CSCI 3400
Fall 2022
Assignment 03: Running Time Analysis of Sorting Algorithms
Assigned date: 10/31/2022
Due date: 11/14/22 (11:59 PM, EST)
Total points: 90 points
Please complete the assignment individually

Learning Goal:

- To apply sorting algorithms
- To keep track of running time
- To use external libraries

Problem description:

Part A: Implementing two sorting algorithms [20 points]

Sorting is an important part of a lot of applications. We want to store our collection of clothes, shoes, books, etc. in some order so that we can quickly find them at the time of need. There are a lot of sorting techniques that people have invented; some are fast and efficient; some are not so fast. Still, we use/need them every day for various tasks. The followings are two sorting techniques (algorithm/pseudocode) that you will implement as part of this assignment.

```
procedure bubbleSort(A : list of sortable items)
    n := length(A)
    repeat
        swapped := false
        for i := 1 to n-1 inclusive do
            /* if this pair is out of order */
            if A[i-1] > A[i] then
                /* swap them and remember something changed */
                swap(A[i-1], A[i])
                swapped := true
            end if
        end for
    until not swapped
end procedure
```

```
procedure insertionSort(A : list of sortable items)

    n := length(A)
    i ← 1
    while i < n
        j ← i
        while j > 0 and A[j-1] > A[j]
            swap A[j] and A[j-1] //interchange the values
            j ← j - 1
        end while
        i ← i + 1
    end while
end procedure
```

In `DemoSorting.java` (check the starter code), you will find a class named SortingAlgorithms. Implement/complete the `insertionSort` and `bubbleSort` methods.

Part B: Analyzing of Four Sorting Techniques [20 points]

In the starter code, the `main` method generates an array of 20 random Integers and calls the `mergeSort` method to sort it. It also prints the elapsed time taken by the mergeSort algorithm. Note that the print and the copy operation times are not part of the elapsed time taken by the mergeSort method.

Your next job is to create a class **RunningTimeRecorder** that has a method, **empiricalRunningTime**. The method does the followings:

a) take a number as input (like n >= 5, 10, 50, 100, 200, 500, 1000, 10000)
b) generate an array of n-random numbers,
c) call insertionSort on the unsorted array and keep track of the time
d) call selectionSort on the unsorted array and keep track of the time
e) call mergeSort on the unsorted array and keep track of the time
f) call bubbleSort on the unsorted array and keep track of the time

Remember: Call the sorting algorithms for the same unsorted array and record the elapsed time.
Note: Do not reuse the array that is already sorted by another sorting algorithm.

You may store the times in a 2D (4 by 1, for the four sorting techniques) array/list and return it to the caller. Each row in your 2D array represents an algorithm and the column represents the required/elapsed time.

Now, from the main method/caller, call the empiricalRunningTime for the following values of n:

5, 10, 50, 100, 200, 500, 1000, 2000, 5000, 10000.

Record (write) the output (running time) in a file called **assignment02RunningTimeLastName.csv**.

Part 03: Visualizing the data [20 points]

- Using the data from your csv file/or the 2D array, create a graph to represent the running time. You may use the **XChart library** (https://knowm.org/open-source/xchart/) to draw the running time chart of the tested algorithms.
- If you are familiar with other libraries that serve the same purpose, feel free to use it.

- **How to draw charts:**
  - First, download **xchart** library.
  - On Eclipse, right click on the package/project you are working on today.
  - Then select BuildPath >> Configure Build Path >> Libraries >> Add External Libraries
  - Then add the two jar files (`xchart-demo-3.6.1.jar, xchart-3.6.1.jar`) to your package/project.

Those who are not using Eclipse: While you are trying to compile and run your source code from command line, add the jars to the class path like the following:

```
java -cp "xchart-demo-3.6.1.jar;xchart-3.6.1.jar" XChartDemo
```

assuming XChartDemo is your source code.

You may need to add some of the classes:

```
import org.knowm.xchart.QuickChart;
import org.knowm.xchart.SwingWrapper;
import org.knowm.xchart.XYChart;
```

More hints: https://knowm.org/open-source/xchart/xchart-example-code/

Sample code:

```
double[] xData = new double[] { 0.0, 1.0, 2.0 };
double[] yData = new double[] { 2.0, 1.0, 0.0 };
    // Create Chart
XYChart chart = QuickChart.getChart("Sample Chart", "X", "Y", "y(x)", xData,
yData);
    // Show it
new SwingWrapper(chart).displayChart();
BitmapEncoder.saveBitmap(chart, "./Sample_Chart", BitmapFormat.PNG);
```

Resources:

[1] https://en.wikipedia.org/wiki/Bubble_sort
[2] https://www.hackerearth.com/practice/algorithms/sorting/bubble-sort/tutorial/
[3] https://knowm.org/open-source/xchart/

Disclaimer:

I do not mind if you find a solution/sample algorithm/class. But you have to cite it. Remember, you are not allowed to post the question to any online board (like StackOverflow or so) and ask for someone to write the solution for you. If I detect that, you will get a 0 and I will report it to the corresponding authority.

Documentation:

Your code has to be well documented. For each class and method that you write, you will have to add JavaDoc comments (at least) at the beginning of the class.

Submission:

Submit all the source codes, one README file, **assignment03RunningTimeLastName.csv**, and one graph (png/jpeg). In the README should look like the following:

```
*******************************************************
Assignment 03: Running Time Analysis
The assignment solution is written by:
Your name

The source code directory contains the following files:
     DemoSorting.java
```

```
    RandomNumberGenerator.java
    ………… etc.
    Assignment03.java
```

Assignment 03.java should contain the main method to produce expected results.

Instructions:

Place all the files in a project ….
Or

```
$ javac *.java
$ java Assignment03 < sampleInputfile01.txt
```

Note: The system will automatically compute the result and store it in a file called sampleOutputfile01.txt

****

Point Distribution:

```
** code:        60
** README file: 10
** csv file:    10
** graph:       10
** total:       90
```

Please note that we may take partial points off if the output is not printed properly, if the algorithm's implementation is wrong, and if the SOLID principles were not properly followed.

Email:
For any clarification, feel free to ask in the class or email your instructor.