Lab #1: Java review and the main() method

General

For this lab, you will review the basics of Java syntax; defining a class, defining fields and methods, using collection classes (ArrayList), writing method implementations, reviewing control structures, and writing JavaDoc formatted comments. ***An additional requirement for this lab it that it is to be updated using simple text editor (notepad, JEdit, VI, Emacs, or whatever your favorite poison is). You will then compile and execute the program directly from the command line instead of using BlueJ.***

At the top of the every document that you create or modify (word processing files or source files) include a class JAVADOC comment with your identifying information:

```
/**
 *  Description of the class or document.
 *
 * @author   YOUR NAME
 * @version  CS162 Lab 1, mm/dd/yyyy    (replace with the date that you last edited the lab)
 */
```

Submit in this lab via Moodle using the “Lab #” link. Your assignment must be uploaded by the assigned due date and time. Moodle will automatically close the link at that time. It is strongly highly recommend you do not wait until the very last minute to submit your work.

Concepts

This is mostly a review lab; but you will also get your first experience with compiling and running Java console applications from the command line.

Background

You may want to review material from Chapters 1-7 of the Objects first text; there are also several review tutorials listed on the class web site. Also review your code from CS-161 Lab #7 as this code will be very similar.

Assignment Instructions

The general requirements for this lab are to create a new project from scratch that contains a single class (name the class anything that seems appropriate to you). This class will have an **ArrayList of integers** as a field and an **integer constant** that defines the **`SIZE`** of the ArrayList. The constructor must create an instance of an ArrayList object and assign it to the private field. Then you will define several additional methods described below (you must use the method signatures as I have given them). Your methods should be written generically enough so that you can change the **`SIZE`** constant and everything should continue to work correctly with the new **`SIZE`** value.

This lab is a re-engineering project on your Lab#7 from CS161 (or simply start it from scratch again for review). There are a couple of additional requirements:

1. You MUST use an **ArrayList, not an Array** for this lab.
2. You must show evidence that you can compile this from the command line and execute from the command line.
3. A loop in one of the methods MUST use a “for-each” loop to traverse the ArrayList.
4. A loop in one of the methods MUST use an Iterator approach to traverse the ArrayList.
5. ArrayLists cannot contain primitive data types; so you will have to handle “boxing” and “unboxing” of the Integer wrapper class. Starting with Java 1.6 (Java 6) there is automatic support for this (boxing and unboxing of “int” primitives and “Integer” objects).

Define 2 constants:
`public static final int SIZE = 10;`
`public static final int RANGE = 100000;`

Then you will define several additional methods that do the following:

1) **`public void fillArray()`**
fills each element of the array with a random integer with a max value of your RANGE constant.

2) **`public int getMax()`**
this method returns the largest integer in the ArrayList.

3) **`public int getMin()`**
this method returns the smallest integer in the ArrayList.

4) **`public int getSum()`**
this method returns the sum of all integers in the ArrayList.

5) **`public int getAvg()`**
this method returns the average value of the integers in the ArrayList.

6) **`public void printContents()`**
prints each element value in the ArrayList from the item in index 0 through the item in the max index location.

7) **`public void testMethods()`**
calls and prints the resulting values from the above methods to verify their correctness. The code for this method must be exactly as follows:

```
/**
 * Tests the methods to verify their correctness; numeric
 * values will change due to the generation of random
 * values, but the math is easy enough
 * to visually verify if things are working
 */
public void testMethods()
{
    fillArray();
    printContents();
    System.out.println("SIZE value: " + SIZE);
    System.out.println("Max value:  " + getMax());
    System.out.println("Min value:  " + getMin());
    System.out.println("Sum:        " + getSum());
    System.out.println("Average:    " + getAvg());
}
```

8) **`public static void main(String[] args):`** Creates an instant of the object, and calls the “testMethods( ) method on the object created.

9) **`public void sort()`** (OPTIONAL Challenge Exercise)
sorts the elements in the ArrayList in numeric order using either bubble sort or selection sort. Search for information on these algorithms on the Internet.

Challenge exercise:
Search the internet for selection or bubble sort Java code. Integrate the code into your program or write a sort method from scratch.

BE SURE that you have set your SIZE constant to 10, then when you execute testMethods() or main( ) your output should match the following, except for the numbers:

```
[0] 95934
[1] 45366
[2] 60428
[3] 81464
[4] 24412
[5] 85560
[6] 14309
[7] 76920
[8] 33567
[9] 61862
SIZE value: 10
Max value:  95934
Min value:  14309
Sum:        579822
Average:    57982
```

You are to compile and execute this lab from the command line. Create a document for this lab and do a **window capture of the command prompt showing a successful compile of your lab and a second window capture showing the execution of your program**. Recall that you can get a capture of the active window into the system clipboard by pressing alt+printscreen. **Turn in the document and your source code file.**