**Assignment2**

FengyiZhang Sec-02 001564247

**Part1.**

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.

**Screen shot for part 1**

**图形用户界面, 文本, 应用程序

描述已自动生成图形用户界面, 文本, 应用程序

描述已自动生成**

**Part2.**

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*.

**Screen shot for part 2**

**图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成**

**Part3.**

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.

**Screen shot for Part3.**

**图形用户界面, 文本, 应用程序

描述已自动生成**

**图表, 折线图

描述已自动生成**

**Conclusion:**

As N goes up, the order of time taken is Ordered(fastest), Partly Ordered,

Random, Reverse(slowest).