CS 145: Preassignment 3

Da-Ta!

1 Description

The goal of this preassignment is to get your hands on arrays. You'll be writing methods that we'll eventually use in class to generate music.

2 Requirements

To receive full credit for this assignment, you must satisfy the following requirements:

- Write all code in a class named `DataUtilities` in a package named `pre3`. Case and spelling matter.
- Compose a handful of `public` and `static` methods, described below. None of them alter their arguments. You may assume that all arrays passed as arguments contain at least one element.
  - A method `getMinimum` that takes in a `double` array argument and returns as a `double` the minimum value in the array.
  - A method `getMaximum` that takes in a `double` array argument and returns as a `double` the maximum value in the array.
  - A method `getSpan` that takes in a `double` array argument and returns as a `double` the array's span of values—i.e., the difference between the maximum and the minimum.
  - A method `normalize` that takes in a `double` array argument and returns the normalized array as a new `double` array. Normalization is the process of shifting and scaling values such that their range [min, max] transforms to [0, 1]. All values are shifted and scaled in the same way so that the data keeps its relative shape. Think about how this might be done before searching for a solution. Work through a small set of numbers, trying to turn its minimum into 0 and its maximum into 1.
  - A method `join` that takes in two `double` array arguments and returns a new `double` array in which the two arrays are joined. In this new array, all elements of the first array are followed by the elements of the second.
  - A method `add` that takes in two `double` array arguments and returns a new `double` array in which the two arrays are summed element-wise. In this new array, element i is the sum of elements i in the two argument arrays. You may assume both array arguments have identical length.

You may share testing code on the discussion board, but please do not share code from your `DataUtilities` class.

3 Submission

This is a preassignment, and preassignments are graded by SpecCheckers, not your instructor or TAs. The good news is you get to tweak your code until you have a perfect score. The bad news is that there's a little setup involved—and you must leave yourself enough time to work out kinks. Deadlines are not extended for botched configurations. Please follow these directions closely to test and submit your work:

1. Add JUnit to the Build Path if you haven't already done so. See preassignment 1 for details.
2. Download the SpecChecker JAR file.
3. Drag the JAR file onto your `pre3` package in Eclipse. Right-click on it, and select Build Path → Add to Build Path. It will migrate to the Referenced Libraries node in your project.
4. Run the JAR file by selecting it and hitting the Play button in the Eclipse toolbar. Run it as a Java application if prompted.
5. Address any errors. **The score you see will be the score you are assigned in the gradebook.**
6. Once you are satisfied with your score, choose a directory in which `pre3.zip` will be saved. This file contains a compressed version of `DataUtilities.java`.
7. Drop your submission into `W:\c s\CJohnson\cs145\<YOUR-USERNAME>`. You may overwrite this file as often as you like before the deadline. Just make sure the directory contains only one *.zip file with lowercase pre3 in its name. Double-click on the file to make sure it's readable and uncorrupted.