# CSC 204 Lab 12: Intro to ArrayLists

## Goals

After doing this lab, you should be able to:

- declare and trace an ArrayLists
- work with ArrayLists of objects
- use the `for` statement to read in, process, and print out an ArrayList
- complete simple methods involving ArrayList processing

This lab should look similar to Lab 12. As you work through it, spend special attention to the differences between arrays and ArrayLists.

## Lab Preparation

Continue reading the material in Chapter 7.

## Materials Needed

Be sure you have the following on hand during the lab.

- This sheet and your course notebook.
- The files for Lab 12 from `http://theochem.mercer.edu/csc204`

## Working with ArrayLists

1. Let's consider the Java program `BegArrayList` that you copied over and is shown on your reference sheet. Notice how it declares an ArrayList. It is not necessary to give the ArrayList a predetermined size. Draw yourself a picture of ArrayList `a` (right before the print) below. You can draw it either horizontally or vertically.

2. Compile and test your program to see if your array is indeed correct. If it is, edit your program and modify the boolean expression in the first for loop to read `k < 10`. Recompile and execute your program. Explain how the program changes below.

3. Even though you can always add onto the end of an ArrayList, you can't add more than one position past the end. What error do you get when you try to add to position 100, using the statement `a.add(100, new Integer (-99))`?

4. Let's build our own ArrayList now and initialize it with random integers. Create a program named `RandomArray`, and declare an ArrayList `random` of Integers. Now, let's set up a `for` loop that will assign a random integer between 9 and 50 inclusively. So, we'll need to use a `Random` variable inside our loop. Think about the expression that you'll use. Hint : You want to get an integer in the range [0 , 42), and then add on a displacement of 9. Remember to cast your result to an integer.

Once your array has been initialized, print it out nicely spread over five lines. For example,

```
36 29 46 22 33 33 41 13 17 40
 13 47 16 50 9 49 32 17 47 38
28 27 29 36 15 44 31 44 34 27
14 12 23 35 18 44 48 26 32 26
31 22 48 31 26 20 45 27 11 15
```

## Fancier ArrayLists

1. Let's trace through the program `Parallel` now that is shown on your reference sheet. Here we have four different parallel ArrayLists. These are arrays with the same size that store different types of data. Remember an array is homogenous data structure which means we can't mix our data types within it. But, we can easily declare multiple arrays of the same size whose different elements can be accessed "in parallel."

Draw and initialize your four parallel ArrayLists below. Notice how you can add to positons other than the last one. What output to you expect for this program? Check your answer once you're done.

5. We're now ready to try out `CrazyArrayList.java`. This program uses indirect referencing of arrays. That is, it uses an array value as an array index. You'll want to draw yourself a picture of the ArrayList `a` in this one, and trace through this program very slowly. Use arrows to help yourself.

Compile and execute `CrazyArray.java` and check your answers found during the trace.

## Completing Methods with ArrayList Parameters

1. Edit file `MyArrayList.java`. Notice how the main method is already complete. You will want to complete the bodies of four methods involving ArrayListss.

2. First off, let's write a void method named `fillUp` which can be used to fill up an ArrayList of six items with integers that are input from the keyboard. Notice how this function takes an ArrayList `a` as a parameter. You just need to add a for loop here that iterates six times and each time prompts the user to insert an integer. Your code should then get the integer to be added to the ArrayList. Sample input is printed at the end of the lab. (Notice that it is easy to let the user enter as many values as they may want; consider trying this!)

3. Our next method call will be to a void method named `printOut` which is used to print out all of the contents of our ArrayList `a` on the same line. It should be set up very similar to method `fillUp`. Include an output prompt as shown on the back.

4. Your third method will be a value-returning method named `sumUp` which takes as input our ArrayList `a` and returns to the calling function the sum of the items in this ArrayList. Once again, you will need to use a `for` loop. Don't forget to set your local sum variable to zero prior to this loop. Also, since this method is "value-returning" we need to include a return statement at the end of it that returns the integer value representing the sum of our ArrayList. For example, `return sum;`.

5. Your fourth and final method is a value-returning method named `posCount` which takes as input our ArrayList `a` and returns to the calling function the number of integers in the ArrayList which are positive.

You will need to declare a local count variable and remember to set it to zero prior to your loop. Within your loop, use a single-alternative `if` statement to check and see if an ArrayList item is positive. Once again, you'll also need a `return` statement.

6. Compile and test your program using the sample input as well as some sample input of your own.

**Sample Input**

```
Enter value 1:  -5
Enter value 2:  9
Enter value 3:  20
Enter value 4:  30
Enter value 5:  40
Enter value 6:  50
```

**Sample Output**

```
*** PrintOut of ArrayList ***
-5 9 20 30 40 50
************************

ArrayList Sum :  144

Num of Positive Values :  5
```

**REFERENCE SHEET**

```
import java.util.ArrayList;

public class BegArrayList

{

     public static void main(String[] args)

     {

          ArrayList<Integer> a = new ArrayList<Integer> ();


          for (int k = 0;  k < 5;  k++)

               a.add (new Integer (k * k));


          a.set( 2 + 4 / 3 , -1);


          System.out.println(a);

     }

}
```

```
import java.util.ArrayList;
public class CrazyArrayList
{
     public static void main(String[] args)
     {

          ArrayList<Integer> a = new ArrayList<Integer> ();

          a.add(0, new Integer (3));
          a.add(0, new Integer (72));
          a.add(1, new Integer (0));
          a.add(1, new Integer (41));
          a.add(2, new Integer (-23));
          a.add(0, new Integer (64));

          System.out.println (a);

          a.set (4, a.get(2) + a.get( a.get(a.size() - 1) ));
          System.out.println (a);
```

```
   }
}

import java.util.ArrayList;

public class Parallel
{
     public static void main(String[] args)
     {
          ArrayList<Character> a = new ArrayList<Character> ();
          a.add(new Character ('C'));
          a.add(new Character ('$'));
          a.add(new Character ('d'));
          a.add(new Character ('Z'));

          ArrayList<Boolean> b = new ArrayList<Boolean>();
          b.add(false);
          b.add(false);
          b.add(true);
          b.add(false);

          ArrayList<String> c = new ArrayList<String>();
          c.add (0, "baseball");
          c.add (0, "basketball");
          c.add (0, "golf");
          c.add (0, "tennis");

          ArrayList<Fraction> d = new ArrayList<Fraction>();
          d.add (new Fraction(1,2));
          d.add (new Fraction(2,3));
          d.add (new Fraction(-2,5));
          d.add (new Fraction(12,25));

          for (int k = 3;  k >= 0;  k--)
          {
               System.out.print(a.get(k) + " " +
                     b.get(k) + " " + d.get(k) + " ");

               if (c.get(k).compareTo("basketball") < 0)
                    System.out.println("Before basketball");
               else if (c.get(k).compareTo("basketball") > 0)
                    System.out.println("After basketball");
               else
                    System.out.println("Hoops Time!");
          }
     }
}
```