Java Basics Unit

Lesson 8: Arrays

The Learning House

427 S 4th Street #300

Louisville KY 40202

Lesson 08: Arrays

Overview

In this lesson, we'll learn how to create, access, and manipulate **arrays** in our Java programs

What is an Array?

An array is a container that holds a fixed number of values of the same data type. The key words in the definition are **fixed number** and **same data type**. The length of the array (i.e. the number of values it can hold) is specified when the array is created and cannot be changed. Further, the type of the data that can be stored in the array is specified when it is created and only values of that data type may be stored in the array.

The data values in the array are referred to as **elements** and the location of each element is indicated by a numerical index. The index numbering starts at 0 and is sequential with the last element's index being equal to the number of elements in the array minus 1. For example, Figure 1 (from the Oracle Java Tutorials) shows an array that holds 10 items. The indexes of the elements range from 0 to 9.

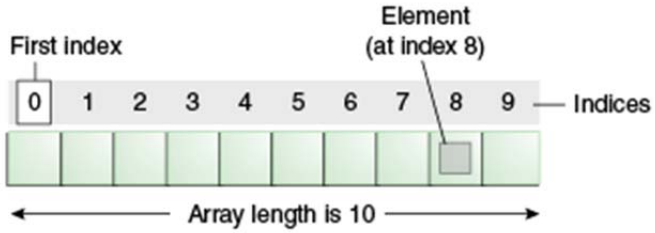

Figure 1: 10 Element Array

Array Declaration

Arrays must be declared and initialized just like variables of any other data type. However, because they are made up of more than one value, they require special syntax for declaration and initialization. The following code declares an array of ints:

```
int[] teamScores;
```

Note the use of the square brackets [ ] which indicate that this should be an array of ints rather than a single int variable. The next line reserves enough memory for 10 ints:

```
teamScores = new int[10];
```

Array Initialization

After declaring and reserving space for our array, we must initialize the elements. This is done using the assignment operator and the array indexes:

```
teamScores[0] = 2;    // initialize first element
teamScores[1] = 45;   // initialize second element
teamScores[2] = 4;    // etc...
teamScores[3] = 8;
teamScores[4] = 99;
teamScores[5] = 23;
teamScores[6] = 67;
teamScores[7] = 1;
teamScores[8] = 88;
teamScores[9] = 42;
```

One important thing to keep in mind is that **index numbering starts at 0, NOT at 1**.

There is a more convenient way to declare and initialize array as well. Using this approach, you will provide the values for the array at the same time you declare the array:

```
int[] teamScores = {
     2, 45, 4, 8, 99,
     23, 67, 1, 88, 42
 };
```

Accessing Array Data Elements

Now that we have values for each of elements, we will look at how we can access and manipulate the values of each element in the array.

Index Operator

We access the individual data elements of the array using the **index operator**. You saw examples of this above in the initialization section. The following line copies the value of the third element and puts it into the variable called currentScore. As mentioned above, note that **index numbering starts at 0, NOT at 1**.

```
int currentScore = teamScores[2];
```

Changing a value in an element is basically the same as initializing that element. The following line sets the fifth element to 109:

```
teamScores[4] = 109;
```

For Loop

Using a for loop is a convenient way to visit each element in an array. For example, to list out all of the scores in the teamScores array, you would write the following code:

Notes:

1. We use int i for the counter just like any other for loop. It starts at 0 because we want to start with the first element in the array, which has an index of 0.
2. Our termination condition says that we'll keep going as long as i < the length of the teamScores array.
3. We increment i just like any other for loop.
4. We access each element in the array by using the value of i as the index.

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

Enhanced For Loop

We can also use an **enhanced for loop** to visit each element in an array. The following example prints the value of each element of teamScores to the screen. The enhanced for loop essentially says that we want to grab each element out of the array and put it into a variable called **num**. We then print the current value of num to the screen each time through the loop:

```
for (int num : teamScores) {
    System.out.println("Element = " + num);
}
```

Two-Dimensional Arrays

Java allows for multidimensional arrays. Multidimensional arrays are simply arrays whose elements are also arrays. This means that the 'rows' in a multidimensional array can be of different lengths (see the example below). When declaring a multidimensional array, simply add a set of brackets for each dimension of the array. The following example declares and initializes a two-dimensional array and then accesses each element of the array:

```
String[][] cityTeamNames = {
    {"Cleveland", "Browns", "Cavs", "Indians"},
    {"Columbus", "Bluejackets", "Buckeyes"},
    {"Pittsburgh", "Steelers", "Pirates", "Penguins"}
};

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