CSCI 221: Computer Programming II **Due Sunday October 23, 10pm**
Programming Assignment 3 Total Points 100

Collaboration Policy: Work alone. Write your own code. You may not use any code found on the internet or anywhere else in this assignment, except for the test code that I provide.

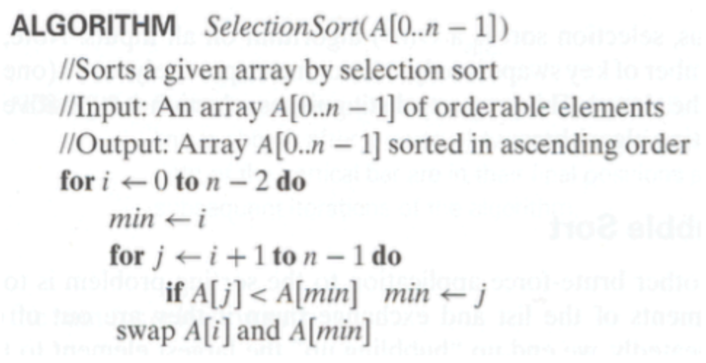
You're working on the open source Apache Commons Mathematics Library (http://commons.apache.org/proper/commons-math/) project and the open source group would like to include the selection sort algorithm in the software baseline. The open source testing team has already developed the sort test cases that are provided in the `SortTest` class. You must use this class to test the correctness of your selection sort implementations. The provided `SortTest` class **is not** to be modified under any circumstances. Include the `SortTest.java` file into your project.

Your job is to develop an `Utils` class whose API specification is provided below. Make sure the names of your methods match those provided below **exactly**, including capitalization. The number of parameters and their order must also match. Otherwise, your implementations will not work correctly and the test cases will fail.

Utils Class
Utils class is very simple; it has a zero-argument constructor that is private, and no instances variables. The API specification for the `Utils` class is provided below.

private Constructor	**Description and Preconditions (if any)**
`Utils()`	A **private** no argument constructor. Making the constructor private ensures the class cannot be instantiated. Precondition: None.

public Methods	**Description and Preconditions (if any)**
`selectionSort( int[] A )`	**static** method that implements the selection sort algorithm shown in Figure 1. This overloaded method sorts, in ascending order, an integer array. Precondition: `A != null`. If the array is null, do nothing. No return value (i.e. return type void)
`selectionSort( double[] A )`	**static** method that implements the selection sort algorithm shown in Figure 1. This overloaded method sorts, in ascending order, a double array. Precondition: `A != null`. If the array is null, do nothing. No return value (i.e. return type void)



```
ALGORITHM SelectionSort(A[0..n − 1])
    //Sorts a given array by selection sort
    //Input: An array A[0..n − 1] of orderable elements
    //Output: Array A[0..n − 1] sorted in ascending order
    for i ← 0 to n − 2 do
        min ← i
        for j ← i + 1 to n − 1 do
            if A[j] < A[min]   min ← j
        swap A[i] and A[min]
```

Figure 1: The Selection Sort Algorithm

Additional Notes:

- Pseudo-code is provided in Figure 1 (i.e. not specific to the Java programming language). Pseudo-code is a high-level description of the algorithm. It is your job to convert the pseudo-code to workable code that meets the Java syntax.
- There are several ways to implement the swap function in Figure 1. In short, the swap function exchanges the values at index positions `min` and `i` in the array. For instance, say `A=[1,2,5,4,3]` if the values at index positions `i=2` and `min=4` are swapped, then result will be `A=[1,2,3,4,5]`. As the swap method is not needed outside of this class, it should be a private method.

Additional Specifications:

- Use NO package for the Utils class in this assignment. Just put the one file you create (with no package line) in a folder named by your last name and zip it up. (See below.)

SortTest Class

As stated earlier, this class is used to test the correctness of the overloaded selection sort methods you implemented in the `Utils` class. In short, you do not need to develop your own test class with your own test cases.

The `SortTest` class **is not** to be modified under any circumstances. If your implementation of the overloaded selection sort methods is 100% correct, the SortTest class will display:

```
[integer] 100 of 100 Sort Tests Passed [100.00 %]
[double] 100 of 100 Sort Tests Passed [100.00 %] ---------------------------
Total Score = 100.00 % ---------------------------
```

Any score less than 100% means your implementation of the overloaded selection sort methods is not correct. **For grading purposes, Paul and I will use the same test cases to grade this assignment.**

Program Documentation

Refer to the documentation standard posted for HW1. Each method MUST be preceded by an appropriate Java doc comment. Use Eclipse short cuts (e.g. /** - return key) to generate comment stubs above class definition and methods. Follow the example given in class, or setup an appointment with instructor or TA if you require more direction.

For each method in the `Utils` class (in the comment block above the method) state the pre- and post-conditions, i.e. what is expected before and after the method is called (see page 300 in the course textbook).

Program Submission

Create a ZIP file that only contains the completed `Utils` java file, that is, no byte-code files (i.e. .class files) are to be submitted. Furthermore, please do not include a file hierarchy (i.e. the package structure). For the ZIP file you **must** use the naming convention:
<lastname>.zip e.g., McCauley.zip

If the assignment is not submitted in the correct format – it will not be accepted – no exceptions this is HW 3. Submit the ZIP file via OAKS in the Dropbox that corresponds to the assignment. Resubmit, as many times as you like, the newest submission will be the graded submission.

Grading Rubric

If the submitted program does not compile: 0 of 80 points
If the submitted program compiles but does not run: 5 of 80 points
If the submitted program compiles and runs: 10 of 80 points
If the submitted program compiles, runs, and produces correct output: 80 of 80 points
Late assignments will not be accepted – no exceptions (**please** do not email me your assignment after the due date, **I will not accept it**).
Please feel free to setup an appointment to discuss the assigned problem. I'll be more than happy to listen to your approach and make suggestions. However, I cannot tell you how to code the solution.