CECS 174 – Lecture 21 – Arrays Continued

Copying and Comparing Arrays - since arrays are objects and the array variable name holds the address of the location of the array rather than the actual array and it's values, we cannot directly copy or compare the arrays.

Example – We <u>cannot</u> do:

```
int [] array2 = array1; //trying to copy an array
```

or

```
if ( array1 == array2) { //trying to compare arrays
```

Both of these will compile, but they will not do what is intended. In the first part, the address of array1 will be assigned to array2. Any modifications made to array2 will also happen to array1. In the second part, two array addresses are compared rather than the array values.

Instead each item in the arrays must be individually copied over or compared.

Example: How to Copy One Array to a Second Array –

```
int [] array1 = { 1, 2, 3, 4, 5 };
int [] array2 = new int [5];

for (int i=0; i<array1.length; i++){
     array2[i] = array1[i];
}
```

Example: How to Compare Two Arrays –

```
boolean arrEqual = true;  //true until proven false

if (array1.length != array2.length){
     arrEqual = false;
}else{
     for (int i=0; i<array1.length; i++) {
          if ( array1[i] != array2[i]) {
               arrEqual = false;
          }
     }
}

if (arrEqual) {
     System.out.println ("The arrays are the same");
} else {
     System.out.println ("The arrays are different");
}
```

Multi-Dimensional Arrays – a matrix of values, which can be indexed.

2-Dimentional Array:

```
final int ROWS = 3, COLS = 4;
int [][] values = new int [ROWS][COLS];
```

values	Column 0	Column 1	Column 2	Column 3
Row 0	values[0][0]	values[0][1]	values[0][2]	values[0][3]
Row 1	values[1][0]	values[1][1]	values[1][2]	values[1][3]
Row 2	values[2][0]	values[2][1]	values[2][2]	values[2][3]

You can initialize a 2-D array by declaring each value:

```
int [][] values = { { 1, 2, 3, 4},
                    { 5, 6, 7, 8},
                    { 9, 10, 11, 12} };
```

values	Column 0	Column 1	Column 2	Column 3
Row 0	1	2	3	4
Row 1	5	6	7	8
Row 2	9	10	11	12

To process all of the elements of the array you can use a set of nested for-loops.

```
for (int i=0; i < ROWS; i++){
     for (int j=0; j < COLS; j++){
          values[i][j] = 0;
     }
}
```

To access individual elements you can specify its location with the indices.

```
values[1][2] = 5;
```

Using the length function with 2-D arrays:

Length of a row or column:

```
int rowSize = values.length;
int colSize = values[0].length;
```

ArrayLists – are a way of accomplishing a similar task as arrays, except they have a few extra handy functions available. ArrayLists keep track of their own size and expand as you add new elements to the list.

```
import java.util.ArrayList;
```

Creating an ArrayList:

```
ArrayList <Integer> list = new ArrayList <Integer> ();
```

An ArrayList can take any object type, which means that you cannot use int or char, but you can use Integer or Character. The line above creates an empty ArrayList of Integers.

Adding Elements:

```
list.add(1);
list.add(2);
list.add(3);
list.add(4);
list.add(5);
```

Removing Elements – there are two ways of removing an element, by its position (starting counting at 0) or by the object. After an item is removed, the values after it are moved up in position to eliminate the empty space.

```
list.remove(1); //removes the 2nd item in the list
list.remove((Integer) 1); //removes the instance of 1
```

Since the case of Integers and positions is ambiguous, we need to typecast the value as an Integer if we want to remove an element by object, for other object types, this would not have been necessary.

```
list.clear(); //removes all elements from the list
```

Accessing Elements:

```
int x = list.get(1);  //gets the value at position 1
list.set(2, 6); //replaces item at position 2 with a 6
```

Checking the List:

```
list.size(); //returns the number of items in the list
list.contains(4); //returns true if the element exists
```

Printing the List:

```
System.out.println(list);
```

This prints out the contents of the list. It is good for checking the contents, but if you need your list to be displayed formatted, then a loop would be better.