CSC 204 Lab 11: Intro to Arrays

Goals

After doing this lab, you should be able to:

- declare and trace a one-dimensional array of simple data types
- work with arrays of objects
- use the `for` statement to read in, process, and print out an array
- complete simple methods involving array processing

Lab Preparation

Make a copy of this document and name it "Lab11 - YOURNAME". Edit your copy of this document to answer questions below. Read through this lab. Have Chapter 7 from our textbook handy for reference. Create a new Java Project in Eclipse named Lab11. Copy all of the Java files from our courses Lab11 folder on Blackhawk into the src folder of this project on Eclipse.

Working with One-Dimensional Arrays

1. Let's consider the Java program `FirstArray` that you copied over and is shown at the bottom of this doc. Notice how it declares an array of size 5 using `new`. It will initialize each of these five slots to zero automatically for you. List the contents of array `a` as is appears before the loops run (i.e. value, value, …)

   [0x0], [1x1], [2x2], [-1], [4x4]

2. Now, let's trace through this program and determine how it will fill the slots in your array above. Notice how we can reference the public instance variable length of an array. This variable contains an array's declared size. List the contents of array `a` after the code runs.

   0, 1, 4, -1, 16

3. Compile and run `FirstArray` to see if your array is indeed correct. Now, edit your program and modify the boolean expression in the first for loop to read `k < a.length + 1`. Recompile and execute your program. Give the first line of this error message below**.**

   Thread [main] (Suspended (exception ArrayIndexOutofBoundsException))

Going out of bounds on an array is a very common run-time error in a program. You want to be very careful to not go past the defined size of your array!

4. Let’s build our own one-dimensional array now and fill it with random integers. Create a program named `RandomArray`, and declare an array named `random` of 50 integers. Now, let’s setup a for loop that will assign a random integer between 9 and 50 inclusively.

   To create a (pseudo) random number, use the `Random` class. Create a variable of type `Random`. Suppose we name it `rand`. To get a random number from 0 to 100 (not including 100), you'll use the function `rand.nextInt(100)`.

   Think about the expression that you’ll use to get a value between 9 and 50. Hint: You want to get an integer in the range [0 , 42), and then add on a displacement of 9.

Once your array has been initialized, print it out nicely spread over five lines (think printf). 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 One Dimensional Arrays

5. We’re now ready to try out `CrazyArray.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 array `a` in this one, and trace through this program very slowly. Use arrows to help yourself.

```
a[a[i]] = a[a[i]] - 1;
  int i = 2;
a[a[i]] = a[a[2]]-1;
a[a[i]] = a[0]-1;
a[a[i]] = 3-1
   a[j] = 2;

System.out.println(a[2+a[j]]);
System.out.println(a[2+2]);
System.out.println(a[4]);
System.out.println("-23");
```

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

Completing Methods with Array Parameters

7. Edit file `MyArray.java`. Notice how the main method is already complete. You will want to complete the bodies of four methods involving arrays.

8. First, let's write a void method named `fillUp` which can be used to fill up an array with integers that are input from the keyboard. Notice how this function takes an array `a` as a parameter. You just need to add a `for` loop here that iterates once for each slot in the array `a` and each time prompts the user to insert an integer. Remember an array’s size can be found with `a.length`. Your code should then read in their integer and add it to the array. A sample input is printed below on this sheet.

9. Our next method call will be to a void method named `printOut` which is used to print out all of the contents of our array `a` on the same line. It could be very similar to method `fillUp` or you could use an enhanced for loop. Include an output label as shown below.

10. Your third method will be a value-returning method named `sumUp` which takes as input our array `a` and returns to the calling function the sum of the items in this array. 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 array. For example, `return result;` .

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

12. 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 array item is positive. Once again, you'll also need a `return` statement.

13. 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 Array ***
-5 9 20 30 40 50
*************************
Array Sum :  144

Number of Positive Values :  5
```

Deliverables

When complete, copy your src folder for this Eclipse project into your shared Google folder, and then change the name of copied src folder to "Lab11". Also, put your copy of this document into the shared Lab11 folder.