Introduction to Computer Programming (CS 1323)
Project 3

September 29, 2014

Introduction

In project 2, we computed the mean and standard deviation of the values from two sensors from our simulated weather station. While these two statistics (mean and standard deviation) can be useful for summarizing a set of values from a sensor (also called "samples"), they are not considered "robust" statistics because they can be very sensitive to "outliers." For example, imagine a set of values from the temperature sensor that are all in the vicinity of 30^o. The mean of these samples would be approximately 30^o. However, if we take one of these values and replace it with an erroneously large value (e.g., 1000^o), the mean would change dramatically.

The median and "inter-quartile range" statistics are similar to median and standard deviation, but are robust to outliers. The median of a set of values $(v_1, v_2, v_3, ..., v_N)$ is computed as follows:

1. Sort the values in ascending order
2. The median is the middle value in the list. For our purposes, if there are N values, the middle is the value at the $(N/2)$ location of the sorted list (rounded down), counting from zero.

For example, the median of the values $(5, 15, 16, 8, 18, 9)$ is 15.

The inter-quartile range (IQR) is computed as follows:

1. Sort the values in ascending order
2. The IQR is the difference between the value at $\frac{3}{4}N$ and the value at $\frac{1}{4}N$. Again, these positions are rounded down and start counting from zero.

For example, the IQR of the values $(5, 15, 16, 8, 18, 9)$ is $16 - 8 = 8$.

As in project 2, your program will prompt a user to input a sequence of valid <temperature, wind velocity> pairs and compute the median of the valid temperature values and IQR of the valid wind velocity values.

When the user provides a non-numeric input for temperature, then your program will report these two statistics and halt. When the user provides a non-numeric **or** out-of-range value for wind velocity, then the user is re-prompted for correct input.

Objectives

By the end of this project, you will be able to:

1. reuse code that you have written for other purposes,
2. create and manipulate **ArrayLists**,
3. use the **Collections** class to manipulate an ArrayList, and
4. compute simple statistics over lists of values.

Project Requirements

1. Copy your **temperature sensor input** and **wind velocity input** methods from project 2 (it is assumed that you have previously implemented these methods according to the project 2 specifications).
2. Write a **computeMedian()** method that takes as input an ArrayList of Doubles and returns a double that is the median value of the list. Do not assume that the values in the ArrayList are ordered before the call to this method.

3. Write a **computeIQR()** method that takes as input an ArrayList of Doubles and returns a double that is the IQR of the list. Do not assume that the values in the ArrayList are ordered before the call to this method.

4. Modify your **main()** method that calls your two input methods repeatedly until NaN is returned by the temperature sensor method. As valid values are received, this method will place them into ArrayLists for temperatures and wind velocities.

   Valid temperatures are -20 and above.

   Valid wind velocities are between -100 and 90.

   When NaN is returned, your main method reports the **median** of the entered temperature values and the **IQR** of the wind velocities.

   If no tuples are entered before the NaN is returned, then your program must explicitly state that there are no values from which to compute these statistics.

Examples

Here are some example inputs and corresponding outputs from a properly executing program (note that user interaction output is not shown). You should carefully consider the cases with which you test your program. In particular, think about the key "boundary cases" (the cases that cause your code to do one thing or another, based on very small differences in input).

Example 1

Input:

```
10
30
20
25
30
35
D
```

Output:

```
20.0 and 10.0
```

Example 2

Input:

```
10
-100
-70
0
200
300
45
5
-50
D
```

Output:

```
5.0 and 145.0
```

Example 3

Input:

```
70
-10
55
D
D
a
10
60
15
10
-10
-70
0
200
300
45
5
-50
D
```

Output:

```
55.0 and 25.0
```

Project Documentation

Follow the same documentation procedures as we used in project 2. In particular, you must include:

- Java source file documentation (top of file)
- Method-level documentation
- Inline documentation

For full credit, you must execute Javadoc on your source file and include the results (in the *doc* directory) in your zip file.

Hints

ArrayList.add() will add a new value to a list.

ArrayList.size() will return the number of values in the list.

Collections.sort(list) will sort the elements in a list in ascending order. Note that this method is declared as static (see the Java API for more details).

Project Submission

This project must be submitted no later than 2:00pm on Tuesday, October 7th to the project 3 D2L dropbox. The file must be called *Project3.zip* and be generated in the same way as for Project 2.

Code Review

For the office hour sessions surrounding the project deadline, we will put together a "Doodle Poll" through which you can sign up for a 5-minute code review appointment. During this review, we will execute test examples and review the code with you. If you fail to show up for your reserved time, then you will forfeit your appointment and you must attend at a later time that is not already reserved by someone else.

If either the instructor or TA are free during office hours (or another mutually-agreed upon time), then we are happy to do code reviews.

We will only perform reviews on code that has already been submitted to the dropbox on D2L. Please prepare ahead of time for your code review by performing this submission step.

Code reviews must be completed no later than Monday, October 13th. If you fail to do a code review, then you will receive a score of zero for the project.

Anyone receiving a 95 or greater on Project 2 may opt to do an "offline" code review (which does not require your presence). Send the instructor email once you have submitted your project in order to schedule an offline review.

Rubric

The project will be graded out of 100 points. The distribution is as follows:

Implementation: 35 points

Program formatting: 15 points

(15) The program is properly formatted (including indentation, curly brace and semicolon locations).

(8) There is one problem with program formatting.

(0) The program is not properly formatted.

Data types and method calls: 10 points

(10) The program is using proper data types and method calls.

(5) There is one error in data type or method call selection.

(0) There are multiple errors in data type and method call selection.

Required Methods: 10 points

(10) All of the required methods are implemented.

(0) The required methods are not implemented.

Proper Execution: 30 points

Output: 15 points

(15) The program passes all tests.

(10) The program fails one test.

(5) The program fails two tests.

(0) The program fails three or more tests.

Execution: 15 points

(15) The program executes with no errors.

(8) The program executes, but there is one minor error.

(0) The program does not execute.

Documentation and Submission: 35 points

Project Documentation: 5 points

(5) The java file contains all of the required documentation elements at the top of the file.

(3) The java file is missing one of the required documentation elements.

(2) The java file is missing two of the required documentation elements.

(0) The java file is missing more than two of the required documentation elements.

Method-Level Documentation: 10 points

(10) Every method contains all of the required documentation elements ahead of the method prototype, and the Javadocs have been generated and included in the submission.

(7) The method documentation is missing one of the required documentation elements.

(3) The method documentation is missing two of the required documentation elements, or the Javadocs have not been submitted.

(0) The method documentation is missing more than two of the required documentation elements.

Inline: 10 points

(10) Every method contains appropriate inline documentation.

(7) There is one missing or incorrect line of inline documentation.

(3) There are two missing or incorrect lines of inline documentation.

(0) There are more than two missing or incorrect lines of inline documentation.

Submission: 10 points

(10) The correct zip file name is used.

(0) An incorrect zip file name is used.

Bonus: 6 points For each unique and meaningful error in this project description that is reported to the instructor, a student will receive an extra 2 points.

References

- Computing median: http://en.wikipedia.org/wiki/Median
- Computing Inter-Quartile Range: http://en.wikipedia.org/wiki/Interquartile_range