What Are Arrays?

In this lesson, an introduction to Arrays data structure is provided.

We'll cover the following

Definition

An *array* is a collection of *similar* data types in a sequenced format. The position of each data member in this sequence is represented with a number known as its *index*.

Elements of an Array

The members of this data collection are referred to as *array elements*.

Length of an Array

The *total number* of elements in an array is called the length of an array. This length is set once at the time of creation of an array and *can't be changed later* in a program. We can check the length of an array using an inbuilt functionality of Java as `arrayName.length`.

Indexing

The indexes in an array always range from **zero** to **length-1**. For example, an array containing the first 100 natural numbers will have a *length of **100*** and *indexes from **0 to 99***.

Declaration

In Java, at the time of declaration an array we must specify the following:

- Datatype
- Name
- Size

Syntactically we can declare an array of an integer data type in the following ways:

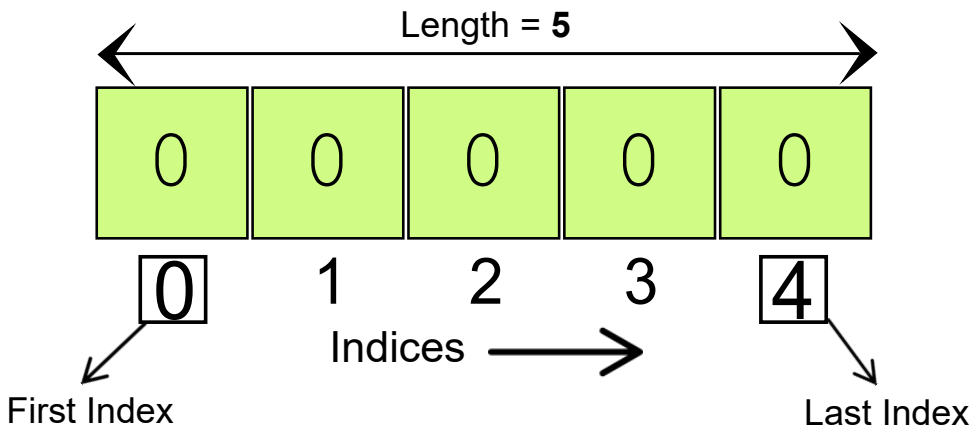
```
int[] myArray;
int []myArray;
int myArray[];
```

In Java an array needs to be instantiated i.e. we need to specify the size of the array which is to be allocated to it in the memory. Instantiation is done using the keyword `new` as follows:

```
int[] myArray = new int[5] // Instantiation of an Array of size 5;
```

Array instantiation using keyword new

Lets see what the above line of code actually does:



Array Illustration

It creates an array to store 5 int values and sets them all to 0.

> **Note**: The starting index of an array is always `0` not `1`

Default values

If the user has not assigned any values to the array elements Java puts some *default* values in there as follows:

Array data Type	Default Value
All numeric data types (`int`, `short`, `double` etc.)	`0`
`boolean`	`false`
`char`	*null character*

Initializing & accessing array elements

Initialization means assigning specific values to array elements and defining the array in the same line of code.

One can *declare*, *instantiate* and *initialize* an array in a single line using curly braces containing the values we want to store in that array. The number of values in the curly braces decide the length of that array. The syntax is as follows:

```
int[] myArray1 = {1,2,3,4,5} //This declares,instantiates and initializes
                             // an array with 5 elements 1,2,3,4 and 5
int[] myArray2 = {2,4,6}     //This declares,instantiates and initializes
                             // an array with 3 elements 2,4 and 6
```

The above approach is normally used when the values to be stored are already known.

The array elements can be declared and assigned individually using the *index values*. To learn how to access the elements individually one can refer to the below example:

```
class Arrays {
    public static void main(String args[]) {
        int[] myArray = new int[5]; //Declaration and Instantiation of Array with length 5
```

```
        System.out.println("Printing the size of myArray: " + myArray.length);

        myArray[0] = 2; //accessing 1st element and assigning value 2 to it
        myArray[1] = 4; //accessing 1st element and assigning value 4 to it
        myArray[2] = 6; //accessing 1st element and assigning value 6 to it
        myArray[3] = 8; //accessing 1st element and assigning value 8 to it
        myArray[4] = 10; //accessing 1st element and assigning value 10 to it

        /*Printing the stored values*/

        System.out.println("First element of myArray is:  " + myArray[0]);
        System.out.println("Second element of myArray is: " + myArray[1]);
        System.out.println("Third element of myArray is:  " + myArray[2]);
        System.out.println("Fourth element of myArray is: " + myArray[3]);
        System.out.println("Fifth element of myArray is:  " + myArray[4]);
    }
}
```

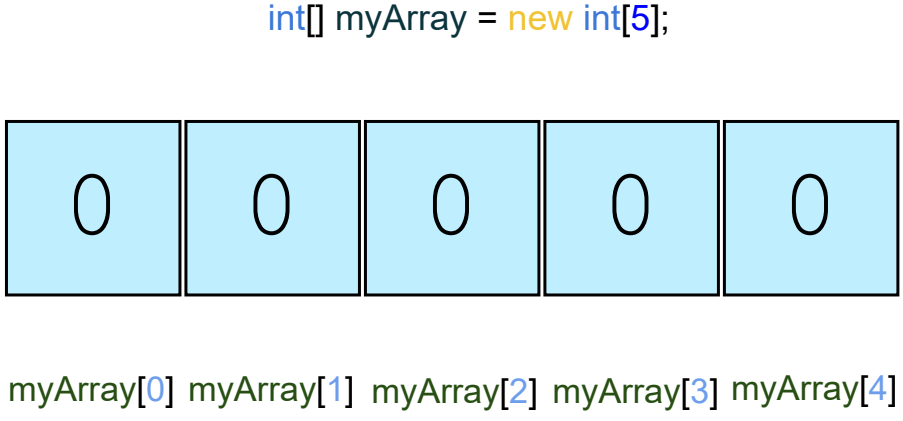
Initializing an Array

The square bracket can be used to:

- Indicate that a variable is an array when declaring it.
- Read from or writing to a specific array element.

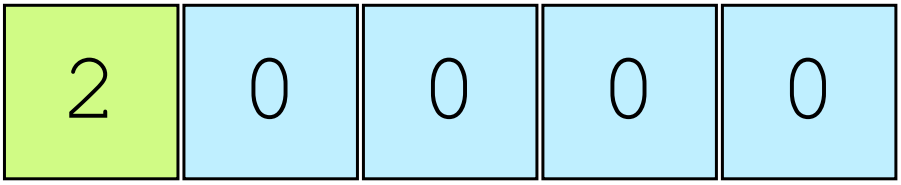
> **Note:** A positive integer index inside square brackets is used to access a certain element as `arrayName[index]`.

A graphical representation of the above is:



Indices of an Array

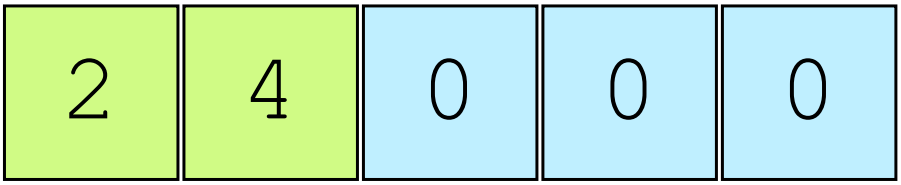
myArray[0] = 2;



myArray[0]

Index = 0

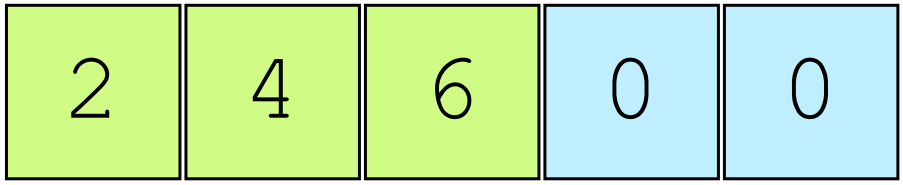
myArray[1] = 4;



myArray[1]

Index = 1

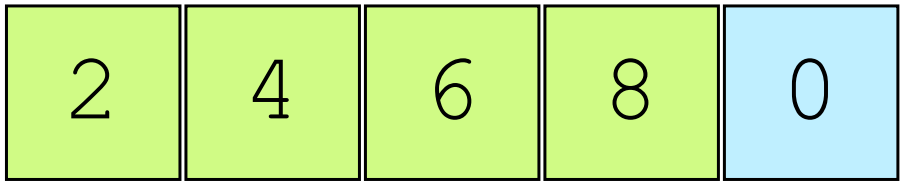
myArray[2] = 6;



myArray[2]

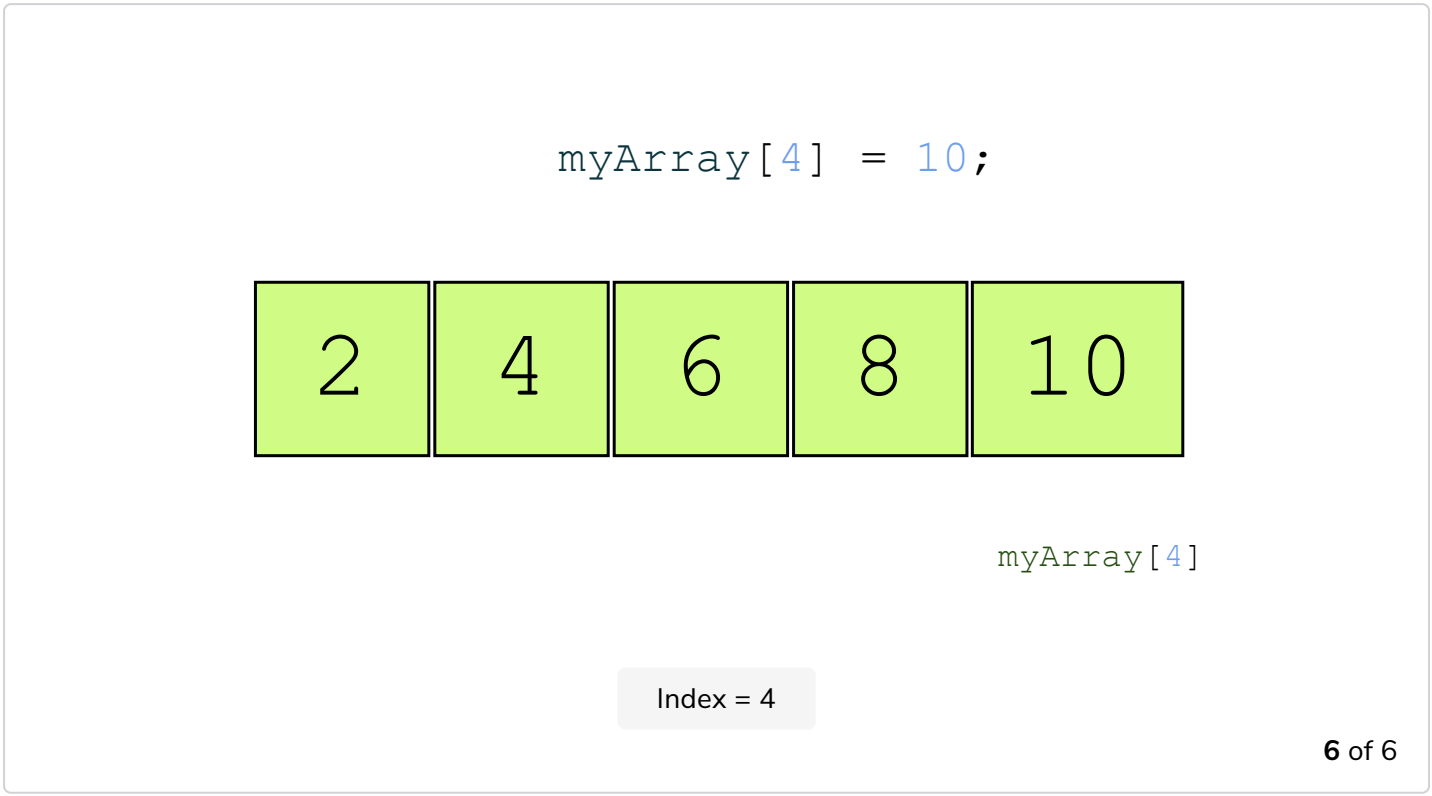
Index = 2

myArray[3] = 8;



myArray[3]

Index = 3



> **Quick question:** What do you think about setting a negative number as the length of an array?

Hide Hint

An array cannot have a negative length!

Using the above methods, one can notice that assigning values to an array with a large number of elements will require a lot of effort and code duplication. It will be significantly inefficient to do so. Hence, to populate an array with values in an efficient way, we can use the following method.

Using loops

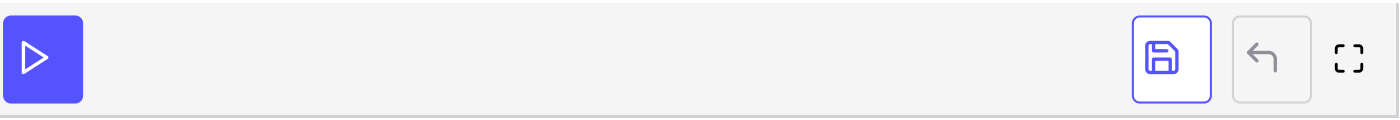
Let's learn how can we use *loops* to initialize arrays:

```
class Arrays {
    public static void main(String args[]) {
```

```
        int[] myArray = new int[10]; // Declaration and Instantiation of Array with length 10


        for (int i = 0; i < myArray.length; i++) // Iterate through indexes 0-9
        {
            myArray[i] = i + 1; // Initialize values to 1-10
        }

        for (int i = 0; i < myArray.length; i++) {
            System.out.println("The value at myArray[" + i + "] is: " + myArray[i]);
            // Printing all values to console
        }
    }
}
```



Initialization using loops

In the next lesson, we will discuss arrays in more detail.