Peng Chen (001098655)

**Program Structures & Algorithms**

**Fall 2021**

**Assignment No.2**

GitHub: [https://github.com/pngchen/INFO6205-Assignments/tree/main/assignment](https://github.com/pngchen/INFO6205-Assignments/tree/main/assignment1)2

**Task:**

* (Part 1) You are to implement three methods of a class called Timer. Please see the skeleton class that I created in the repository. Timer is invoked from a class called Benchmark\_Timer which implements the Benchmark interface.
* (Part 2) Implement InsertionSort (in the InsertionSort class) by simply looking up the insertion code used by Arrays.sort. If you have the instrument = true setting in test/resources/config.ini, then you will need to use the helper methods for comparing and swapping (so that they properly count the number of swaps/compares). The easiest is to use the helper.swapStableConditional method, continuing if it returns true, otherwise breaking the loop. Alternatively, if you are not using instrumenting, then you can write (or copy) your own compare/swap code. Either way, you must run the unit tests in InsertionSortTest.
* (Part 3) Implement a main program (or you could do it via your own unit tests) to actually run the following benchmarks: measure the running times of this sort, using four different initial array ordering situations: random, ordered, partially-ordered and reverse-ordered. I suggest that your arrays to be sorted are of type Integer. Use the doubling method for choosing n and test for at least five values of n. Draw any conclusions from your observations regarding the order of growth.

**Conclusion:**

**Time of sorting:**

Reverse Ordered Array > Random Array > Partially Ordered Array > Ordered Array

About Part 3, sorting the ordered array is the fastest one. Because the array has been sorted. Sorting the reverse ordered array is the slowest one. Because every element in the array needs to be sorted. And sorting the partially ordered array is faster than sorting the random array. (The data is in Assignment2.xlsx)

**Evidence:**

* Part 1:

BenchmarkTest:

Graphical user interface, text

Description automatically generated

TimerTest:

Graphical user interface, text

Description automatically generated

* Part 2:

InsertionSortTest:

Text

Description automatically generated

* Part 3:

(The main is in Benchmark\_Timer.java.)

Graphical user interface

Description automatically generated with medium confidence