IST 256
Lab
Week 7 – Tuesday, February 25, 2014

1. Working with arrays in an application

In this program, we will use a Java application to create an array and write a program to compute the average of the array. To put values in the array, we will use the Java random number generator. This is a program that if you give it an upper bound, say 100, it can generate a sequence of random numbers that are all between 0 and 99.

A. Start writing the program

Create a new Java application and call it something like ArrayAverage.

We can place other methods in this program, such as this method which will create an array of random numbers. First **place this import statement between the package statement and the public class Main line**, which will make the Java Random class available to the program.

```
        import java.util.Random;
```

Then place this method definition **between the public class Main line and the public static main method header**.

```
/*
 * This method creates an array of random integers.
 *    The random numbers are created by a Java random number generator:
 * Given an upper bound 100, the numbers are uniformly distributed between 0 and 99.
 * Parameter:  sizearray specifies the number of elements of the array
 * Result:     returns an array of random integers
 */
private static int [] getRandomNumbers (int sizearray)
{
   // create the array of the specified size
   int [] numberarray = new int[sizearray];
   // create the Java random number generator
   Random randomGenerator = new Random();

   // for each element of the array, get the next random number
   for (int index = 0; index < sizearray; index++)
   {
      numberarray[index] = randomGenerator.nextInt(100);
   }
   return numberarray;
}
```

There are several interesting features about this method, including the fact that it can return an entire array as a value by giving the return type of integer array, "int [ ]".

Now for the body of the main method, place the following code, which uses the getRandomNumbers method to create an array of values and display them.

```
// create an array of length 10 that is initialized with random integers
int [ ] numbers = getRandomNumbers(10);

// print these numbers
for (int i = 0; i < numbers.length; i++)
{
   System.out.println("Num " + i + ":   " + numbers[i]);
}
```

Test this program several times and observe the numbers that are generated. Write one set of numbers here (from **one run** of the program).

Array Values:										

B. Add code to compute the average of the array

Use the solution developed in class to add code to compute the average of all the values in the array. Add this code to the main method after your other code. (Note that the average is computed by adding up all the values in the array, using a for loop, and then dividing the total by the number in the array to get the average. The average should be of type double.)

Test this program at least three times and write three different average numbers here (from **three runs** of the program).

Three runs of the program:	
First average:	
Second average:	
Third average:	

C. Add code to compute the maximum value of the array

Use the solution developed in class to add code to compute the maximum of all the values in the array. Add this code to the main method after your other code. (Note that the maximum is computed by starting the index of the maximum so far to be the first index in the array, and then using a for loop to compare with the remaining values of the array and setting the maximum so far to be the index of any larger value. We complete the program by getting the maximum value so far to be the value of the index.)

Test this program at least three times and write three different maximum numbers here (from **three runs** of the program).

Three runs of the program:	
First maximum:	
Second maximum:	
Third maximum:	

2. Debugging Techniques

If your program is running but is getting errors or not getting the results it should, it can be useful to see the values of some of the variables while the program is running.

A. Printing values

One way to see the values of some of the variables while the program is running is to add print statements. These will be removed or "commented off" in the final version of the program.

Example 1: We can do this in the average program by printing out the total so far during the loop.

Example 2: In a GUI program, we can also do this and note that the result of the print statement shows in the output pane. This is o.k. because we are not showing these values to the user. Look at the ComputeSavings program.

B. Use NetBeans debugging system

Like most IDE's, NetBeans has a debugging system that allows you to stop your program in mid-run, or step through your program line by line, and observe the values of the variables.

http://www.cs.uga.edu/~shoulami/sp2009/cs1301/tutorial/NetBeansDebuggerTutorial/NetBeansDebuggerTutorial.htm

We will

- set a breakpoint to stop the program at a location during the loop
  - click on the line where you want to stop
- run the program in debugging mode
  - under Debug menu or use Debug Main Project small triangle icon
  - project name must be in bold, showing main project
- when the program stops, observe variables in the variables window
  - select Windows -> Debugging ->Variables
- continue debugging, step over and stop debugging session
  - use the icons at the top of the main window