has an index of size-1. The size of an array cannot be changed after it has been created.

You can think of an array as a shelf full of the same sized containers waiting to be filled with some content. The containers are all the same size and can only hold the same type of content.


7 int variables
0
int
1
int
2
int
3
int
4
int
5
int
6
int
int array object (int[])
nums
int[]
Notice that the array itself is an object, even though the 7 elements are primitives.

Head First Java

Here's an example of how to declare and initialize an array of integers in Java:

```
int[] numbers = new int[5];   // Declare an array of size 5
numbers[0] = 1;               // Set the first element to 1
numbers[1] = 2;               // Set the second element to 2
numbers[2] = 3;               // Set the third element to 3
numbers[3] = 4;               // Set the fourth element to 4
```

```
numbers[4] = 5;               // Set the fifth element to 5
```

You can also declare and initialize an array in one statement, like this:

```
int[] numbers = {1, 2, 3, 4, 5};   // Declare and initialize a
n array of size 5
```

You can access the elements of an array using square brackets and the index of the element you want to access. Here's an example:

```
// Access the third element (index 2) of the array
int thirdNumber = numbers[2];
// Prints 3 to the console
System.out.println(thirdNumber);
```

Arrays in Java also provide a `length` property that returns the size of the array. Here's an example of how to get the length of an array:

```
int length = numbers.length;   // Get the length of the array
System.out.println(length);   // Prints 5 to the console
```

You can also use loops to iterate over the elements of an array, like this:

```
for(int i = 0; i < numbers.length; i++) {
    // Prints each element of the array to the console
    System.out.println(numbers[i]);
}
```

Arrays in Java provide a fast and efficient way to store and access a fixed-size collection of elements, but they have some limitations, such as their fixed size and the need to manually manage their storage and manipulation. For more dynamic and flexible collections of elements, you can use the Java collections framework, which provides a rich set of interfaces and classes for working with dynamic collections of elements.

Two Dimensional Arrays

Two Dimensional Arrays, also called 2D array, is a data structure that stores a matrix of values in a two-dimensional grid of rows and columns. Each element in a 2D array is identified by a pair of indices, representing the row and column number.

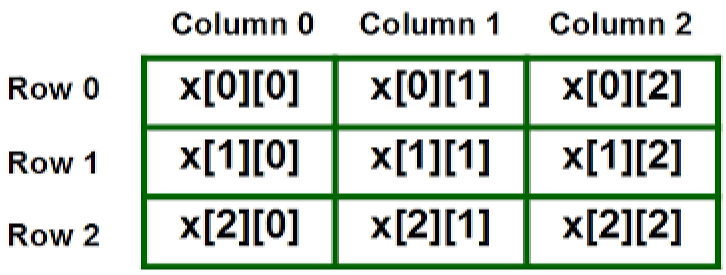
To declare and initialize a 2D array in Java, you can use the following syntax:

```
type[][] arrayName = new type[numRows][numColumns];
```

Here, `type` is the data type of the elements in the array (such as `int`, `double`, or `String`), `arrayName` is the name of the array, `numRows` is the number of rows in the array, and `numColumns` is the number of columns in the array.

For example, to create a 2D array of integers with 3 rows and 4 columns, you can use the following code:

```
int[][] array2D = new int[3][4];
```



	Column 0	Column 1	Column 2
Row 0	x[0][0]	x[0][1]	x[0][2]
Row 1	x[1][0]	x[1][1]	x[1][2]
Row 2	x[2][0]	x[2][1]	x[2][2]

You can access the elements of a 2D array using the row and column indices, like this:

```
int element = array2D[rowIndex][columnIndex];
```

Here, `rowIndex` and `columnIndex` are the zero-based indices of the row and column, respectively.

To initialize the values of a 2D array, you can use nested loops to iterate over the rows and columns and set the values of each element, like this:

```
for (int i = 0; i < numRows; i++) {
    for (int j = 0; j < numColumns; j++) {
        array2D[i][j] = i * numColumns + j;
    }
}
```

This code initializes the 2D array with the values `0` to `numRows * numColumns - 1`, in row-major order.

You can also iterate over the elements of a 2D array using nested loops, like this:

```
for (int i = 0; i < numRows; i++) {
    for (int j = 0; j < numColumns; j++) {
        System.out.print(array2D[i][j] + " ");
    }
    System.out.println();
}
```

This code prints the values of the 2D array in row-major order, with each row on a new line.
2D arrays in Java are useful for storing and manipulating matrices of values, such as images, game boards, and other 2D grids of data. They can be used in conjunction with loops, conditions, and other control structures to perform complex computations and operations on the data.

ArrayIndexOutOfBoundsException

This is the most common and most likely mistake that are made when working with arrays.

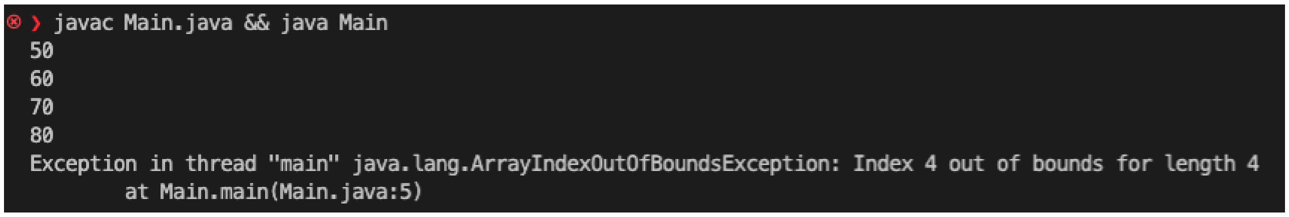
> *:attention: In a developer interview, if you need to review code, this error is most likely going to be there somewhere.*

The Java Virtual Machine (JVM) throws an ArrayIndexOutOfBoundsException if the length of the array is negative, equal to the array size or greater than the array size while traversing the array. In short, we are trying to access data from the array that doesn't exist.

```
//Java Program to demonstrate the case of
```

```java
//ArrayIndexOutOfBoundsException in a Java Array.
public class Main {
  public static void main(String args[]) {
    int array[] = {50, 60, 70, 80};
    for (int i=0; i<=array.length; i++) {
      System.out.println(array[i]);
    }
  }
}
```

Copy, paste and run the piece of code. The above code should throw this exception(error).



```
❯ javac Main.java && java Main
 50
 60
 70
 80
 Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 4 out of bounds for length 4
         at Main.main(Main.java:5)
```

Why is the index going out of bounds? Do you spot the 1 character change you need to make for the code to run?

The ArrayIndexOutOfBoundsException is a runtime error/bug. The compiler sees valid code but only when running the application is the exception thrown and do we see there is a problem with our application. Don't hide ArrayIndexOutOfBoundsExceptions with a `try/catch` statement. It is a programming error that needs to be prevented with good coding practice.