Basic Arrays

Keleah Bliven – BIT 115 – Professor Panitz – Spring Quarter 2011 – Section 1

Assignment Definition And General Feedback By Michael Panitz
at Cascadia Community College (http://www.cascadia.edu)

Table of Contents:

Quick Summary:

In Java, you can create a container object, known as an array, to store and organize data. An array is commonly used as a basic method to store a large number of values of the same type. An array consists of several blocks of data that can be referred to by a common name. Each block of data inside an array is known as an element, which can be referred to by the element's numerical index.

This method is not only an easier and quicker way to record information, but it is also helpful for storing mass amounts of data. An array is an object, in Java, that has the capability to hold a certain number of values. In an array, there can only be one type of value (ea. integers, characters, strings, and etcetera). The number of values in an array is set during its creation. Once an array has been created, the program may not change the length of the array. The numerical index, by which elements can be called, begins at the integer zero. For example, if you wanted to create an array that contained ten elements, the last element would be identified by the integer nine.

When To Use This / Avoid This:

There are several situations in which you would want to use this technique. An array is most commonly used to store a large amount of values as a single block of data. Elements within an array can be one of the primitive data types: byte, short, long, int, float, double, boolean, char, or String. They can also be a more complex type (ea. objects), but this paper will only be focusing on the simpler types.

As an example of when an array would be used, imagine that you were a professor and you needed to record various information about each individual student in your class (name, grades, etcetera). This is where an array would come in handy. Instead of diligently writing out all of the information for each student (which would take an enormous amount of time), you could use an

array to contain and organize all of the data. The array would be the entire block of data that could be called under a common name and the elements would be the blocks of data that would contain each of the student's information. You could also use this technique in many other scenarios.

There are several situations in which you would want to avoid this technique. An array has a fixed capacity for values; the number of values in array cannot be changed after the array has been created. This could be a problem if you were trying to use an array to record data that had growth or decay (or needed to change the number of elements being used).

As an example of when not to use an array, imagine that you need to record information (name, address, phone number, and etcetera) about people who were joining a baseball team. Once you created an array for the baseball team you could not resize the array (without revising the previously written code to create the arrays and the elements it contained) to include new people (or people who joined late). There are many other scenarios in which using an array would not be appropriate.

Example Of Usage:

Let us say that you needed to store information for a group of people in an orchestra and you needed to create a String array for the orchestra sections and each of the elements would be a section in the orchestra (first violins, harps, trumpets, and etcetera).

Please note that there are many types of data that you can store in an array. The type used in this example is the String type, but you could use any one of the primitive data types: byte, short, long, int, float, double, boolean, char, or String.

There are two steps in this example of creating a basic String array.

First, you need to declare the array and initialize the elements by typing, 'String[] theNameOfYourStringArray = {"element1", "element2", "element3", "etcetera"};' (of course, you could name the array something other than theNameOfYourStringArray and change the strings as needed). Lines 6 and 7 in the example below create the array and initialize the elements.

Second, now that you have created a String array, you should have the program print out the results of the creation of your array. There are a few different ways to have the program output the information in the array. I have shown a couple specific ways in the example below. One way to display the output of the array would be to use the System.out.println command (see lines 10 and 11 in the example below). Your code should follow this outline: System.out.println ("Here is the data stored in the first element at the index 0: " + theNameOfYourStringArray[0]); (lines 10 and 11 illustrate this method).

For a quicker, easier way to print out the rest of the values in the array, I have used a for loop (as shown below in lines 14 through 17). I used a for loop to create an integer to count through each of the elements and display them to the console.

Now you have successfully created and printed out an array!

Line	Code
1.	`public class Orchestra_Array`
2.	`{`
3.	`public static void main(String[] args)`
4.	`{`
5.	`//declares the array and intializes the elements within the array`
6.	`String[] orchestraSections = {"firstViolins", "secondViolins", "violas", "cellos", "doubleBasses", "harps", "flutes", "clarinets",`
7.	`"bassoons", "oboes", "frenchHorns", "trumpets", "trombones", "tubas", "percussion"};`
8.	
9.	`//prints out each element separately`
10.	`System.out.println("The element stored at index 0 is: " + orchestraSections[0]);`
11.	`System.out.println("The element stored at index 1 is: " + orchestraSections[1]);`
12.	
13.	`//prints out the rest of the elements using a loop to save time`
14.	`for(int i = 2; i < 15; i++)`
15.	`{`
16.	`System.out.println("The element stored at index " + i + " is: " + orchestraSections[i]);`
17.	`}`
18.	
19.	`}`

Example Trace:

Overall, this technique allows you to create a basic array that contains elements that are organized by a numerical index beginning with the integer zero. This method is an easier and faster way to record mass amounts of data and organize the information.

To explain how arrays work in greater detail, here is a trace of the above program with some additional explanation afterwards.

Please note that I put all of the values for each element into one column in the table below (I could not fit a column for each element in this layout). The values will be displayed in the following format: [0] - this is the value stored at index 0 (and the same will be used for the rest of the values). Also, just a quick side note, whenever an array of Strings is created and no values have been assigned to the elements, Java will automatically store the word 'null' in each element.

Line #	Program Statement	Value at each index (beginning from 0 and ending at 14)	True/False	i	Output	What does this line do?
1	`public class Orchestra_Array`	-	-	-	-	This line creates the class.
3	`public static void main(String[] args)`	-	-	-	-	This line creates main.
6	`String[] orchestraSections = {"firstViolins", "secondViolins", "violas", "cellos", "doubleBasses", "harps", "flutes", "clarinets"`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets	-	-	-	This line (and line 7) creates the array and initializes the elements.
7	`"bassoons", "oboes", "frenchHorns", "trumpets", "trombones", "tubas", "percussion"};`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets [8] - bassoons [9] - oboes [10] - frenchHorns [11] - trumpets [12] - trombones [13] - tubas	-	-	-	This line (and line 6) creates the array and initializes the elements.

		[14] - percussion				
10	`System.out.println("The element stored at index 0 is: " + orchestraSections[0]);`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets [8] - bassoons [9] - oboes [10] - frenchHorns [11] - trumpets [12] - trombones [13] - tubas [14] - percussion	-	-	`The element stored at index 0 is: firstViolins`	This line prints out the element stored at index 0.
11	`System.out.println("The element stored at index 1 is: " + orchestraSections[1]);`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets [8] - bassoons [9] - oboes [10] - frenchHorns [11] - trumpets [12] - trombones [13] - tubas [14] - percussion	-	-	`The element stored at index 1 is: secondViolins`	This line prints out the element stored at index 1.
14	`for(int i = 2; i < 15; i++)`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets [8] - bassoons [9] - oboes [10] - frenchHorns [11] - trumpets [12] - trombones [13] - tubas [14] - percussion	True	2	-	This line creates a for loop and checks to make sure integer i is under 15.

16	`System.out.println("The element stored at index " + i + " is: " + orchestraSections[i]);`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets [8] - bassoons [9] - oboes [10] - frenchHorns [11] - trumpets [12] - trombones [13] - tubas [14] - percussion	-	2	`The element stored at index 2 is: violas`	This line prints out the element located at index 2.
14	`for(int i = 2; i < 15; i++)`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets [8] - bassoons [9] - oboes [10] - frenchHorns [11] - trumpets [12] - trombones [13] - tubas [14] - percussion	True	3	-	This line stills checks to see if the integer i is under 15.
16	`System.out.println("The element stored at index " + i + " is: " + orchestraSections[i]);`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets [8] - bassoons [9] - oboes [10] - frenchHorns [11] - trumpets [12] - trombones [13] - tubas [14] - percussion	-	3	`The element stored at index 3 is: cellos`	This line prints out the element located at index 3.
14	`for(int i = 2; i < 15; i++)`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets [8] - bassoons [9] - oboes	True	4	-	This line stills checks to see if the integer i is under 15.

		[10] - frenchHorns [11] - trumpets [12] - trombones [13] - tubas [14] - percussion				
16	`System.out.println("The element stored at index " + i + " is: " + orchestraSections[i]);`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets [8] - bassoons [9] - oboes [10] - frenchHorns [11] - trumpets [12] - trombones [13] - tubas [14] - percussion	-	4	`The element stored at index 4 is: doubleBasses`	This line prints out the element located at index 4.
14	`for(int i = 2; i < 15; i++)`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets [8] - bassoons [9] - oboes [10] - frenchHorns [11] - trumpets [12] - trombones [13] - tubas [14] - percussion	True	5	–	This line stills checks to see if the integer i is under 15.
16	`System.out.println("The element stored at index " + i + " is: " + orchestraSections[i]);`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets [8] - bassoons [9] - oboes [10] - frenchHorns [11] - trumpets [12] - trombones [13] - tubas [14] - percussion	-	5	`The element stored at index 5 is: harps`	This line prints out the element located at index 5.

14	`for(int i = 2; i < 15; i++)`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets [8] - bassoons [9] - oboes [10] - frenchHorns [11] - trumpets [12] - trombones [13] - tubas [14] - percussion	True	6	-	This line stills checks to see if the integer i is under 15.
16	`System.out.println("The element stored at index " + i + " is: " + orchestraSections[i]);`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets [8] - bassoons [9] - oboes [10] - frenchHorns [11] - trumpets [12] - trombones [13] - tubas [14] - percussion	-	6	`The element stored at index 6 is: flutes`	This line prints out the element located at index 6.
14	`for(int i = 2; i < 15; i++)`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets [8] - bassoons [9] - oboes [10] - frenchHorns [11] - trumpets [12] - trombones [13] - tubas [14] - percussion	True	7	-	This line stills checks to see if the integer i is under 15.
16	`System.out.println("The element stored at index " + i + " is: " + orchestraSections[i]);`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets [8] - bassoons [9] - oboes	-	7	`The element stored at index 7 is: clarinets`	This line prints out the element located at index 7.

		[10] - frenchHorns [11] - trumpets [12] - trombones [13] - tubas [14] - percussion				
14	`for(int i = 2; i < 15; i++)`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets [8] - bassoons [9] - oboes [10] - frenchHorns [11] - trumpets [12] - trombones [13] - tubas [14] - percussion	True	8	-	This line stills checks to see if the integer i is under 15.
16	`System.out.println("The element stored at index " + i + " is: " + orchestraSections[i]);`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets [8] - bassoons [9] - oboes [10] - frenchHorns [11] - trumpets [12] - trombones [13] - tubas [14] - percussion	-	8	`The element stored at index 8 is: bassoons`	This line prints out the element located at index 8.
14	`for(int i = 2; i < 15; i++)`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets [8] - bassoons [9] - oboes [10] - frenchHorns [11] - trumpets [12] - trombones [13] - tubas [14] - percussion	True	9	-	This line stills checks to see if the integer i is under 15.

16	`System.out.println("The element stored at index " + i + " is: " + orchestraSections[i]);`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets [8] - bassoons [9] - oboes [10] - frenchHorns [11] - trumpets [12] - trombones [13] - tubas [14] - percussion	-	9	`The element stored at index 9 is: oboes`	This line prints out the element located at index 9.
14	`for(int i = 2; i < 15; i++)`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets [8] - bassoons [9] - oboes [10] - frenchHorns [11] - trumpets [12] - trombones [13] - tubas [14] - percussion	True	10	-	This line stills checks to see if the integer i is under 15.
16	`System.out.println("The element stored at index " + i + " is: " + orchestraSections[i]);`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets [8] - bassoons [9] - oboes [10] - frenchHorns [11] - trumpets [12] - trombones [13] - tubas [14] - percussion	-	10	`The element stored at index 10 is: frenchHorns`	This line prints out the element located at index10.
14	`for(int i = 2; i < 15; i++)`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets [8] - bassoons [9] - oboes	True	11	-	This line stills checks to see if the integer i is under 15.

		[10] - frenchHorns [11] - trumpets [12] - trombones [13] - tubas [14] - percussion				
16	`System.out.println("The element stored at index " + i + " is: " + orchestraSections[i]);`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets [8] - bassoons [9] - oboes [10] - frenchHorns [11] - trumpets [12] - trombones [13] - tubas [14] - percussion	-	11	`The element stored at index 11 is: trumpets`	This line prints out the element located at index 11.
14	`for(int i = 2; i < 15; i++)`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets [8] - bassoons [9] - oboes [10] - frenchHorns [11] - trumpets [12] - trombones [13] - tubas [14] - percussion	True	12	-	This line stills checks to see if the integer i is under 15.
16	`System.out.println("The element stored at index " + i + " is: " + orchestraSections[i]);`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets [8] - bassoons [9] - oboes [10] - frenchHorns [11] - trumpets [12] - trombones [13] - tubas [14] - percussion	-	12	`The element stored at index 12 is: trombones`	This line prints out the element located at index 12.

14	`for(int i = 2; i < 15; i++)`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets [8] - bassoons [9] - oboes [10] - frenchHorns [11] - trumpets [12] - trombones [13] - tubas [14] - percussion	True	13	-	This line stills checks to see if the integer i is under 15.
16	`System.out.println("The element stored at index " + i + " is: " + orchestraSections[i]);`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets [8] - bassoons [9] - oboes [10] - frenchHorns [11] - trumpets [12] - trombones [13] - tubas [14] - percussion	-	13	`The element stored at index 13 is: tubas`	This line prints out the element located at index 13.
14	`for(int i = 2; i < 15; i++)`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets [8] - bassoons [9] - oboes [10] - frenchHorns [11] - trumpets [12] - trombones [13] - tubas [14] - percussion	True	14	-	This line stills checks to see if the integer i is under 15.
16	`System.out.println("The element stored at index " + i + " is: " + orchestraSections[i]);`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets [8] - bassoons [9] - oboes	-	14	`The element stored at index 14 is: percussion`	This line prints out the element located at index 14.

		[10] - frenchHorns [11] - trumpets [12] - trombones [13] - tubas [14] - percussion				
14	`for(int i = 2; i < 15; i++)`	[0] - firstViolins [1] - secondViolins [2] - violas [3] - cellos [4] - doubleBasses [5] - harps [6] - flutes [7] - clarinets [8] - bassoons [9] - oboes [10] - frenchHorns [11] - trumpets [12] - trombones [13] - tubas [14] - percussion	False	15	-	This line stills checks to see if the integer i is under 15.

Syntax Explanation:

In regards to the exact syntax used in creating an array, each of the primitive data types can be used to create array as shown below:

int[] intArray;

byte[] byteArray;

short[] shortArray;

long[] longArray;

float[] floatArray;

double[] doubleArray;

boolean[] booleanArray;

char[] charArray;

String[] stringArray;

To declare an array and initialize the elements by directly assigning a value for each element when the array is created (with whichever data type you choose, but let us use a String array for this example), your code should follow the following pattern:

String[] stringArray = {“Here ”, “is ”, “my ”, “array ”, “of ”, “strings!”};

Please note that straight brackets are used to create the array and the curly brackets are used to initialize the elements in the same line of code.

You could also use the following syntax to create and initialize your array:

stringArray = new String [10];

Please note that you may name your array anything you like. Also, you may change the length of your array to any integer you like.

After you have completed the creation of the array, you must assign a value to each element, with the following syntax:

stringArray[0] = “text stored in the first element”

If you wish to print out the text stored in the array, you could use the System.out.println command.

Similarly, if you wanted to print out the line of code from the example above (also copied below) you would use this line of code:

String[] stringArray = {“Here ”, “is ”, “my ”, “array ”, “of ”, “strings!”};

System.out.println (stringArray [0] + stringArray [1] + stringArray [2] + stringArray [3] + stringArray [4] + stringArray [5]);

And you would see the output: Here is my array of strings!

The original example code with the new syntax/code (highlighted in yellow) is provided in the table below.

Below, you can see the finished program with the differences highlighted in yellow, so it is easier to see what has been added/changed.

Line #	Program Source Code
1.	`public class Orchestra_Array`
2.	`{`
3.	`public static void main(String[] args)`
4.	`{`
5.	`//declares the array and intializes the elements within the array`
6.	`String[] orchestraSections = {"firstViolins", "secondViolins", "violas", "cellos", "doubleBasses", "harps", "flutes", "clarinets",`
7.	`"bassoons", "oboes", "frenchHorns",`

```
        "trumpets", "trombones", "tubas", "percussion"};
8.
9.              //prints out each element separately
10.             System.out.println("The element stored at
index 0 is: " + orchestraSections[0]);
11.             System.out.println("The element stored at
index 1 is: " + orchestraSections[1]);
12.
13.             //prints out the rest of the elements
using a loop to save time
14.             for(int i = 2; i < 15; i++)
15.             {
16.                     System.out.println("The element
stored at index " + i + " is: " +
orchestraSections[i]);
17.             }
18.
19.     }
```

Licensing

Plagiarism

If you believe that some or all of this document infringes on your intellectual property (i.e., part or all of this document is copied from something you've written) please immediately contact Mike Panitz at Cascadia Community College (perhaps using the Faculty And Staff Directory at http://www.cascadia.edu/pages/searchtemplate.aspx)