**Lesson 8 – Arrays**

* **Variable** = Symbolic name for a location in memory that is used to store data
  + Example:
    - int age;
      * Sets up a location in memory to store the integer value of a person’s age and calls this location age
* **Array** = Location in memory that stores a set of items of the same type under the same name
  + Example:
    - int[] studentAges;
      * Sets up a location in memory to store the integer values of a set of student ages and calls this location studentAges.
* **Advantages of Arrays**
  + Single location/name for a large amount of data
  + Individual items (elements) may still be accessed
  + Operations on entire data group can be carried out using repetition structures
* **Array Facts**
  + Items in an array are referred by the name of the array and their position in the array
  + An element’s position in the array is called an index, starting at 0
  + Items in an array can be any of Java’s Primitive Datatypes as well as any Object
  + Items in an array must all be of the same type
  + An array cannot have its length altered after creation (length must stay static)
* **Using Arrays (3 steps)**
  + **Create the Array**
    - **Array Declaration**
  + **Give the array a size**
    - **Memory Allocation**
  + **Setting Values of the Array Elements**
    - **Array Initialization**
* **Array Declaration** 
  + Write the type and the variable name
  + The type of the variable is the same as the type of information stored in each element
  + To indicate that it is an array add the [] after the type
    - Example: int[] studentAge;
* **Memory Allocation**
  + Declaring an Array does not create it, it only gives it a name to an array that will be created during memory allocation
  + Allocating memory for the array is like using a constructor for an object
    - E.g *studentAge = new int[12];*
      * Specifies that the array will have 12 elements
      * JVM allocates memory for the array at this time
      * Number inside the bracket can be anything that evaluates to an integer (e.g. another variable)
* **Array Initialization**
  + The array can now be filled with values, one element at a time
    - Process is sped up when using repetition structures
  + Initializer List:
    - *String[] studentNames;*
    - *studentNames = new String[]{“Shalee”, “Bob”};*
    - ***OR***
    - *int[] studentAges = {16, 17, 18};*
* **Accessing Elements** 
  + Elements in an array are referred by the name of their array and their position in the array
    - Position of an element is called an index
    - Array index must always be a non-negative integer value
  + To access all elements in an array you can use a repetition structure
    - Mainly a for loop
* Programmers often use more than 1 array at a time, copying values from one array to another which often leads to errors if not done properly
  + Common mistakes:
    - array1 = array2;
      * This will cause 2 variable names pointing to the same array object
      * The original array1 is now lost
      * Editing the contents of either array will edit the contents of both arrays
    - **To copy data from 1 array to another**
      * Individual elements must be copied from 1 array to another using a repetition structure