ITCS 209 – Object Oriented Programming

Project 0: Kalculator (We're just getting started)

Update Logs:

```
2019-01-27     Initial Version
```

Due: Monday Feb 18, 2019 11:55PM

Learning Objectives:

On the course of implementing this programming project, you will learn some of the basic concepts of the object oriented programming paradigm, and how to apply them to practical, real world programming applications. Specifically, upon accomplishing this project, we expect you to be able to:

1. Be familiar with Java syntax and its coding/compilation environment.
2. Be familiar with project submission protocol.
3. Understand class construction, different types of methods, primitive variables, and loops, through implementation.
4. Be a good problem solver, think algorithmically, and write a program to solve computational problems.
5. Enjoy coding with Java.
6. Brag on your resume that you are a "good (Java) programmer."

Introduction:

Implementing an automated routine to do things for you is a fundamental application of programming. The ability to speak programming languages fluently allows you to communicate with computers in order to develop a wide range of useful, intelligent applications. In this project, you will implement a simple calculator, ***Kalculator***, which supports the following calculation functions:

- Kalculator(): The constructor to initialize a Kalculator object.
- addNumber(double number): Add a number to the list of data.
- deleteFirstNumber(): Remove the least recently added number.
- deleteLastNumber(): Remove the most recently added number.
- getSum(): Return the sum of all the numbers in the list.
- getAvg(): Return the average of all the numbers in the list.
- getStd(): Return the sample standard deviation (std) of all the numbers in the list. The formula for sample std is as follows:

  $$s = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \overline{x})^2}{N-1}}$$

  where $\{x_1, x_2, \ldots, x_N\}$ are the observed values of the sample items, $\overline{x}$ is the mean value of these observations, and *N* is the number of observations in the sample.

- getMax(): Return the maximum of all the numbers in the list.

- getMin(): Return the minimum of all the numbers in the list.
- getMaxK(int k): Return an array of maximum k numbers of all the numbers in the list.
- getMinK(int k): Return an array of minimum k numbers of all the numbers in the list.
- printData(): Print all the numbers in the list.

After initialization, a Kalculator maintains a list of numbers (you can use either an array or an ArrayList to implement this), each of which can be inserted via addNumber(double number) only. That is, the user programs must not be able to access any Kalculator's class attributes directly. They must do so via the above public methods. You can assume that a Kalculator can hold not more than 10,000 numbers.

Instructions:

In the package, you will find the following Java files:

```
Kalculator.java
KalculatorStudentTester.java
```

Your tasks are to implement the missing code in Kalculator.java, as specified (i.e. see `//INSERT YOUR CODE HERE`).

`KalculatorStudentTester.java` is for you to test your code. See Test Cases section. Do not modify `KalculatorStudentTester.java.`

*DO NOT MODIFY THE JAVA FILE NAMES AND EXISTING METHOD DECLARATIONS as we may test your code against an auto-grader, hence the interfaces must be consistent. However, you are welcome to ***add*** your own variables and methods as long as the existing method interfaces and variable names are not altered.

Since this project aims to test specific aspects of Java/OOP, you are only allowed to use the following mechanisms available in Java: loops, arrays, List, ArrayList, primitive data types, Double, String, methods, and classes. You may not use Math and other computation libraries, except for `Math.sqrt()` and `Math.pow()`. All the computation must be implemented from scratch using loops, arrays, and/or Lists. You may not use other container libraries (e.g., Set, Map, etc.) other than `ArrayList`. You are also not allowed to use third-party Java libraries. Your code should be able to compile and run without having to install external ***jar*** files.

Test cases:

Four test cases are provided for you to debug your code. Executing `KalculatorStudentTester`

should give the following output:

```
---- Welkome to Kalculator ----
@@ ========== TestCase 1 =========== @@
@@@ Loading data:
DATA[5]:[1.0, 2.0, 3.0, 4.0, 5.0]
@@@ Sum: 15.0
@@@ Avg: 3.0
@@@ Std: 1.5811388300841898
@@@ Max: 5.0
@@@ Max 3 numbers: [4.0, 5.0, 3.0] //order does not matter
@@@ Min: 1.0
@@@ Min 3 numbers: [1.0, 2.0, 3.0] //order does not matter
@@@ Removing last number:
DATA[4]:[1.0, 2.0, 3.0, 4.0]
@@@ Removing first number:
DATA[3]:[2.0, 3.0, 4.0]
@@ ========== TestCase 2 =========== @@
@@@ Loading data:
DATA[100]:[345.15, 312.05, 295.83, 615.04, 69.56, 503.49, 749.43, 183.16, 182.57, 235.97, 836.34, 861.94,
285.7, 701.31, 414.23, 402.71, 74.21, 442.75, 459.11, 404.91, 316.85, 74.98, 75.23, 558.88, 275.02, 164.72,
649.63, 646.32, 522.59, 119.77, 511.02, 2.74, 886.29, 842.32, 477.86, 173.03, 952.6, 22.82, 46.93, 308.46,
272.7, 130.69, 571.02, 906.69, 383.53, 269.72, 223.07, 22.34, 684.29, 119.31, 115.39, 853.67, 534.63, 166.93,
962.78, 617.22, 230.03, 405.11, 221.77, 100.41, 304.45, 972.89, 82.8, 245.05, 67.19, 227.95, 335.86, 126.91,
852.6, 655.57, 452.68, 175.06, 296.74, 85.48, 43.8, 555.17, 905.24, 121.54, 119.7, 943.05, 317.27, 22.06,
969.0, 800.87, 762.9, 846.74, 994.54, 225.32, 792.93, 88.92, 310.3, 210.52, 330.81, 843.69, 205.59, 214.26,
655.11, 393.64, 812.71, 602.67]
@@@ Sum: 41764.399999999994
@@@ Avg: 417.64399999999995
@@@ Std: 293.9801345690081
@@@ Max: 994.54
@@@ Max 3 numbers: [994.54, 969.0, 972.89] //order does not matter
@@@ Min: 2.74
@@@ Min 3 numbers: [22.34, 22.06, 2.74] //order does not matter
@@@ Removing last number:
DATA[99]:[345.15, 312.05, 295.83, 615.04, 69.56, 503.49, 749.43, 183.16, 182.57, 235.97, 836.34, 861.94, 285.7,
701.31, 414.23, 402.71, 74.21, 442.75, 459.11, 404.91, 316.85, 74.98, 75.23, 558.88, 275.02, 164.72, 649.63,
646.32, 522.59, 119.77, 511.02, 2.74, 886.29, 842.32, 477.86, 173.03, 952.6, 22.82, 46.93, 308.46, 272.7,
130.69, 571.02, 906.69, 383.53, 269.72, 223.07, 22.34, 684.29, 119.31, 115.39, 853.67, 534.63, 166.93, 962.78,
617.22, 230.03, 405.11, 221.77, 100.41, 304.45, 972.89, 82.8, 245.05, 67.19, 227.95, 335.86, 126.91, 852.6,
655.57, 452.68, 175.06, 296.74, 85.48, 43.8, 555.17, 905.24, 121.54, 119.7, 943.05, 317.27, 22.06, 969.0,
800.87, 762.9, 846.74, 994.54, 225.32, 792.93, 88.92, 310.3, 210.52, 330.81, 843.69, 205.59, 214.26, 655.11,
393.64, 812.71]
@@@ Removing first number:
DATA[98]:[312.05, 295.83, 615.04, 69.56, 503.49, 749.43, 183.16, 182.57, 235.97, 836.34, 861.94, 285.7, 701.31,
414.23, 402.71, 74.21, 442.75, 459.11, 404.91, 316.85, 74.98, 75.23, 558.88, 275.02, 164.72, 649.63, 646.32,
522.59, 119.77, 511.02, 2.74, 886.29, 842.32, 477.86, 173.03, 952.6, 22.82, 46.93, 308.46, 272.7, 130.69,
571.02, 906.69, 383.53, 269.72, 223.07, 22.34, 684.29, 119.31, 115.39, 853.67, 534.63, 166.93, 962.78, 617.22,
230.03, 405.11, 221.77, 100.41, 304.45, 972.89, 82.8, 245.05, 67.19, 227.95, 335.86, 126.91, 852.6, 655.57,
452.68, 175.06, 296.74, 85.48, 43.8, 555.17, 905.24, 121.54, 119.7, 943.05, 317.27, 22.06, 969.0, 800.87,
762.9, 846.74, 994.54, 225.32, 792.93, 88.92, 310.3, 210.52, 330.81, 843.69, 205.59, 214.26, 655.11, 393.64,
812.71]
@@ ========== TestCase 3 =========== @@
@@@ Loading data:
DATA[1]:[345.15]
@@@ Sum: 345.15
@@@ Avg: 345.15
@@@ Std: 0.0
@@@ Max: 345.15
@@@ Max 3 numbers: null //order does not matter
@@@ Min: 345.15
@@@ Min 3 numbers: null //order does not matter
@@@ Removing last number:
DATA[0]:[]
@@@ Removing first number:
DATA[0]:[]
@@ ========== TestCase 4 =========== @@
@@@ Loading data:
DATA[0]:[]
@@@ Sum: 0.0
@@@ Avg: 0.0
@@@ Std: 0.0
@@@ Max: 0.0
@@@ Max 3 numbers: null //order does not matter
@@@ Min: 0.0
@@@ Min 3 numbers: null //order does not matter
@@@ Removing last number:
DATA[0]:[]
@@@ Removing first number:
DATA[0]:[]
---- Have a kood day. Bye. ----
```

Coding Style Guideline:

It is important that you follow this coding style guideline strictly, to help us with the grading and for your own benefit when working in groups in future courses. Failing to follow these guidelines may result in a deduction of your project scores.

1. Write your name, student ID, and section on top of every code file that you submit.
2. Comment your code, especially where you believe other people will have trouble understanding, such as purposes of each variable, loop, and chunk of code. If you implement a new method, make sure to put an overview comment at the beginning of each method that explains 1) Objective, 2) Inputs (if any), and 3) Output (if any).

Submission:

It is important that you carefully follow the submission instructions. Failing to do so may result in a deduction and/or delay of your scores.

1. Put `Kalculator.java` in a folder.
2. Rename the folder to P0_<your *Student ID*>.
3. Zip the folder. Make sure the extension of the zip file is .zip (E.x., `P0_61881234.zip`).
4. Submit the zip file on MyCourses before the deadline.

Note: Late submission will suffer a penalty of 20% deduction of the actual scores for each late day. You can keep resubmitting your solutions, but the only latest version will be graded.

Suggestions:

1. Though Kalculator is a fairly simple calculator compared to other commercial ones in the market, the small details can overwhelm you during the course of implementation. Our suggestion is to spend some time to understand the requirements from the specs. Then, incrementally implement and test one method/player type at a time, to see if it behaves as expected. ***You are discouraged to implement everything up and test it all at once.—NOT COOL;***

2. Don't wait until the last weekend to start working on the project. Start early so you have enough time to cope with unforeseen situations. Plus, it can take some time to completely understand the project specs.
3. Use Java JDK 1.8 or 1.9. Some of the functionalities may not be available in the older versions.
4. ***Don't use packages***. Put everything in the default package.

Bug Report:

Though not likely, it is possible that our solutions may contain bugs. A bug in this context is not an insect, but error in the solution code that we implemented to generate the test cases. Hence, if you believe that your implementation is correct, yet yields different results, please contact us immediately so proper actions can be taken.

Need help with the project?

If you have questions about the project, please first post them on the forum on MyCourses, so that other students with similar questions can benefit from the discussions. The TAs can also answer some of the questions and help to debug trivial errors. If you still have questions or concerns, please make an appointment with one of the instructors. ***We do not debug your code via email***. If you need help with debugging your code, please come see us.

Academic Integrity (Very Very Important)

Do not get bored about these warnings yet. But please, please ***do your own work***. Your survival in the subsequent courses and the ability to get desirable jobs (once you graduate) heavily depend on the skills that you harvest in this course.

Though students are allowed and encouraged to discuss ideas with others, the actual solutions must be originated and written by themselves. Collaboration in writing solutions is not allowed, as it would be unfair to other students. Students who know how to obtain the solutions are encouraged to help others by guiding them and teaching them the core material needed to complete the project, rather than giving away the solutions. ***You can't keep helping your friends forever, so you would do them a favor by allowing them to be better problem solvers and life-long learners.*** Your code will be compared with other students' (both current and previous course takers) and online sources using state-of-the-art source-code similarity detection algorithms which have been proven to be quite accurate. If you are caught cheating, serious actions will be taken, and severe penalties will be bestowed on all involved parties, including inappropriate help givers, receivers, and intermediaries.