Zhen Luo (001528318)

**Program Structures & Algorithms**

**Fall 2021**

**Assignment No. 2**

* **Task (List down the tasks performed in the Assignment)**
  + 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.
  + Implement InsertionSort(in the InsertionSort class) by simply looking up the insertion code used byArrays.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.
  + 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.
* **Relationship Conclusion:** 
  + **When the data size is small, different inputs have a great impact on the sorting time. When the data size becomes large, different inputs have little impact on the sorting time.**
* **Evidence to support the conclusion:**

1. **Output**

**电脑萤幕的截图

描述已自动生成**

1. **Graphical Representation**

**图表, 条形图

描述已自动生成**

* **Unit tests result:**

电脑萤幕的截图

描述已自动生成文本

描述已自动生成文本

描述已自动生成