**INFO 6205**

**Program Structures & Algorithms**

**Fall 2020**

**Assignment 2**

* **Task**

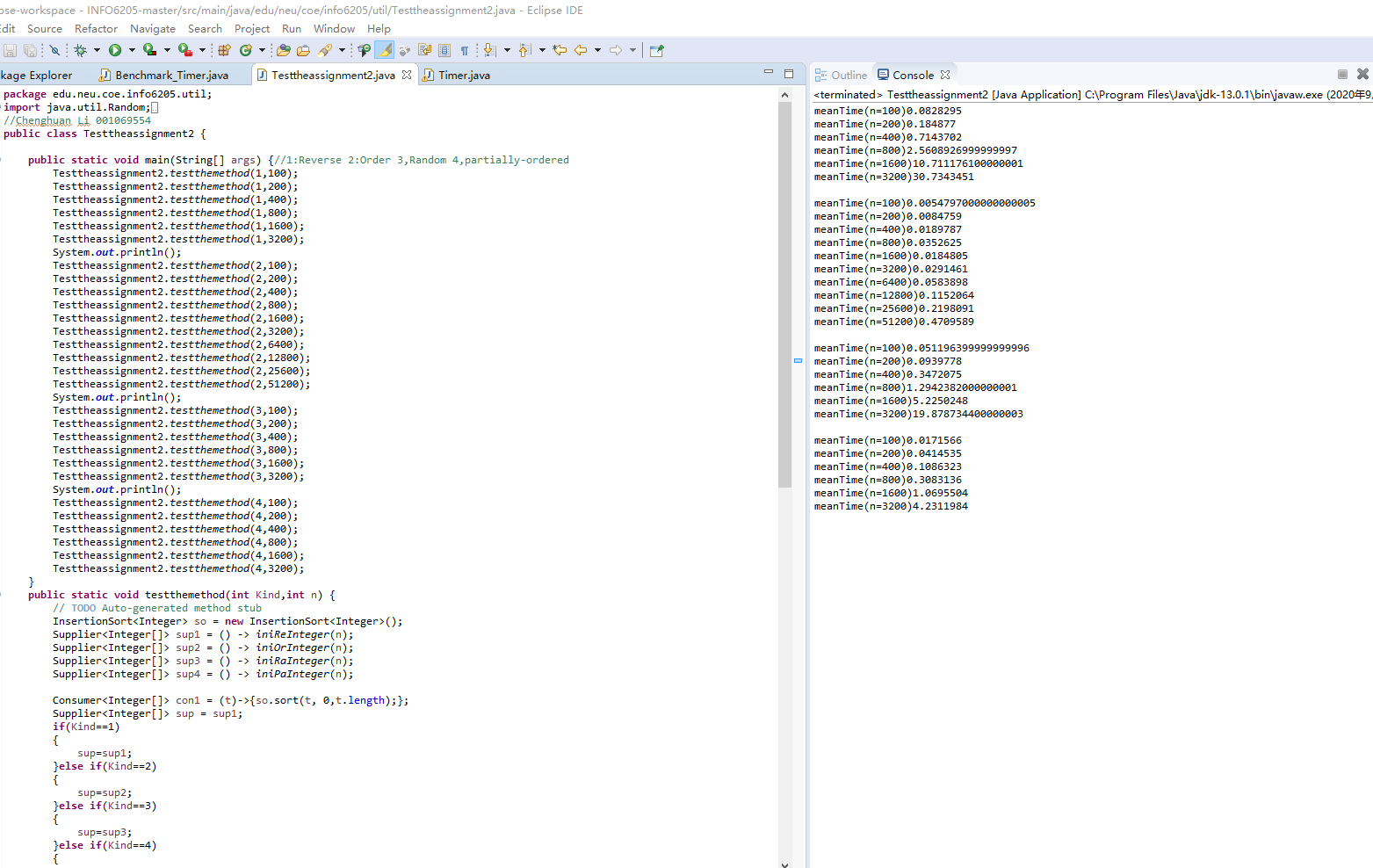
(Part 1) You are to implement four methods of a class called *Timer*. Please see the skeleton class that I created in the repository. Timer is invoked from a class called Benchmark\_Timer which implements the Benchmark interface.

(Part 2) Implement *InsertionSort*(in the *InsertionSort* class) by simply looking up the insertion code used by*Arrays.sort.* You should use the *helper.swap* method although you could also just copy that from the same source code. In the *main* method of *Benchmark*, remove the reference to *SelectionSort*.

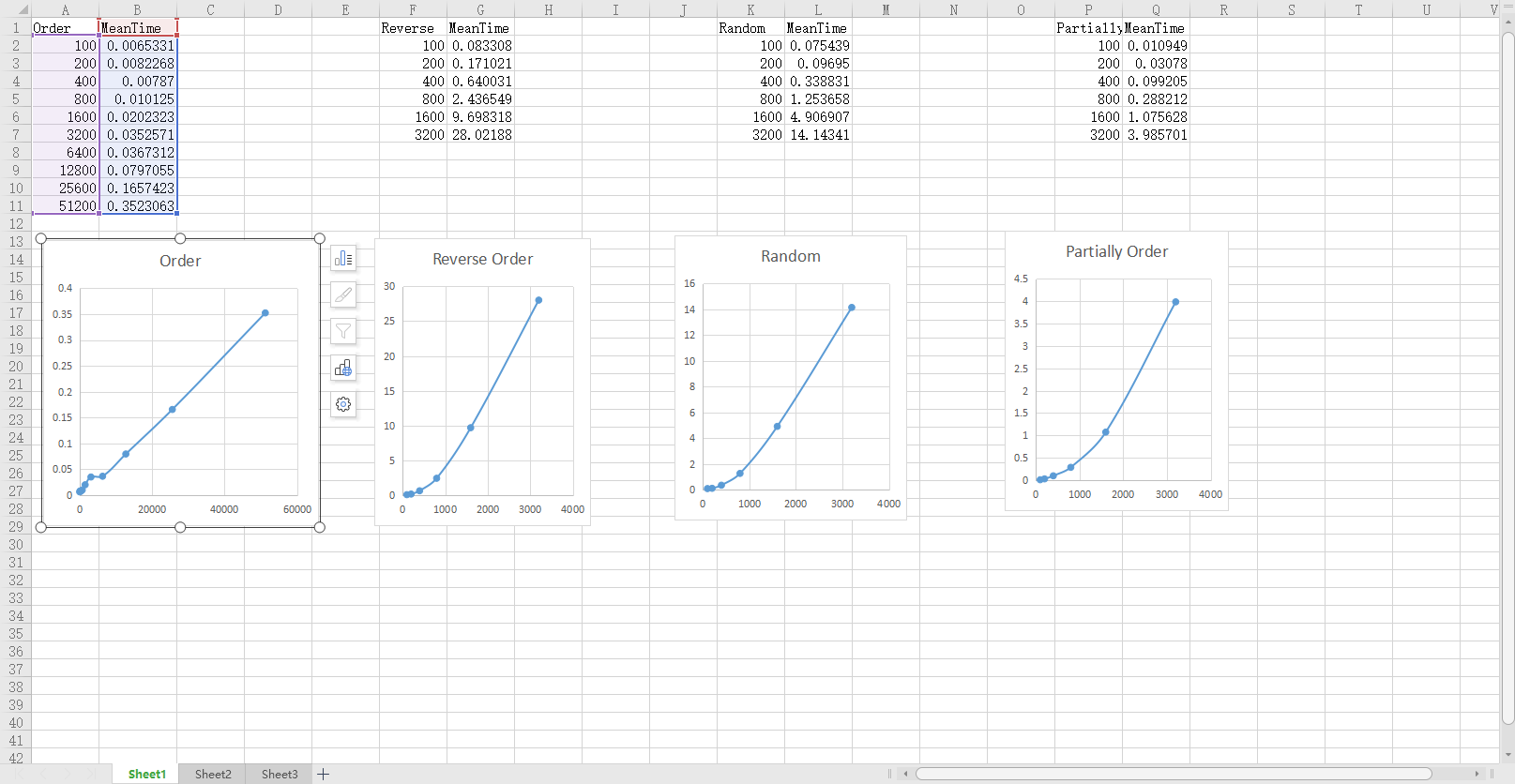
(Part 3) Measure the running times of this sort, using four different initial array ordering situations: random, ordered, partially-ordered and reverse-ordered. I suggest that your arrays to be sorted are of type *Integer*. Use the doubling method for choosing *n*and test for at least five values of *n.*Draw any conclusions from your observations regarding the order of growth.

* **Output** (few outputs to prove relationship)

1. Result



2，Relartionship Chart



* **Relationship conclusion**

Order 

Reverse-Order 

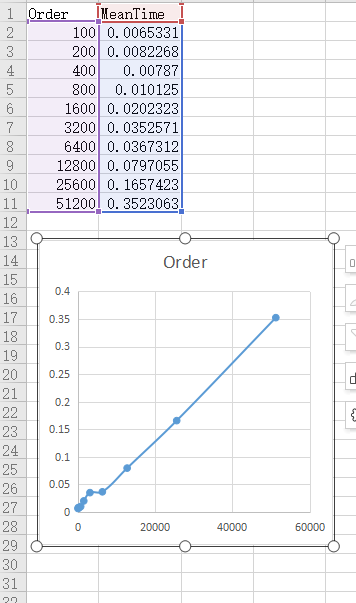
Random-Order 

Partially-Order 

* **Evidence to support relationship** (screen shot and/or graph and/or spreadsheet)

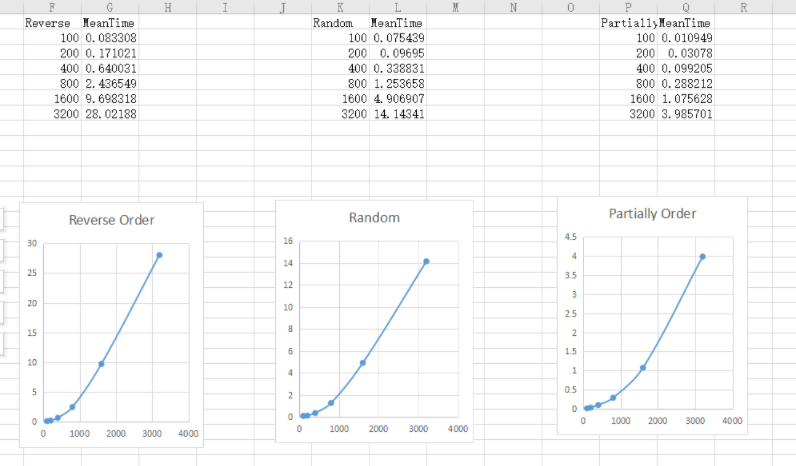
About the Order, through the test data, we found that when the value of n is not too small. The average time increases with the increase of N. In addition, the relationship between them is approximate Meantime = k N

Their relationship line is approximately a straight line, so their relationship is 



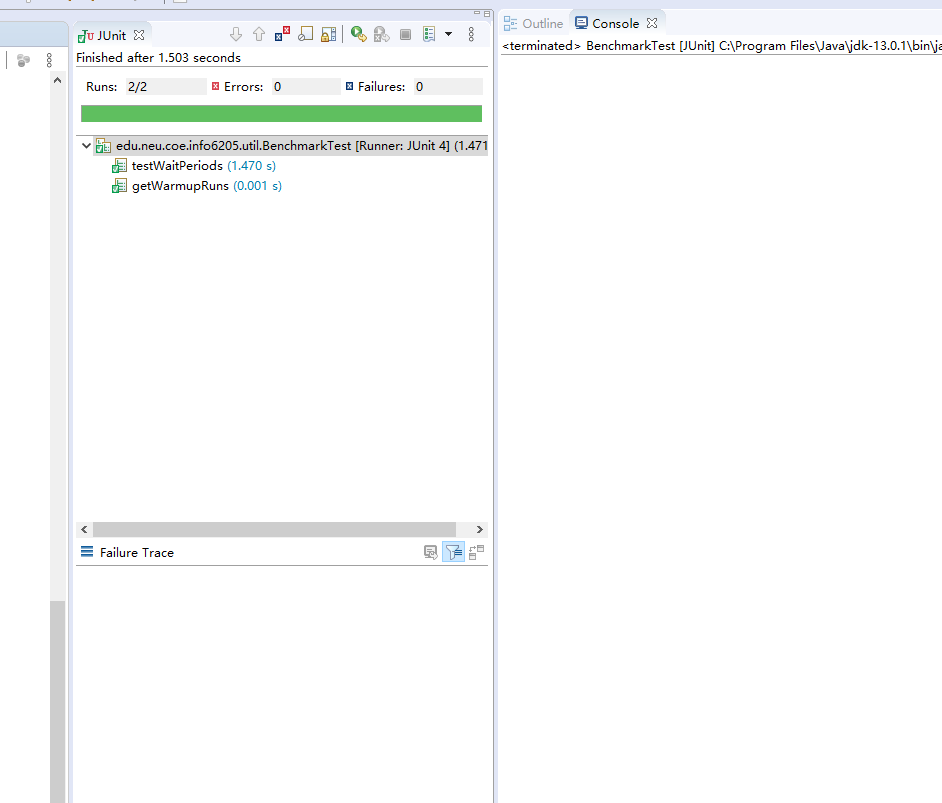
About the Reverse-Order, Partially-Order, Random-Order, Curves is near

, So the relationship is .

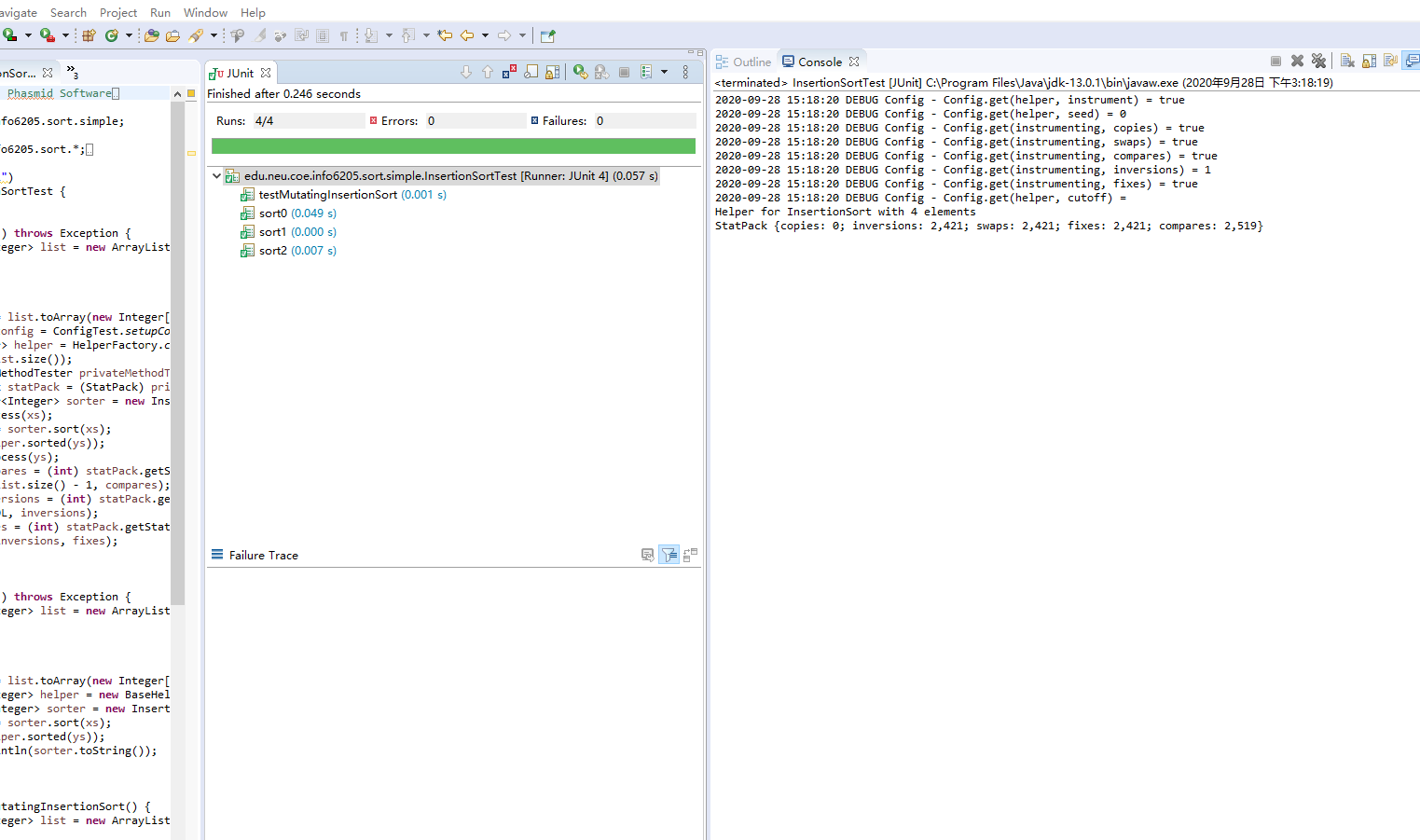


* **Screenshot of Unit test passing**

**BenchMarkTimer Test Pass**



**InsertionSort Test Pass**



**TimerTest Pass**

