CSCI 1301 – Programming Principles I
Georgia Southern University
Department of Computer Science
Fall 2024

Lab 8
Point Value: 20 points
Due: Friday October 25, 2024, end of lab

Objectives
1. Create and use methods to write working programs.
2. Overload methods to work with different data types.
3. Use the debugger to effectively diagnose and fix errors in code.
4. Archive Java source code files and other documents into a zip file.

Description
Complete the following steps as described and upload requested materials to Folio in the appropriate dropbox in a single zip file. Submit Java source code files to Gradescope.

- Students should work in groups of two for this lab. When submitting your zip file, fill out the comment header and indicate both group member names.

- Your submitted zip file should be named Lab#LastName1LastName2.zip. For instance, Lab8JonesSmith. Any requested screenshots and/or answers to questions should be included in a separate Word document.

- All programs should have a comment header. That comment header should include information similar to:
```
/**
 * File: PAssign0.java
 * Class: CSCI 1301
 * Author: Christopher Williams
 * Created on: Jun 6, 2016
 * Last Modified: Aug 16, 2018
 * Description:  Display three messages to the console
 */
```

- Before writing any code, for each problem, discuss with your partner how you plan to approach solving the given problem. Once you both agree on a way to attempt the problem, only then should code be written.

- Pay attention to names and any other requested material. Failure to include material will result in the loss of points as described below.

Lab Problems
For all problems, use inline comments to describe major actions. Ask for assistance if needed.

1. Value-Returning Methods/Invoking Methods – Write a program named Lab08Prob01.java that contains a value-returning method that determines the average of three double values.

   IMPORTANT: Copy your average method from last lab that averaged three ints into this program along with your main method from that problem.

   In this program, overload the average method so that the method can take and average three double values as parameters.

   Your main method should be updated to include a call to this overloaded method. Make sure that it is called with double values and use the debugger to verify the correct method is being executed.

   Do not remove or delete any code from Problem 2. Numbers used should be constrained to 2 decimal places, the average should stay at 5 decimal places.

   Expected Output (for shown values):
   ```
   The average of 2, 3, and 6 is 3.66667
   The average of 2.74, 3.45, and 6.21 is 4.13333
   ```

   After passing the Gradescope tests, show a working version of this program to the instructor or TA before moving on.

   After instructor check-off, BOTH students should submit a final, identical version to Gradescope. This version is used for your final grade.

2. Overloading Methods/Invoking Methods – Copy and paste your code from Problem 1 into a new file named Lab08Prob02.java. In this program, overload both of the existing average methods so that they can receive and average four values. Do not remove or delete any code from Problem 1.

   Numbers used should be constrained to 2 decimal places, and the average should stay at 5 decimal places.

   This new method should **NOT** call the method used/created in Problem 1. Remember that in class, our max/min methods could reuse “smaller” versions of themselves to get accurate results without rewriting code. **In a comment in your submitted code, explain why these new methods should NOT call existing methods.**

   Expected Output (for shown values):

```
The average of 2, 3, and 6 is 3.66667
The average of 2.74, 3.45, and 6.21 is 4.13333
The average of 2, 3, 6, and 11 is 5.50000
The average of 2.74, 3.45, 6.21, and 11.91 is 6.07750
```

After passing the Gradescope tests, show a working version of this program to the instructor or TA before moving on.

After instructor check-off, BOTH students should submit a final, identical version to Gradescope. This version is used for your final grade.

3. Using Overloaded Methods – Copy and paste your code from Problem 2 into a new file named Lab08Prob03.java. In this program, you will provide one version of a standard deviation method that uses one of the existing average methods to calculate standard deviation. Name your method stdDevSample(), it should take 4 double parameters, and return a double value.

   Update your main method to call this stdDevSample() method with four double values and then print the returned result to the console.

   To calculate standard deviation for a sample:

$$SD_{sample} = \sqrt{\frac{\sum |x-\mu|^2}{n-1}}$$

   Where μ is the mean (average) of the set, the summation is the squared distance of every data point from the mean, and n is the number of items in the sample[1].

   NOTE: There will be no solution you can use with what we know at the moment to make this process easier. These calculations, minus the average, will mostly have to be done manually, one at a time. We will learn how to solve this problem in a general manner in the next chapter.

   Do not remove or delete any code from Problem 2 except in the main method. Numbers used should be constrained to 2 decimal places, and the standard deviation should be constrained to 5 decimal places.

   Expected Output (for shown values):

```
The standard deviation of 2.74, 3.45, 6.21, and 11.91 is 4.16646
```

[1] See https://www.khanacademy.org/math/statistics-probability/summarizing-quantitative-data/variance-standard-deviation-population/a/calculating-standard-deviation-step-by-step for a detailed description of this process. Do NOT use the formula given there as it calculates standard deviation of a population, not a sample.

After passing the Gradescope tests, show a working version of this program to the instructor or TA before moving on.

After instructor or TA check-off, BOTH students should submit a final, identical version to Gradescope. This version is used for your final grade.

4. Demonstrate all working programs to the instructor or TA to receive credit for each problem. Turn in each problem inside the zip file detailed below.

5. Name the zip file Lab8LastName1LastName2.zip (e.g. Lab8JonesSmith) and submit that zip file to Folio. Both students must submit the exact same zip file for either to receive a grade.

Grade Breakdown

Demonstrate running application to instructor or TA:	
Problem 1	6 points
Problem 2	6 points
Problem 3	6 points
Zipped source files (in Folio):	2 points
Total Possible:	20 points

Last modified: October 21, 2024