Introduction to Computer Programming (CS 1323) Project 2

September 24, 2014

Introduction

Each of the weather stations in the Oklahoma Mesonet collection a range of information, including wind velocity. In this project, we will simulate the collection of both the air temperature and the East/West wind velocity from a single sensor. The wind velocity in our simulation is only considered to be valid in the range of -80 to 100 $meters/sec$. The temperature sensor will have the same constraints as in Project 1.

Your program will prompt a user to input a sequence of valid <temperature, wind velocity> pairs (also called *tuples*) and compute the standard deviation of the wind velocities.

When the user provides a non-numeric input for temperature, then your program will report the mean of the temperatures and the standard deviation of the wind velocities 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. write a method,
2. use the Math class to perform computations,
3. write java-qualified documentation.

Project Requirements

1. Copy your **temperature sensor input method** from project 1.
2. Write a new **wind velocity sensor input method** that takes as parameters a Scanner and both minimum and maximum allowable sensor values (a double). This method returns a double.

   This method must read a double as input from the Scanner. If this value is in the valid range (between minimum and maximum, inclusive), then this value is returned by the method.

   If the value is outside the range or if the input is not a double, then the program will re-prompt the user for a valid value. This is repeated until the range constraint is met.
3. Write a **main method** that calls your two methods repeatedly until NaN is returned by the temperature sensor method. When NaN is returned, your main method reports the **mean** of the entered temperature values and the **standard deviation** of the wind velocities. These computation are to be done in this method. You *may not* use arrays for this.

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 5.0
```

Example 2

Input:

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

Output:

```
5.0 and 61.44
```

Example 3

Input:

```
70
-10
55
D
D
a
10
60
15
D
```

Output:

```
61.67 and 13.23
```

Project Documentation

Follow the same documentation procedures as we used in project 1. 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

As in project 1, any time you use **Scanner.nextDouble()**, you must first check to make sure that a double is available using **Scanner.hasNextDouble()**.

The standard deviation of a set of numbers $v_1, v_2, v_3, ..., v_N$ is defined as:

$$std = \sqrt{\frac{\sum_{i=1}^{N} (v_i - \bar{v})^2}{N-1}},$$

where

$$\bar{v} = \frac{\sum_{i=1}^{N} v_i}{N}.$$

With this definition and a bit of algebra, we can show that standard deviation is also:

$$std = \sqrt{\frac{1}{N-1}\left[\left(\sum_{i=1}^{N} v_i^2\right) - \frac{1}{N}\left(\sum_{i=1}^{N} v_i\right)^2\right]}.$$

What this means is that as your **main()** method is gathering wind velocity values, it must only keep track of three things:

1. the number of valid wind velocity measures,
2. the sum of the valid wind velocity measures, and
3. the sum of the squares of valid wind velocity measures.

Once your method is ready to report the standard deviation, it can take these three values and perform the final computation.

Math.sqrt(val) returns the square root of val.

After your wind velocity input method detects a non-double, it can “eat” the offending non-double by using **Scanner.next()**. This will remove all characters from the input up to the next white spaces (spaces, new lines, carriage returns and tabs).

Project Submission

This project must be submitted no later than 2:00pm on Tuesday, September 30th to the project 2 D2L dropbox. The file must be called *Project2.zip* and be generated in the same way as for Project 1.

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 10-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 6th. If you fail to do a code review, then you will receive a score of zero for the project.

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 variance: http://en.wikipedia.org/wiki/Algorithms_for_calculating_variance