Note: there is no included solution program for lesson 13.

For this lesson we'll step back from object-oriented programming for just a bit and go back to some more basic syntax/variable/data stuff. Specifically, arrays, and the "ArrayList" syntax. The concept of arrays is general to most programming languages, but "ArrayList" is specific to Java. Other languages have similar constructs.

An array is a set of variables of the same type. For example, you could make an array of days that has seven items, and call it a week. Or you could make an array of months with 12 objects, and call it a year.

Of course, you could create whatever variables you want to create *without* putting them in an array. For example, if you had a class called "Day" and you wanted to make a week, you could declare seven different Day objects, and you could call them monday, tuesday, etc. However, arrays let you group objects together and perform operations on the set. For example, you could write a for loop that does some action on every item in an array. For example, in our week array, it could output the weather for each day. If you have a bunch of individual variables you declared that aren't grouped, this would be difficult or even impossible. For example, you could write seven lines to output the weather for each day of a week, since every week has seven days. But imagine trying to do this for a month. Not only would 31 lines of output be awful to write, but some months don't even have 31 days. The code might work for January but you'd get an error when you tried to output the 30th day of February.

Read through the following page for how to create and use arrays: https://www.tutorialspoint.com/java/java_arrays.htm Pay careful attention to the "processing arrays" section as this shows you how to access individual elements and use the .length property of the array to set up a for loop. You can use the syntax to access an index or an array both to read values and to update them.

Array Exercises

1. Create an integer array with 70 values called "testScores". Populate the array with values; it doesn't matter what the values are. (You may want to use a loop to do this.)
2. Write a loop that calculates and outputs the mean value of the test scores.
3. Write a foreach loop that outputs every value in the array.
4. Create a new integer array, also with 70 values, called "testScoresWhenYouDontStudy". Copy the values from "testScores" into your new array, but subtract 25 from each value.
5. Create an array of seven strings called "days" that has each day of the week as a variable. Output this array using a for loop.

ArrayLists

In Java, arrays are useful but have some limitations. They are of a fixed size, which means that once you create them, you can't change how big they are. This sometimes isn't an issue – for example, if you're trying to keep track of days in a year, you can know that ahead of time and declare your array to be 365 items. However, for some things it's problematic. Say you have a program that calculates a student's grade by simply calculating the mean of all their assignment scores. Every time a new assignment is graded, that score gets added to the array. You decide to store all the scores in an array,

but it's impossible to know how many scores there are going to be ahead of time. In this situation, every time you want to make the array bigger, you would need to declare a new array of a larger size, copy the old array into the new array, and delete the old array. An alternative to this is to use Java's ArrayList, which is basically a convenient wrapper for the basic array syntax. The ArrayList is going to do the actions just described, but it will do it behind the scenes so you don't have to manually do it yourself.

The following page shows the syntax for setting up ArrayLists. ArrayLists are objects so you will use methods on them, e.g. .size() and .get(), instead of just calling .length or accessing an index directly. You don't need to memorize all of these methods just yet, but you should at least read through the page until you get to the part where it says "Collections.min, max." The most basic methods – size, get, set – you do need to know. Note how they use "<" and ">" as part of the syntax. This is kind of weird, but don't worry about it for now. Just go with it and put an object type in between. In this case we'll use "Integer".

You might wonder why we use the word "Integer" instead of "int", which we use to declare integer variables. Don't worry about this for now – just know that Java will automatically convert between "int" and "Integer".

https://www.dotnetperls.com/arraylist-java

ArrayList Exercises

Do the array exercises from above, but use ArrayLists instead of arrays.

Two-dimensional Arrays

A two-dimensional array is basically an "array of arrays". If instead of making an array of integers, you made an array of *arrays of integers*, you would end up with, essentially, a square or grid of integers. For example, you could make an 8x8 two-dimensional array to represent a chessboard. In Java, 2d arrays don't actually need to be squares. The first dimension must have some length – how long your array of arrays is – but the second dimension, the length of *each array within the array*, can vary. Read through this link for some examples and the syntax for how to set these up.

http://www.java67.com/2014/10/how-to-create-and-initialize-two-dimensional-array-java-example.html

Two-Dimensional Array Exercises

1. Create a 10x10 array of integers, and populate it with the numbers 1 to 100. Then print the array, with a new line after each multiple of 10. So your output should look as follows:
   1 2 3 4 5 6 7 8 9 10
   11 12 13 14 15 16 17 18 19 20
   etc.
2. Create an 8x8 "chessboard", by making a 2d array of Boolean values. For this, the value "true" will represent a black square, and "false" will represent a black square. Then output the results. It should look as follows:

false true false true false true false true
true false true false true false true false
etc.