The Array Collection

In the following lesson, you will learn how to use arrays in Scala.

We'll cover the following

Introduction

Arrays in Scala are collections which come under the sequence class. They are a mutable collection, hence, when elements in an array are modified, the original array is updated. However, it is important to mention here that in one-way arrays are also immutable as the size of the array cannot change.

Scala arrays are represented as Java arrays. So, the Scala array `Array[Int]`, is represented as the Java array `int[]` and the Scala array `Array[String]` is represented as the Java array `string[]`, and so on. That being said, Scala arrays have a lot more to offer and in the next two lessons, we will see exactly how that is.

Creating and Populating an Array

While there are multiple ways of populating and defining an array in Scala, the basic syntax of creating an array from scratch remains the same with the keyword `Array` followed by `()` in which you mention the elements you want to populate

your array with.

```
val arrayName = Array(element 1, element 2,...,element n)
```

The `Array` method is creating an `Array` object.

Let's look at a simple example where we want to create and populate an array with the first ten positive integers.

This code requires the following environment variables to execute:

LANG C.UTF-8

```
val intArray = Array(1,2,3,4,5,6,7,8,9,10)

// Driver Code
intArray.foreach(println)
```

With Sequence Methods

Remember how we briefly discussed classes in the previous lesson? We mentioned that classes are used to create objects and also specify which operations the object can use (built-in methods). The great thing about Scala arrays is that they can use all the operations of the sequence class. Let's use some of those operations to populate an array.

In the example below, we will populate another array with a few selective elements from the `intArray` in the previous example.

This code requires the following environment variables to execute:

LANG C.UTF-8

```
val intArray = Array(1,2,3,4,5,6,7,8,9,10)
val evenArray = intArray.filter(_ % 2 == 0)

// Driver Code
evenArray.foreach(println)
```

The `filter` method is being called on an array and can be called on any collection

which is of sequence type. It takes a `Boolean` type expression as an argument which specifies the criteria each element of the new array should have. It will act on every element of the old array checking to see which element, when inserted inside the expression, will return `true` or `false`. If the expression returns `true`, the element will be added to the new array, and if the expression returns `false`, the element will be excluded from the new array.

The result of the code above would give us an array with five elements: $[2, 4, 6, 8, 10]$. Now we want to double each element and want the elements in reverse. Luck has it that we have sequence operations which can help us without the code getting too long.

This code requires the following environment variables to execute:

LANG	C.UTF-8

```
val intArray = Array(1,2,3,4,5,6,7,8,9,10)
val evenArray = intArray.filter(_ % 2 == 0)
val doubleArray = evenArray.map(_ * 2)
val finalArray = doubleArray.reverse

// Driver Code
finalArray.foreach(println)
```

Success! We were able to get the array we wanted.

With `new`

But what if we don't initially know what we want to populate our array with? For this, we will use the `new` keyword. `new` is used to instantiate objects.

This code requires the following environment variables to execute:

LANG	C.UTF-8

```
val colorArray = new Array[String](3)

// Driver Code
colorArray.foreach(println)
```

The code above is initializing an array `colorArray` which has size 3 and can store

elements of type `String`.

When we print the empty `colorArray`, `null` is displayed three times (size of the array). This is simply letting us know that the array doesn't have any elements.

Let's now populate the array.

This code requires the following environment variables to execute:

LANG C.UTF-8

```
val colorArray = new Array[String](3)
colorArray(0) = "red"
colorArray(1) = "blue"
colorArray(2) = "yellow"

// Driver Code
colorArray.foreach(println)
```

A Handful of Methods

Below is a list of some `Array` methods you can use to populate an array.

Using `range`

This code requires the following environment variables to execute:

LANG C.UTF-8

```
val array1 = Array.range(0, 5) //range(start, number of elements)
array1.foreach(println)
```

This code requires the following environment variables to execute:

LANG C.UTF-8

```
val array2 = Array.range(0, 10, 3) //range(start, end, interval)
array2.foreach(println)
```

Using `fill`

This code requires the following environment variables to execute:

LANG C.UTF-8

```
val array3 = Array.fill(2)("educative") //fill(number of elements)(value)
array3.foreach(println)
```

Using `toArray`

This code requires the following environment variables to execute:

LANG C.UTF-8

```
val array4 = "hello".toArray //converts arguments to an array
array4.foreach(println)
```

Accessing Elements of an Array

To access an element of an array you simply mention the name of the array from which you want to access an element followed by `()` in which the index of the element will be inserted.

```
arrayName(index of element 2)
```

This code requires the following environment variables to execute:

LANG C.UTF-8

```
val intArray = Array(17, 34, 23, 6, 50)
val elem = intArray(3)

println(elem)
```

Length of an Array

To get the length of an Array, or the length of any collection for that matter, you can use the `length` method. It is called on a collection and doesn't take any

parameters. Running the code below will give you the length of `intArray`.

This code requires the following environment variables to execute:

LANG	C.UTF-8

```
val intArray = Array(17, 34, 23, 6, 50)
val len = intArray.length

println(len)
```

You cannot really modify the elements of a Scala array. As the size is static, elements cannot be added or removed. But Scala has a solution.

Let’s move on to the next collection, ArrayBuffer, in the next lesson.