Basic Arrays

By Ryan Erickson

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

Table of contents:

(Required sections have a * after them in the above list)

Quick Summary:

What is an Array?

One of the harder programming concepts to grasp, and especially to visualize, is an array. Basically, an array is a "list" of memory locations where we can store, access, search, and modify the data in our "list". Similar to declaring variables such as "int", or "String", at the end of the day all an array does is store data. However, with declarations such as "int" or "String", you have to declare each variable separately, and then assign each variable separately. With Arrays, we can construct a large list of integers or Strings (or any other variable type) with fewer commands than declaring each piece of data separately. While this is convenient, it only hints at the true power of an Array. Where most of the power lies in actually in your ability to access, search, modify, sort, and give any other command to our Array in a way that takes a fraction of the time and a fraction of the code (compared to individual variable declaration).

What makes up an Array?

As we said above, an Array is just a "list" of data. However, in order to reference the data later, we must have some sort of location for each data. This is what an INDEX is for. An index in an array is very similar to an index that you might find in a book. However, an index in an array starts at ZERO (0). The data in an array is stored at a particular INDEX number.

Visual Representation of an Array:

arrayName	-
Index #	Data Stored
0	7
1	52
2	3
3	12

When To Use This / Avoid This:

Arrays have a myriad of uses that, among other things, save a giant amount of time for programmers. Instead of declaring 100 pieces of data with "int variableName = value;", we can simply create an array to do the same thing.

Examples of when to use this:

Grabbing 3000 names from EverettCensusData.doc and storing them in a program for later use.

Creating a program which "flips a coin" 50,000 times, records every outcome of every flip, and calculates probabilities/means/standard deviations/etc.

While arrays can be fun and interesting, it is unreasonable to create entire arrays to store just a few pieces of data. If you are representing three of four flips of a coin or just one or two peoples names, then an array is most likely overkill for the task. While you could definitely complete the task using an array, in the long run it may prove more of a hassle. (That being said, the examples below will create arrays that only store small amounts of data ☺)

Example Of Usage:

For our example, we will finally be messing around with an array!

This example contains ***three basic steps***.

First, we must declare the array so that the computer knows what it is!

(ArrayType arrayName[] = new ArrayType[# of objects in your array];)

Second, we tell one of our indexes to hold a specific number (and we do so by hand).

(arrayName[index#] = value;)

Third, we ask the system to print out the values at the first index.

(System.out.println("Index value x currently holds " + arrayName[index#] + ".");)

(System.out.println("Index value y currently holds " + arrayName[index#] + ".");)

Last, we use a for loop to print the rest of the values in the array.

for (int i = 1; i < 5; i++;)

{ System.out.println("Index" + i + "equals " + arrayName[i]); }

Line	Code
1.	public class FirstArray extends Object
2.	{
3.	public static void main(String args[])
4.	{
5.	int dogsAge[] = new int[5]; //Declare your array! (all 5 values currently = 0)
6.	
7.	dogsAge[0] = 8; //tells the first index (remember 0, 1, 2) to hold the value 8
8.	
9.	System.out.println("Index 0 currently holds the # " + dogsAge[0] + ".");
10.	
11.	for (int i = 1; i < 5; i++;)
12.	{ System.out.println("Index" + i + "equals " + dogsAge[i]); }
	}
13.	}

Example Trace:

Above, all we have done is create a very simple array that holds five values (which represent the age of a dog).

In order to go over these details more thoroughly, here is a (partial) trace of the above program, with some additional explanation afterwards

Line #	Program Statement	Index 0	Index 1	Index 2	Index 3	Index 4	i	Output
5	int dogsAge[] = new int[5];	0	0	0	0	0	-	

7	dogsAge[0] = 8;	8	0	0	0	0	-	
9	System.out.println("Index 0 currently holds the # " + dogsAge[0] + ".");	8	0	0	0	0	-	Index 0 currently holds the # 8.
11	for (int i = 1; i < 5; i++;)	8	0	0	0	0	1	-
12	{ System.out.println("Index" + i + "equals " + dogsAge[i]); }	8	0	0	0	0	1	Index 1 currently holds the # 0.
11	for (int i = 1; i < 5; i++;)	8	0	0	0	0	2	
12	{ System.out.println("Index" + i + "equals " + dogsAge[i]); }	8	0	0	0	0	2	Index 2 currently holds the # 0.
11	for (int i = 1; i < 5; i++;)	8	0	0	0	0	3	
12	{ System.out.println("Index" + i + "equals " + dogsAge[i]); }	8	0	0	0	0	3	Index 3 currently holds the # 0.
11	for (int i = 1; i < 5; i++;)	8	0	0	0	0	4	
12	{ System.out.println("Index" + i + "equals " + dogsAge[i]); }	8	0	0	0	0	4	Index 4 currently holds the # 0.
11	for (int i = 1; i < 5; i++;)	8	0	0	0	0	5	

Our trace begins at line 3 at main, as usual. It then drops down to line 5 where it declares the array named dogAge (which contains 5 values). Next it moves down to line 7 where it assigns the value "8" to the dogsAge array at index 0. Then, it moves on to line 9 where it prints one message ("Index 0 currently holds the # 8.") with a System.out.println command and a reference to the index 0 of dogsAge via dogsAge[0]. Finally, it runs a for loop four times (i = 1, i < 5) which prints the same message as in step three, but with a dynamic (changing) reference to the dogsArray rather than a static one.

Syntax Explanation:

The following is a line by line explanation of the syntax of this program.

FirstArray.java

Line #	Program Source Code
1.	public class FirstArray extends Object
2.	{
3.	public static void main(String args[])
4.	{
5.	int dogsAge[] = new int[5]; //Declare your array! (all 5 values currently = 0)
6.	
7.	dogsAge[0] = 8; //tells the first index (remember 0, 1, 2) to hold the value 8
8.	
9.	System.out.println("Index 0 currently holds the # " + dogsAge[0] + ".");
10.	
11.	for (int i = 1; i < 5; i++;)
12.	{ System.out.println("Index" + i + "equals " + dogsAge[i]); }
13.	}
14.	}

Line 1: First we must create a class for this program which has the same name as our file name (minus the .java) and which extends some other class (in this case we extend Object which is unnecessary as every class extends Object however, "extends Object" is left in for clarity and understanding).

Line 2: This is the opening brace for our FirstArray class.

Line 3: We create the main method which is the first method that our JRE will attempt to run when the file is ran.

Line 4: This is the opening brace for our main method.

Line 5: This line creates an array. The syntax for an array requires you to declare the type (*int*) followed by the name (*dogsAge*) with brackets attached to it (*dogsAge[]*) to signify that it is an ARRAY and not just a primitive variable. The syntax then requires you to give more information about the array (I chose to include this all in one line but there are other ways to do it.) The additional information means that you assign (=) a new object (*new*) of type int (*int*) to the dogsAge[] array along with a specification of how many indexes the array should have (*int[5]*). It is a syntax error to leave the brackets blank as in (*…=new int[];*).

Line 7: This line references a piece of data (*dogsAge[0]*) and then assigns a new integer to that location (= *8;*). Just like with a normal variable (*int age; age = 43;*) you are simply replacing the previous data with your new data. However, with an array you must reference the location with *arrayName[index#]* instead of just *variableName*.

Line 9: In order to print out a message that references an array, we must again use our special array reference (like in step 7) to tell the compiler that we are talking about an array index and not just a regular variable. In this case we reference (*dogsAge[0]*) which is equal to 8, so the number 8 prints on the screen thanks to the (*+ dogsAge[0]*) piece of the command.

Line 11: This line creates a for loop using the (init; condition; step) syntax and it will run a total of four times starting at i = 1. The init creates a variable which will be used for counting, the condition allows the for loop to run as long as it returns true, and the step is the manipulation that you want to do to your counting variable after each loop (in this case i++ or add one to i).

Line 12: This bit of code uses multiple concepts to do one simple thing: print out a message which says the index # and the value at that index #. First, we use the System.out.println(" "); command because we want to print a message. Second, we display some text explaining what we are displaying to the user *("Index" + "Equals"*). Last, we add our variables in to the equation so that it prints out a visual representation of the variables (*i and dogsAge[i]*). For each repetition of the loop, i will be a different number. Therefore, both the printed i and the i in dogsAge[i] will change for each repetition of the loop. The first loop will run with i=1 which will print out 1 and print out the value of dogsAge at index # 1.

Help With The Logic:

While the arrays shown in this guide are very simple, and as a result lacking any practical use, please do not think that arrays are simple, boring tools. One of the more cool things you can do with arrays, once you have some practice with them, is execute one (or thousands) of commands on one (or thousands) of pieces of data with very little work on your end. For instance, with an array you could take the US Census data of every person in the USA and, with one or two lines of code, (like the for loop above) print every single piece of data to the console. Doing this any other way would require as many lines of code as there are pieces of data you wanted to print. (declare each piece of data separately, then print each data with its unique name).

Important Details:

- "ArrayType arrayName[];" (the declaration) and "arrayName = new ArrayType[# of objects]" can easily be combined or broken up as you wish (as long as you have the proper syntax). Quite similar to "Robot Karel;" and "Karel = new Robot(city, street, ave, dir, things);", there are multiple ways to assemble an array. However, for clarity you may want just keep it all together so you don't end up forgetting a part of the array.

Mistakes People Commonly Make (And How To Fix Them):

- **Mistake 1: Attempting to reference the third value in an array with the number [3].**

  This mistake is often made because our normal method of counting objects conflicts with the way we number objects in arrays. Under normal conditions

if you had two fruits in front of you the first would be fruit 1, and the second fruit 2. However, with arrays, numbering starts at ZERO (0) and ends with one less than the number of values (n – 1).

- **Mistake 2: Often times, when new to arrays, one will attempt to reference an array with just the array's name.**

  For instance, it is very easy to accidentally leave out the braces after the arrayName since it is a very similar process to reference a regular variable.

  "int myArray;" (WRONG) vs "int myArray[7];" (CORRECT)

  In order to fix this mistake, we simply must remember that with an array you have to reference an INDEX in your list of data instead of just referencing a name.

Hyperlinks

The official Java website's info on arrays:

http://download.oracle.com/javase/tutorial/java/nutsandbolts/arrays.html

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)