**CSc 110 Lab 7**

**Objectives:**

1. Declare, define, initialize, print and use arrays
2. Practice writing methods for arrays
3. Set up for next Assignment (#6)
4. Reading in command-line parameters

**Review:**

A possible solution for the file input exercise from last lab is here: [Ex2.java](http://webhome.csc.uvic.ca/~csc110l/2012_5/Lab7/code/Ex2.java)

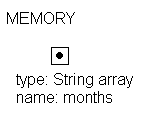
A solution to the Navigation exercise (with while-loops) was posted at the bottom of last lab.

**What Are Arrays?**

An Array is "an indexed structure that holds multiple values of the same type." (Suggest reading pp. 426-433 and 445 in 2nd ed. textbook. Chapter 7 is on arrays.)

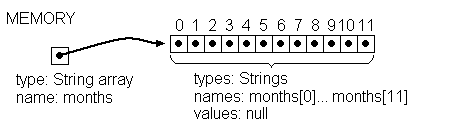
Declaring an array:

String [] months;



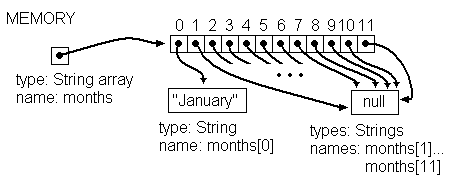
Defining an array:

months = new String [12];



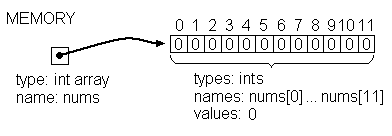
Assigning one value in the array:

months[0] = "January";



Declaring and defining an array of ints:

int [] nums = new int [12];



**EXERCISES**

**Exercise 1 All About Arrays**

In this exercise you are given a program that defines 4 arrays.  Each array is to be initialized and printed.  At the end of the exercise you should be able to answer the questions that follow.  There are 3 init methods and 2 print methods already written and you should not have to write more than 1 additional method to complete the program.

a) Download [ArrayExamples.java](http://webhome.csc.uvic.ca/~csc110l/2012_5/Lab7/code/ArrayExamples.java).  You will also need  [inputStrings.txt](http://webhome.csc.uvic.ca/~csc110l/2012_5/Lab7/code/inputStrings.txt), in the same directory (folder).

Given that declaring a variable is the assigning of a variable name and defining is the assigning of memory for a variable to reference and that initialization is the assigning of value(s) answer the next 3  Questions:

1. Which array(s) is/are declared, defined and initialized in a single statement.
2. Which array(s) is/are declared in one statement, defined in another and then initialized in a method.
3. Which array(s) are defined and initialized in a method.  
     
   Answer the rest of the questions based on this exercise
4. By the end of the exercise you will have 3 methods called initArray, and 3 methods called printArray.  How are each of the 4 arrays initialized and printed?  What determines which "init" or "printArray" method is used when the methods are called from the main?
5. Explain why two init methods change the array in the main but do not need to return anything.
6. Why does one method return an array?

b) Board exercise class participation and discussion. What's in a name?  Using the references in ArrayExamples.java answer the following:

After initialization what is the **type**of

* array1?
* array1[2]?

What is the **value**of array1[2]?  
  
By the end of the main which of the following 4 expressions (if any) will generate an error if embedded in a print statement?

1. array4[0] \* array1[2]
2. array3[0] + array4[4]
3. ((array1[2] \* array1[2]) == 9)
4. array4 = array2

At the end of the statement         String [] array3;  
What is the value of array3?  
  
At the end of the statement         array3 = new String[count];  
What is the value of array3[0]?

**String [] args**

At last: what is that array you've been passing into your main method in every program?

It is an array of Strings, from the command line:

javac Args.java

java Args.java these are command line arguments

Try it: [Args.java](https://connex.csc.uvic.ca/access/content/group/766cb70a-9ef1-463a-9e71-78eb524c1e1d/lab%20resources/Lab7/code/Args.java).

**Paper Exercises: Quick Code Arrays**

Writing or tracing methods that involve arrays are a fundamental exit skill in CSc 110.  The actual implementation of array code is, for the most part,  staight forward once the algortithm is known.

The following exercises should be carried out on paper.  The exercises should add to your skill set as a programmer, help with the current and the next assignment, and better prepare you for this segment in the final exam.  
  
The TA will first audit to see how people are doing then write snippits on the board to get people past sticking points.  The TA may use pseudocode or assume the existence of a helper method especially when considering special cases.   
  
For each of the following first write a shell for the method leaving room for the declaration of local variables and a for loop which is almost always used with an array.

a) Write a method printArray that given an array of doubles prints it.  
  
b) Write a method minPosArray that given an array of doubles and a start position returns the position in the rest of the array that contains the smallest value.  
  
c) Write a method swapPos that given an Array of doubles and two positions in that array swaps the values at those positions.  (No for loop required here.)  
  
d) Write a method SortArray that given an array of doubles loops through the array, and for each ith position from 1 to length of the array, uses minPosArray and swapPos, sort the smallest value into position i.  
  
e) Write a method called add2Arrays that given two arrays of doubles returns a third array where each position is the sum of the values at the same positions in the input arrays.  Assume both arrays are the same length.  Later discuss what would be done if one is shorter.  
  
 f) Write a method called mergeArrays which given two arrays of doubles ( assume they are exactly the same size) declares, populates and returns a single array of twice the size where the zeroth, 2nd 4th ... end positions of the new array have the first input array's values, and the 1st 3rd 5th ... end have the second input arrays values.  
  
g) (Challenge) Write a method mergeSortedArrays given two sorted arrays a and b, in returns a single sorted array.  Duplicates are allowed and preserved since a separate method could be wrriten to remove duplicates.  A possible solution is that the larger 3rd array would have installed at it's current position the smaller of the next of a or next of b.

You can try your solutions out in code once you have planned them on paper. This is one way to check them. Sample solutions will be available next week, showing one way to code the methods.

**Sound Exercise**

You will need to assemble a folder with a sound file and with the media.jar.  You will need to compile and run with the media jar.  A bare bones shell program is supplied.  Get this all to work together before you start to define any doubles arrays or work on any methods.  
  
Since you need headphones to hear your sound you only have the program output to tell you that it played the file.  
  
You will need the following files all place in the same folder and not on the H:Drive for running.  (Save your folder to the H:Drive before you leave.)

* [media.jar](http://webhome.csc.uvic.ca/~csc110l/2012_5/Lab7/code/media.jar)
* [two\_second.au](http://webhome.csc.uvic.ca/~csc110l/2012_5/Lab7/code/two_second.au)
* [MySound.java](http://webhome.csc.uvic.ca/~csc110l/2012_5/Lab7/code/MySound.java)

Compile and run MySound.java in place of  (MyJavaApp.java) with classpath explicitly set to the media.jar (from the assignment):

|  |
| --- |
| http://webhome.csc.uvic.ca/~csc110l/2012_5/Lab7/images/compile.jpg |

1. Copy the two coded methods (setSampleValues and getSampleValues) from the [lab7Sound.txt](http://webhome.csc.uvic.ca/~csc110l/2012_5/Lab7/lab7Sound.txt) into MySound.java
2. Test. Remember to compile and run with the command explained above.
3. Write a method which uses a for-loop to **print** out an array of doubles
4. Test with the array of doubles from the sound (samples)
5. Test with a new array of doubles, which isn't as long (for example 10 doubles)

**Looking back...**

In this lab you'll have practiced and/or seen examples of:

* Arrays: declaring, defining, initializing, assigning, indexing and referencing.
* Writing methods which print and edit arrays (often using for-loops)
* More return statements and passing-by-reference

Carry on working on your assignment and asking any questions you have about the above.

**The End**