CSSE 221: Fundamentals of Software Development Honors Quiz: arrays and ArrayLists

Benjamin Carter, Nathaniel Bonshire, Katherine Lee, Austin May

An array is a list of values or objects. All of the elements (values or objects) in an array must be the same type. Once you initialize an array its size cannot change. The two ways to initialize an array are as follows:

```
type[] arrayName = new type[x];
type[] arrayNameTwo = {1,2,3,4,5,6,7,8,9};
```

In the above declarations x represents the number of elements that are in the array arrayName. That number, or the number of elements within the {} brackets determines the fixed size of the array. The first declaration above does not assign any values to the elements in the array. The second declaration assigns the specified values during initialization. Before you can change or access the value of an element, you must know its index. The indexes in an array begin at zero, not one. Therefore, the first element in an array has an index of zero, second element has an index of one, and so on until the final element which has an index of x-1. The following are a few ways to access and manipulate one dimensional arrays:

```
arrayName[index];     //accesses the value of the element at the index  in [ ]
arrayName[index] = value;     //assigns a value to the element at the index in [ ]
arrayName.length;      //returns the capacity of the array
```

In Java, arrays can be more than one dimension. A 2D array in Java can be visualized as a matrix or table (this one would be 2 x 3):

67	-2	-451
100	42	777

Instead of having one set of brackets, they have two. The first set of brackets contains the number of rows, and the second contains the number of columns. To reference a point in the array, you need to specify both parts of the location. Java can support 3D arrays as well; there would be three sets of brackets instead of two, and so on. There's no limit to how many dimensions an array can have. It is also possible to have a 2D array with a different number of columns within each row; this is called a jagged array. 2D arrays can be initialized as follows:

```
type[][] arrayName = new type[][];
type[][] arrayName = {{row0column0, r1c1,r2c2},{r1c0,r1c1}};
```

A 2D array is really just an array of arrays. Here are a couple of ways to manipulate 2D arrays:

```
arrayName[row][column];    //access the value at index [row][column]
arrayName[row][column] = value;    //sets the value at index [row][column]
arrayName.length;     //returns the number of rows in the 2D array
arrayName[row].length    //returns the number of columns in the stated row
```

Unlike array, an ArrayList has no set size, allowing the size of the list to change by adding or removing objects. ArrayLists are therefore advisable to use over 1D and 2D arrays when representing a set of data where the number of elements is changing frequently. A 2D array can be represented as an ArrayList, as one ArrayList can store other ArrayList objects. The indexes of ArrayLists also begin at zero instead of one. ArrayLists can only hold objects and are initialized like this:

```
ArrayList<Object> listName = new ArrayList<Object>();
ArrayList<Object> listName = new ArrayList<Object>([value1, value2]);
```

Although the add(Object object) method has a O(1), the worst case is O(n) if the object surpasses the memory of the list and the ArrayList must then be resized and copied. The remove() method also has a O(n) as once an object is removed the rest of the objects in the ArrayList must be shifted down an index. The set, size, get, isEmpty, iterator, and listIterator run in constant time.

Here are some basic ways to manipulate ArrayLists:

```
listName.add(index, value);    //adds a value at specified index
listName.remove(index);     //removes the value of specified index
listName.size();     //returns the number of elements in the ArrayList
```

Source: Horstmann, Chapter 7