Programming Assignment 9

`ScoreCalculator`

Overview

As you know, *perspecTV* is a company that designs, creates and markets products that provide a new perspective on television. Their products make television both more interactive and more informative.

They are in the process of developing a suite of products called forScore for judged competitions of various kinds (e.g., sporting events like diving and gymnastics, singing contests, dance competitions). These products will be used by the organizers of the events, the broadcasters of the events, and the viewers/audience. They have hired you to construct several interfaces/classes that will, ultimately, become part of these products.

They were happy with your work on `Shortening` and have now contracted with you to create a class called `ScoreCalculator`.

Again, this project may lead to additional work in the future.

Specifications

The `ScoreCalculator` class must comply with the following specifications:

1 It must have a method with the following signature:

```
public static double rawScore(String[] data)
```

This method must return the raw score calculated from the `String` representations of the individual judge's scores. Specifically:

1.1 If there are fewer than five judges, the raw score is just the mean (which is sometimes called the average) of all of the judge's scores. For example, if the method is passed the array:

```
{"1.4", "3.9", "2.2", "1.4"}
```

the raw score would be:

(1.4 + 3.9 + 2.2 + 1.4) / 4 = 8.9 / 4 = 2.225

1.2 Otherwise, the raw score is the mean after "throwing out" the high and low scores. For example, if the method is passed the array:

```
{"1.0", "2.0", "5.0", "7.0", "6.0"}
```

the raw score would be (approximately):

$$(2.0 + 5.0 + 6.0) / 3 = 13.0 / 3 = 4.3333$$

Note that exactly two scores are "thrown out", whether or not there are ties.

1.3 This method is passed an array of `String` representations of `double` values. It can and must assume that all of the `String` objects correspond to valid `double` values.

2 It must have a method with the following signature:

```
public static double totalScore(String[] data,
                                double difficulty)
```

2.1 This method must return the raw score (calculated from the individual judge's scores) multiplied by the degree of difficulty. For example, if the method is passed the array:

```
{"1.0", "2.0", "5.0", "7.0", "6.0"}
```

and the degree of difficulty 2.6 it must return (approximately):

$$4.3333 \cdot 2.6 = 11.2667$$

2.2 This method is passed an array of `String` representations of `double` values. It can and must assume that all of the `String` objects correspond to valid `double` values.

The `ScoreCalculator` class may have other private static methods as well.

An Important Change

You **must not** use the `Text` class for this assignment (or, indeed, any longer in this course). The `Text` class was created just for this course to make things more convenient for the early assignments. You must, instead, use "standard" Java classes.

For this assignment, you might find the `Double.parseDouble(String s)` and `Integer.parseInt(String s)` methods useful.

Hints

As we have discussed, there are many ways to solve most problems, and problem-solving/design involves formulating/identifying multiple solutions and choosing the best solution. In this case, there are many different ways to implement the `rawScore()` method, and the most obvious approach (i.e.,

the approach that most people think about first) is probably not the best. Indeed, it is very likely that the approach you think of first will seem easy at the beginning but will be very difficult to complete. So, before you start "typing", try to come up with multiple solutions and then choose the one you think is best.

In this regard, you should review the lecture/chapter on accumulators because there is an example in that lecture/chapter that is particularly relevant.

Recommended Process

1. Read and understand the entire assignment.
2. Create a directory/folder for this assignment.
3. Design and implement the `rawScore()` method.
4. Test (and debug, if necessary) the `rawScore()` method.
5. Attempt to refactor (i.e., improve the structure/elegance of the code in, without changing the functionality of the code in) the `rawScore()` method. (Hint: Think about adding a private method to simplify the code.)
6. Design and implement the `totalScore()` method.
7. Test (and debug, if necessary) the `totalScore()` method.
8. Submit your implementation of `ScoreCalculator.java` in a file named `pa9.zip` using Autolab. Do not include any other files in the `.zip` file.

Grading

Your code will first be graded by Autolab and then by the Professor. The grade you receive from Autolab is the maximum grade that you can receive on the assignment.

Autolab Grading

Your code must compile (in Autolab, this will be indicated in the section on “Does your code compile?”), and all class names and method signatures comply with the specifications (in Autolab, this will be indicated in the section on “Do your class names, method signatures, etc. comply with the specifications?”) for you to receive any points on this assignment.

Autolab will then grade your submission as follows:

Conformance to the Course Style Guide:	**20 points** (All or Nothing)
Correctness:	**80 points** (Partial Credit Possible)
`rawScore()`:	60 points
`totalScore()`:	20 points

Manual Grading

After the due date, the Professor may manually review your code. At this time, points may be deducted for inelegant code, inappropriate variable names, bad comments, etc.