# Arrays & ArrayLists

**Stefan Zorcic**

Arrays and arrayLists are data structures that are commonly used in programming for their ability to store multiple objects/values into a single variable.

## Arrays:

A datatype and an identifier is need to declare an array in Java. For example, below the declared array is uses the identifier “trees” and is using the datatype String.

```
String [] trees;
```

Arrays have the property of being immutable, meaning that whatever size (number of elements) the array is initialized to have is a fixed amount and cannot be changed later in the program.

The size of an array can be initialized in two ways: array literals and explicit declaration. Array literals work by declaring all the elements in the array between two curly braces as demonstrated below.

```
String [] trees = {"Oak","Birch","Spruce"};
```

The second way the size of an array can be declared is by explicit declaration as demonstrated below. The example below uses the keyword “new” followed by the same datatype as initialized finally ended with two brackets with a number in-between indicating the size of the array; in this case five elements.

```
String [] trees = new String [5];
```

When using explicit declaration the elements in the array are declared to be null and can be assigned a value later in the program.

Arrays are 0-indexed meaning the first element in the array is identified by index 0, the second element identified by index 1, the third element identified by index 2, etc. To assign a value to an index in the array you must declare the index and use the same datatype the array is defined as.

```
String [] trees = new String [5];
trees[0]="Oak";
```

The example above assigned the value “Oak” to the first element of the array. Displaying elements of an array follows a similar syntax to assigning values. Below the code prints all the values of the “trees” array using the **.length** command.

***.length** returns the size (i.e. number of elements) of the array

```
String [] trees = new String [3];

trees[0]="Oak";
trees[1]="Birch";
trees[2]="Spruce";

for (int i=0;i<trees.length;i++){
   System.out.println(trees[i]);
}
```

The output of the code is as follows:

```
Oak
Birch
Spruce
```

**ArrayLists:**

ArrayLists are from the java.util package so to use ArrayLists in your programming you must first use the appropriate input statements at the head of the code.

```
import java.util.ArrayList;   or   import java.util.*;
```

ArrayLists are declared using a datatype and an identifier like arrays, the syntax/example is below.

```
ArrayList <String> trees = new ArrayList<String>();
```

ArrayLists are **not** immutable meaning elements can be append to the end of the data structure using the **.add** command. The **.add** command append the value to the array-list. Conversely the **.get** command returns an element at a specific index. An example where both commands are used is below.

*Note **.size()** is the same as .length however used for ArrayList's

```
ArrayList <String> trees = new ArrayList<String>();

trees.add("Oak");
trees.add("Birch");
trees.add("Spruce");

for (int i=0;i<trees.size();i++){
   System.out.println(trees.get(i));
}
```

Output of the code:

```
Oak
Birch
Spruce
```

Finally both arrays and array-lists have a special for each loop that iterates through each element of the data structure without needing to specify index, an example is below.

```
ArrayList <String> trees = new ArrayList<String>();

trees.add("Oak");
trees.add("Birch");
trees.add("Spruce");

for (String x : trees){
   System.out.println(x);
}
```

Output of the code:

```
Oak
Birch
Spruce
```

*For all commands & uses of arrays and array-lists visit official Oracle Documentation