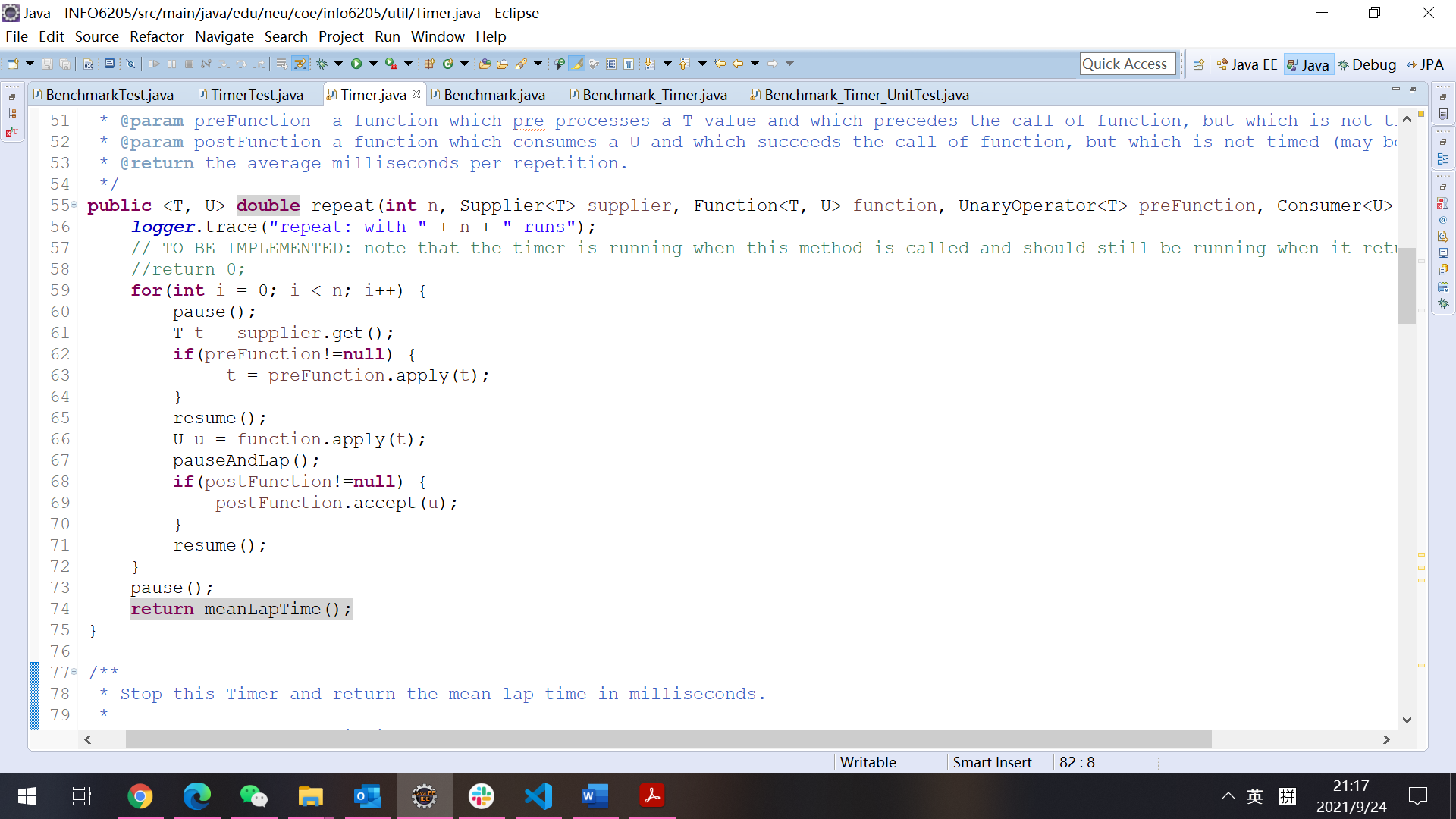
**Program Structures & Algorithms**

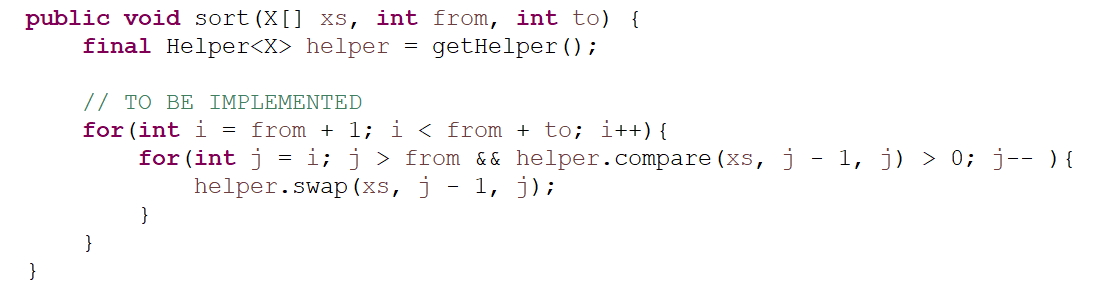
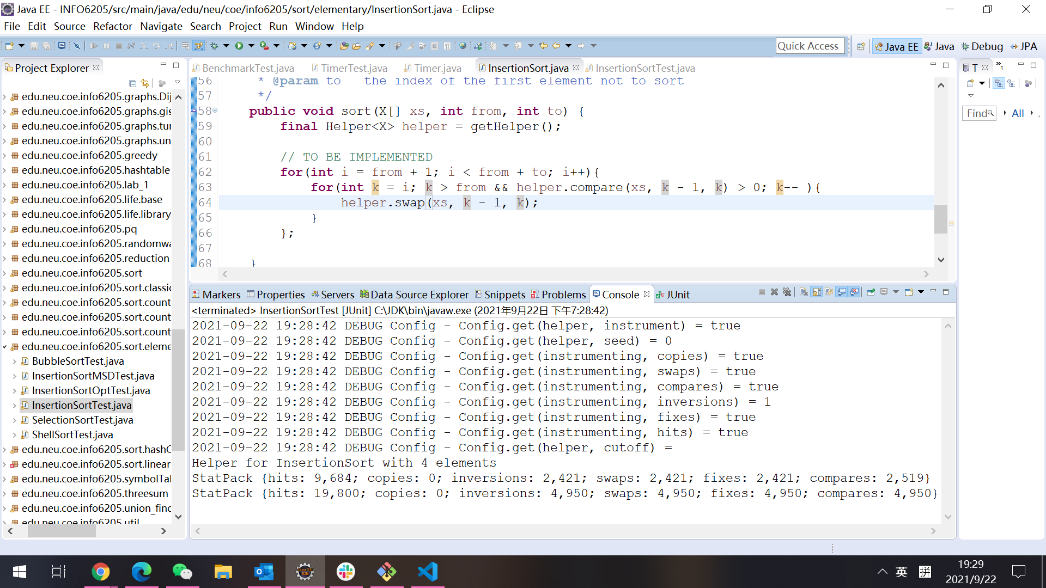
**Fall 2021**

**Assignment No. 2**

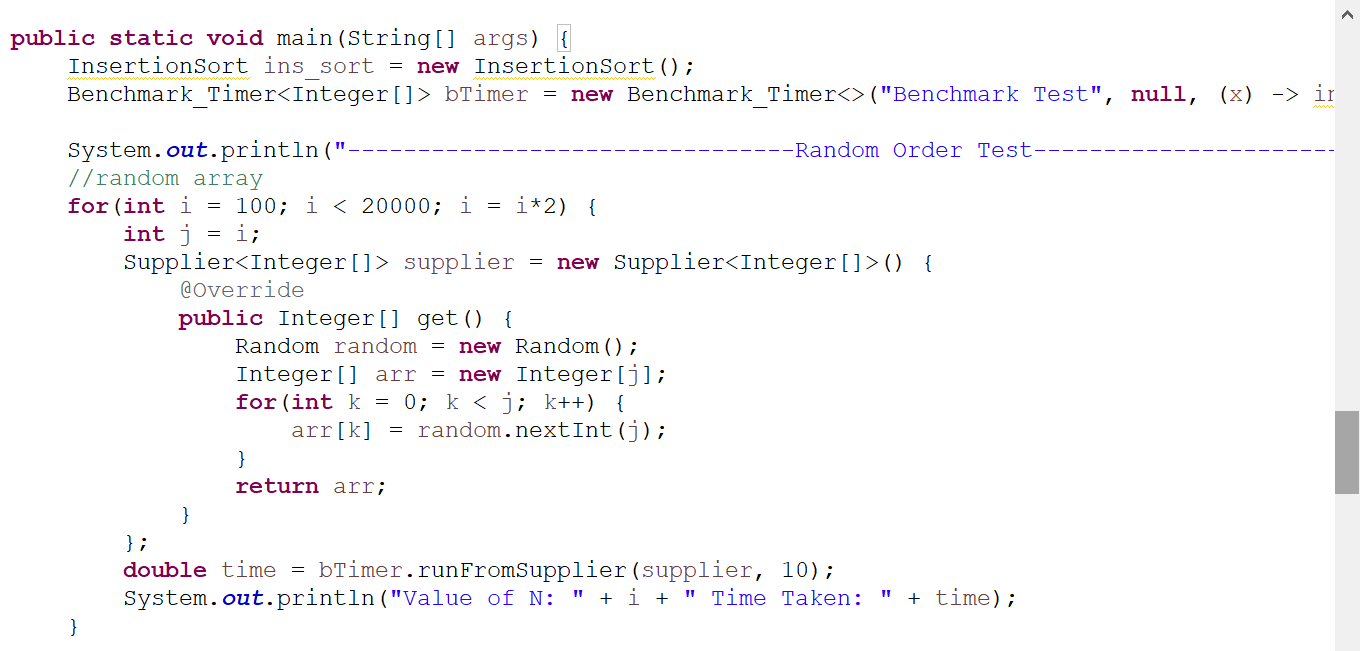
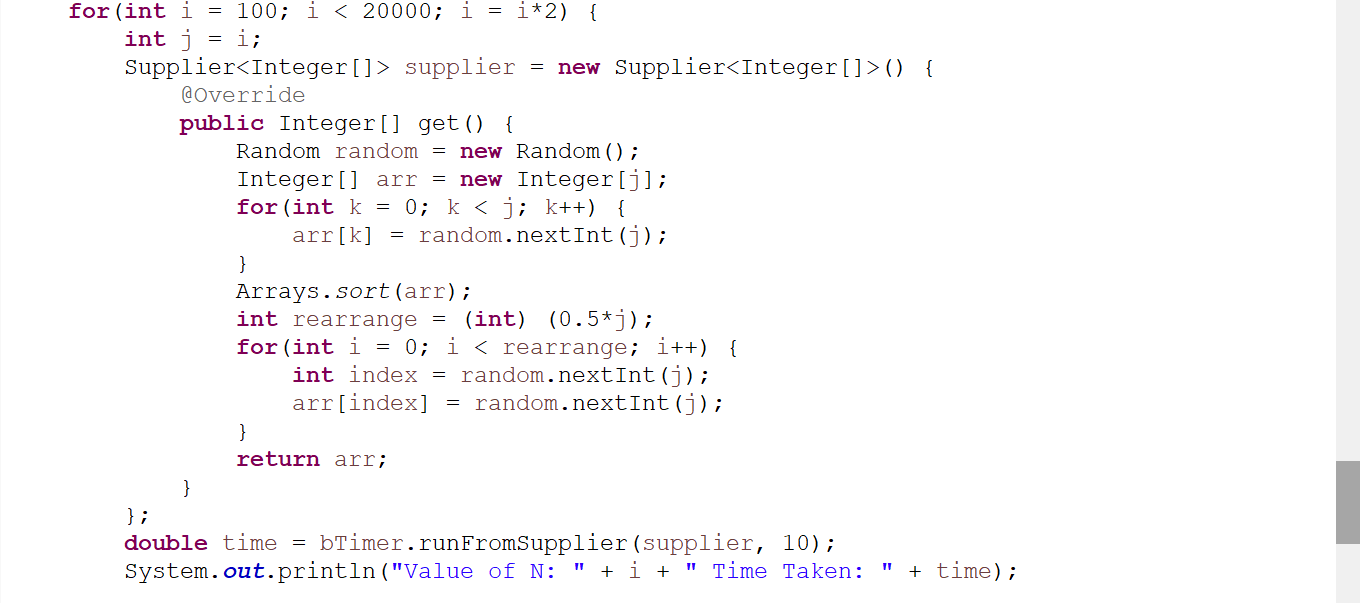
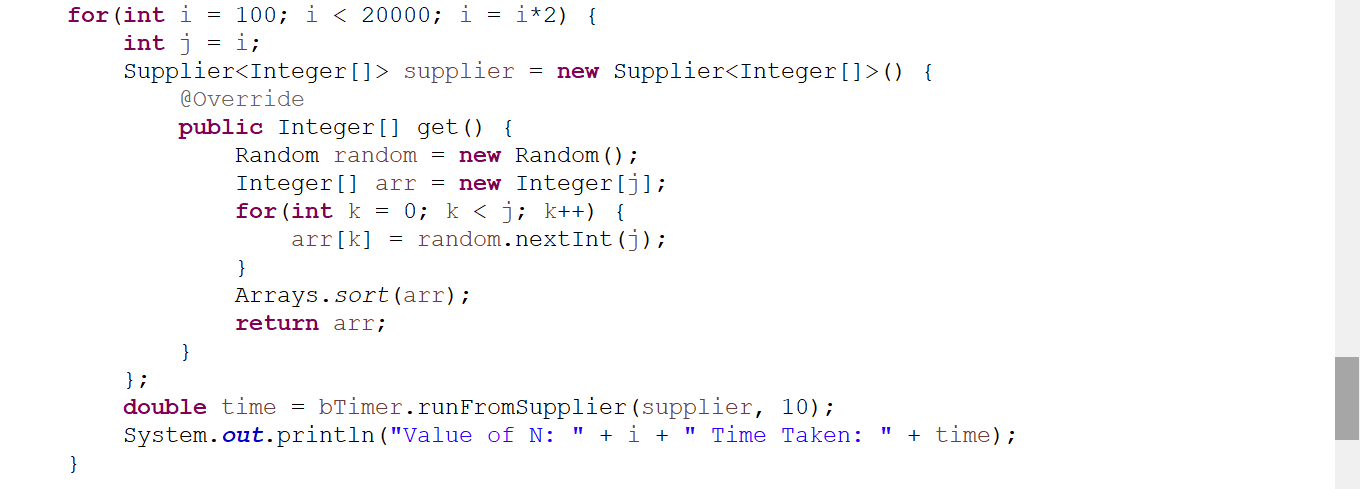
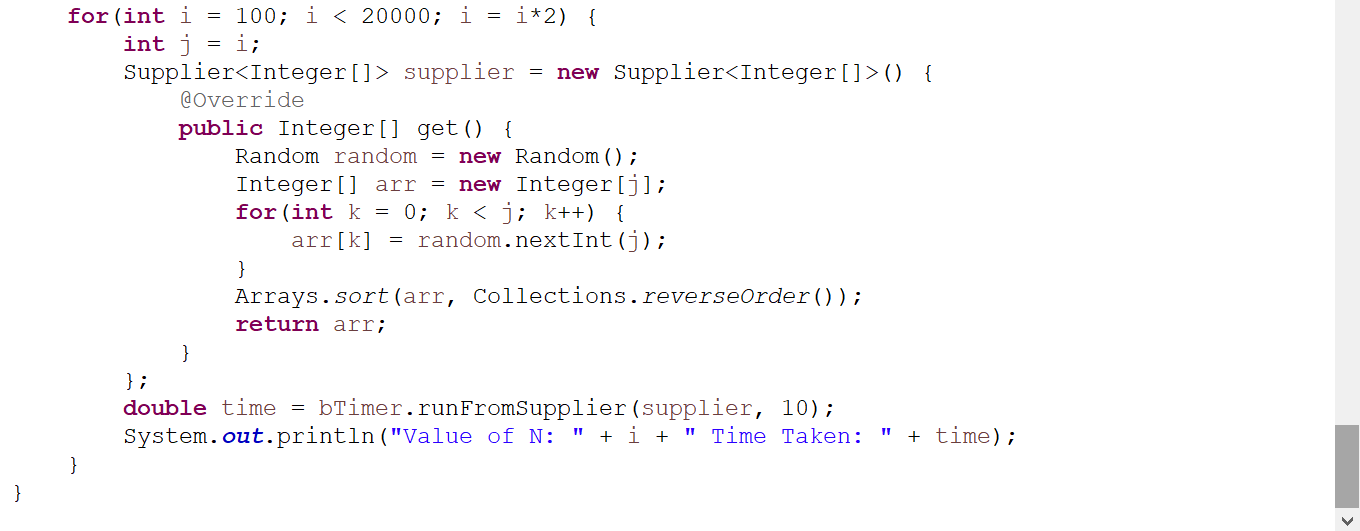
* + **Task (List down the tasks performed in the Assignment)**
  + Part1

Implement three methods of a class called Timer. The function to be timed, hereinafter the "target" function, is the Consumer function fRun (or just f) passed in to one or other of the constructors. The generic type T is that of the input to the target function. The first parameter to the first run method signature is the parameter that will, in turn, be passed to target function. In the second signature, supplier will be invoked each time to get a t which is passed to the other run method. The second parameter to the run function (m) is the number of times the target function will be called. The return value from run is the average number of milliseconds taken for each run of the target function.

* + Part2

Implement InsertionSort (in the InsertionSort class) by simply looking up the insertion code used by Arrays.sort.

* + 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.

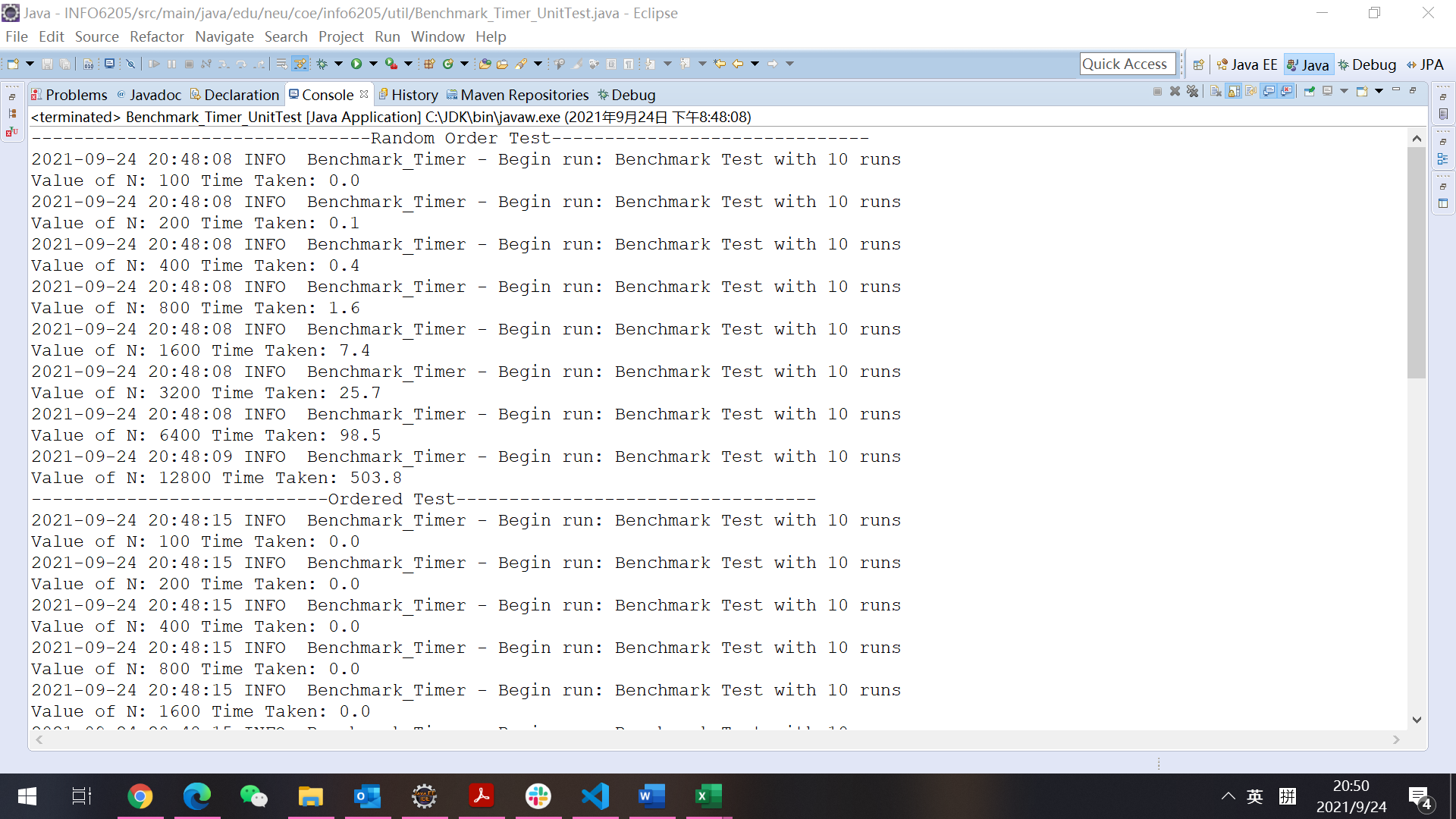
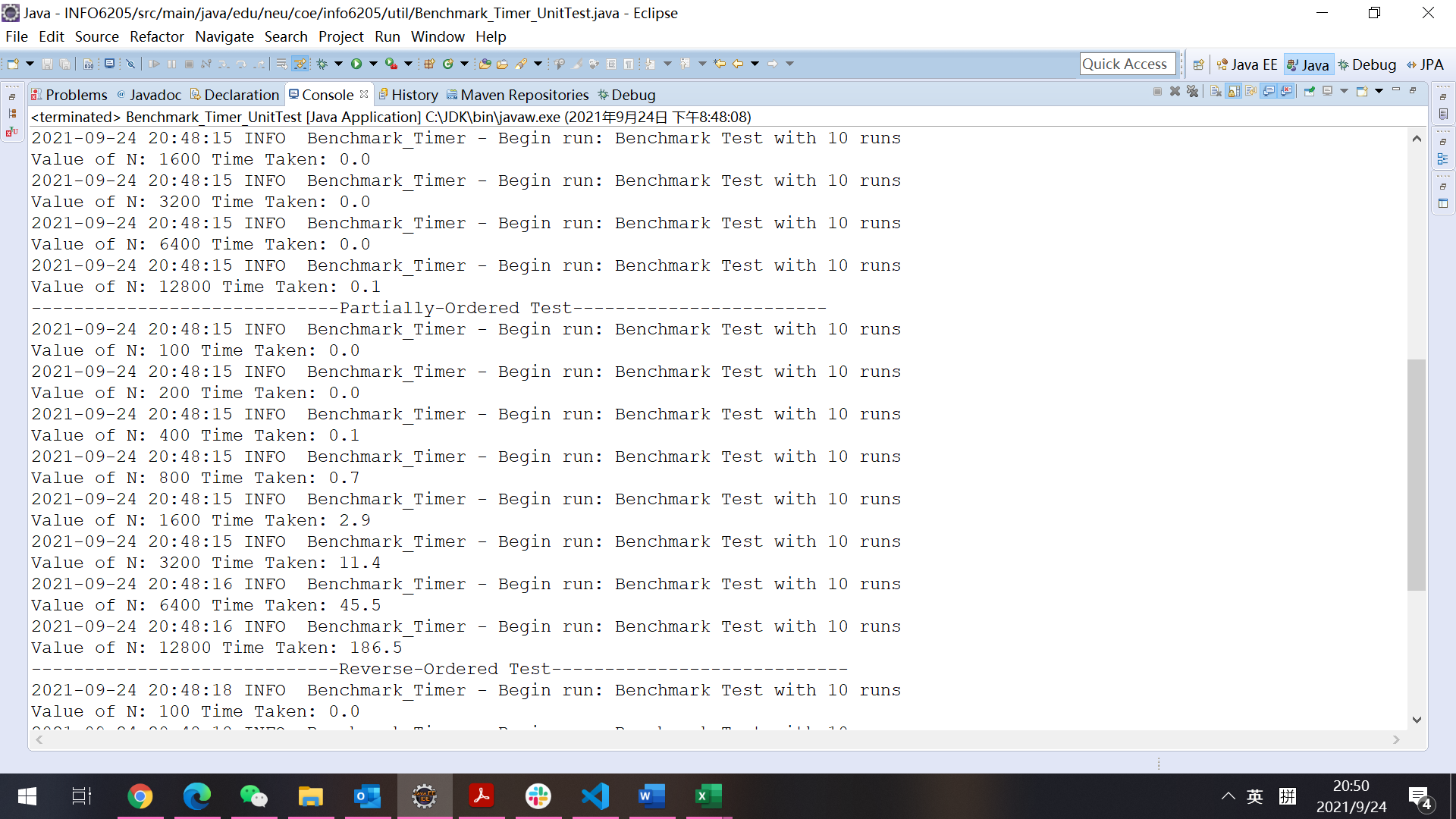
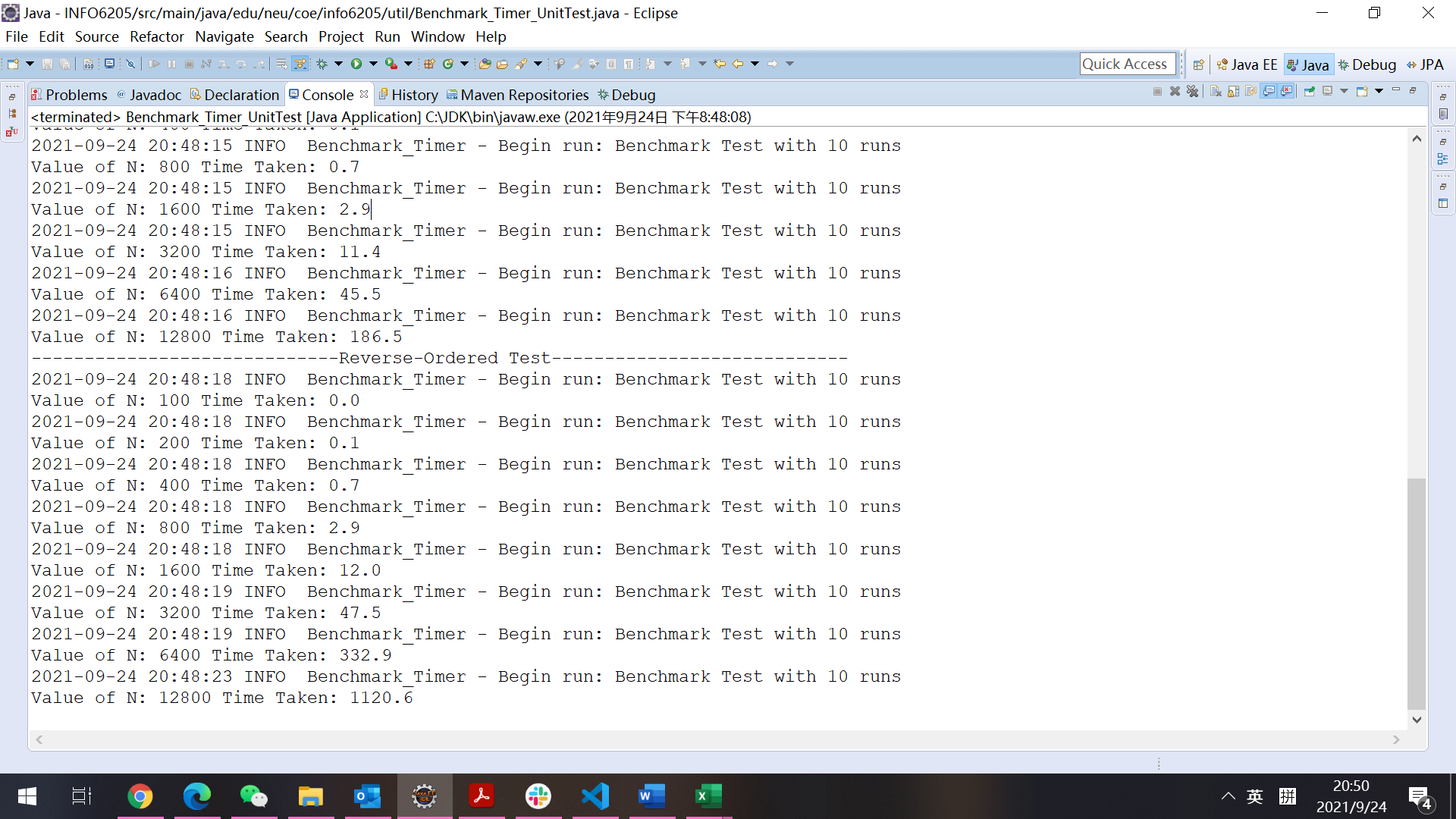
* + **Relationship Conclusion: (For ex : z = a \* b)**

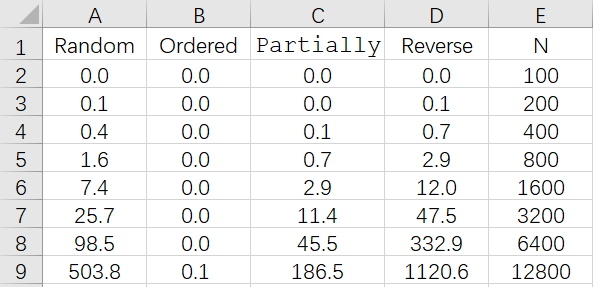
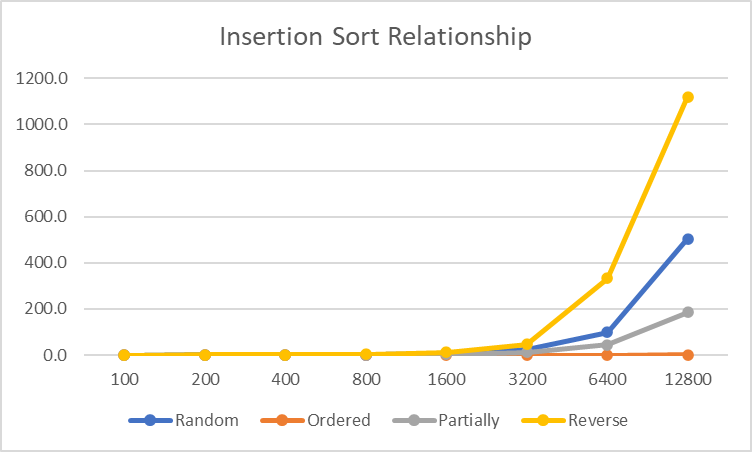
In ordered cases, the array is ordered, the insertion sort runs in **O(𝑛)** times.

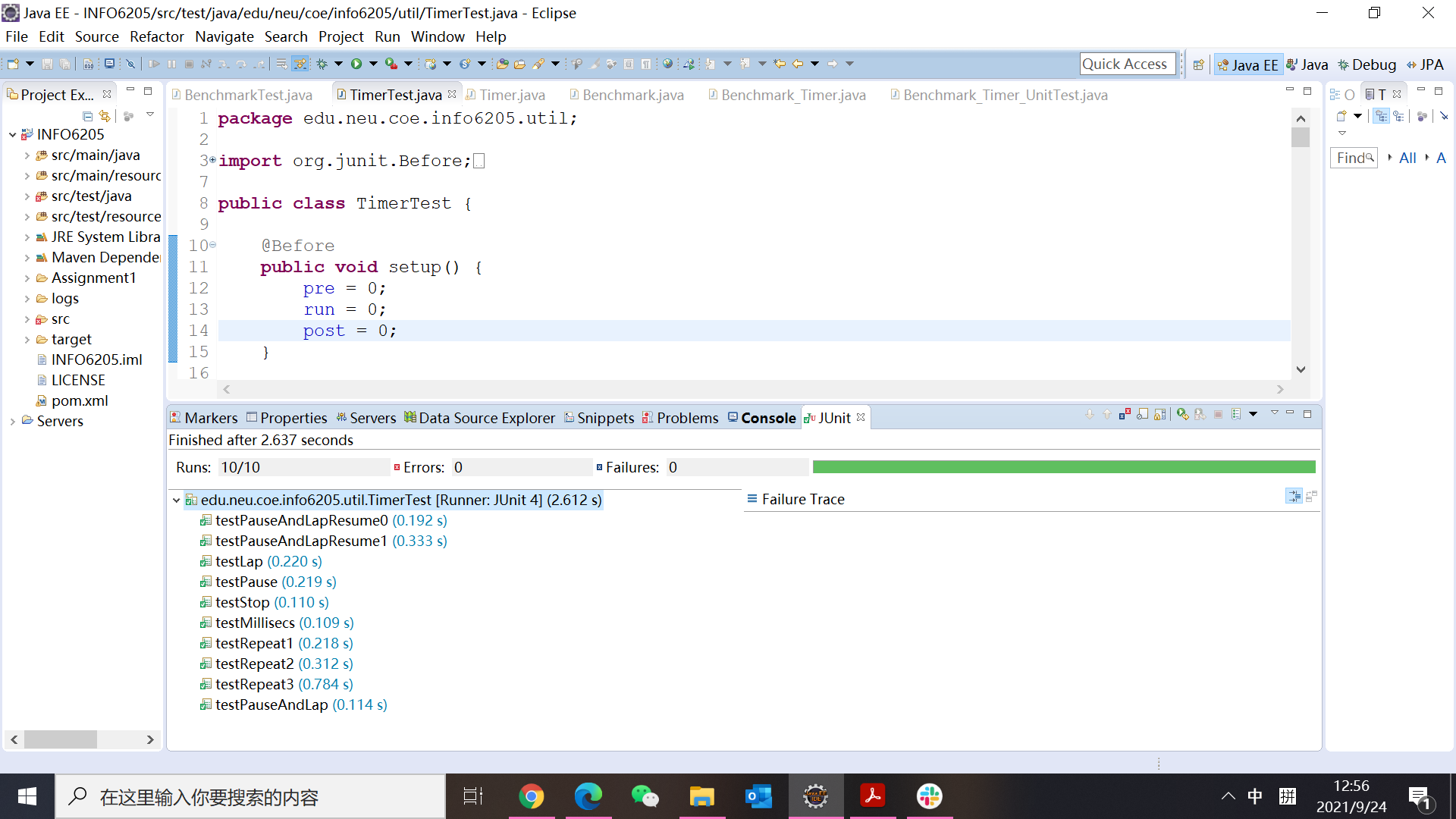
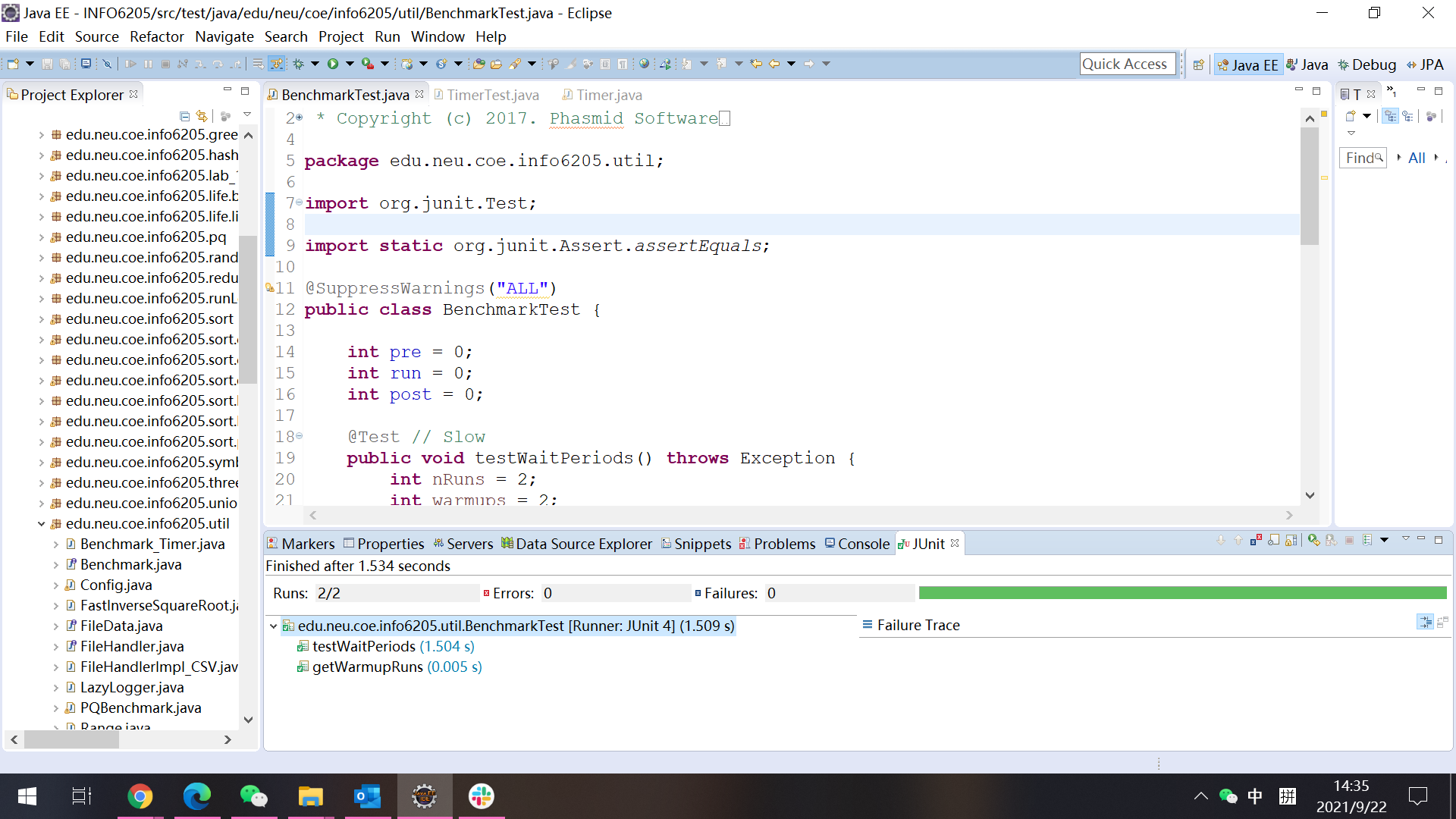
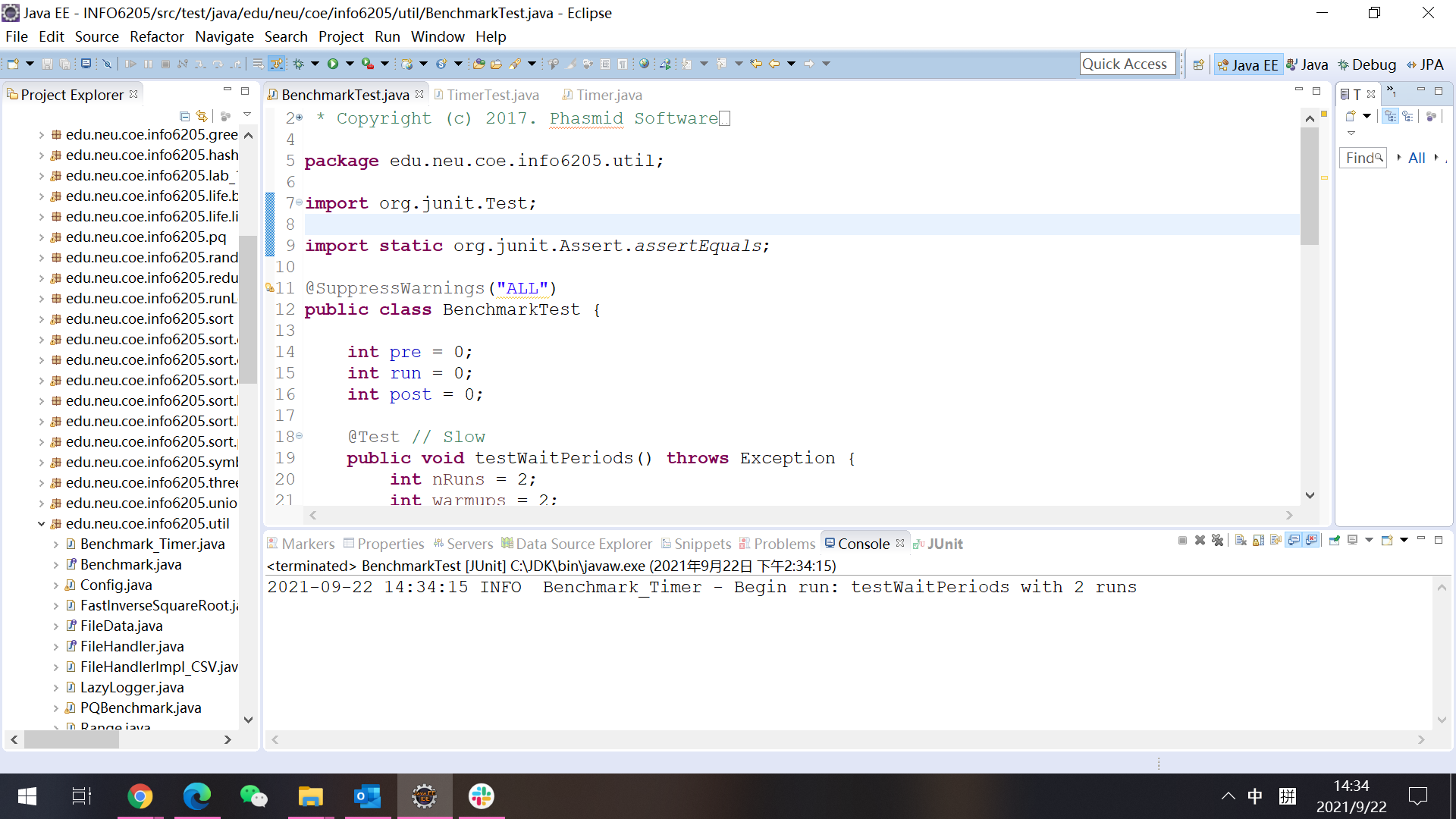
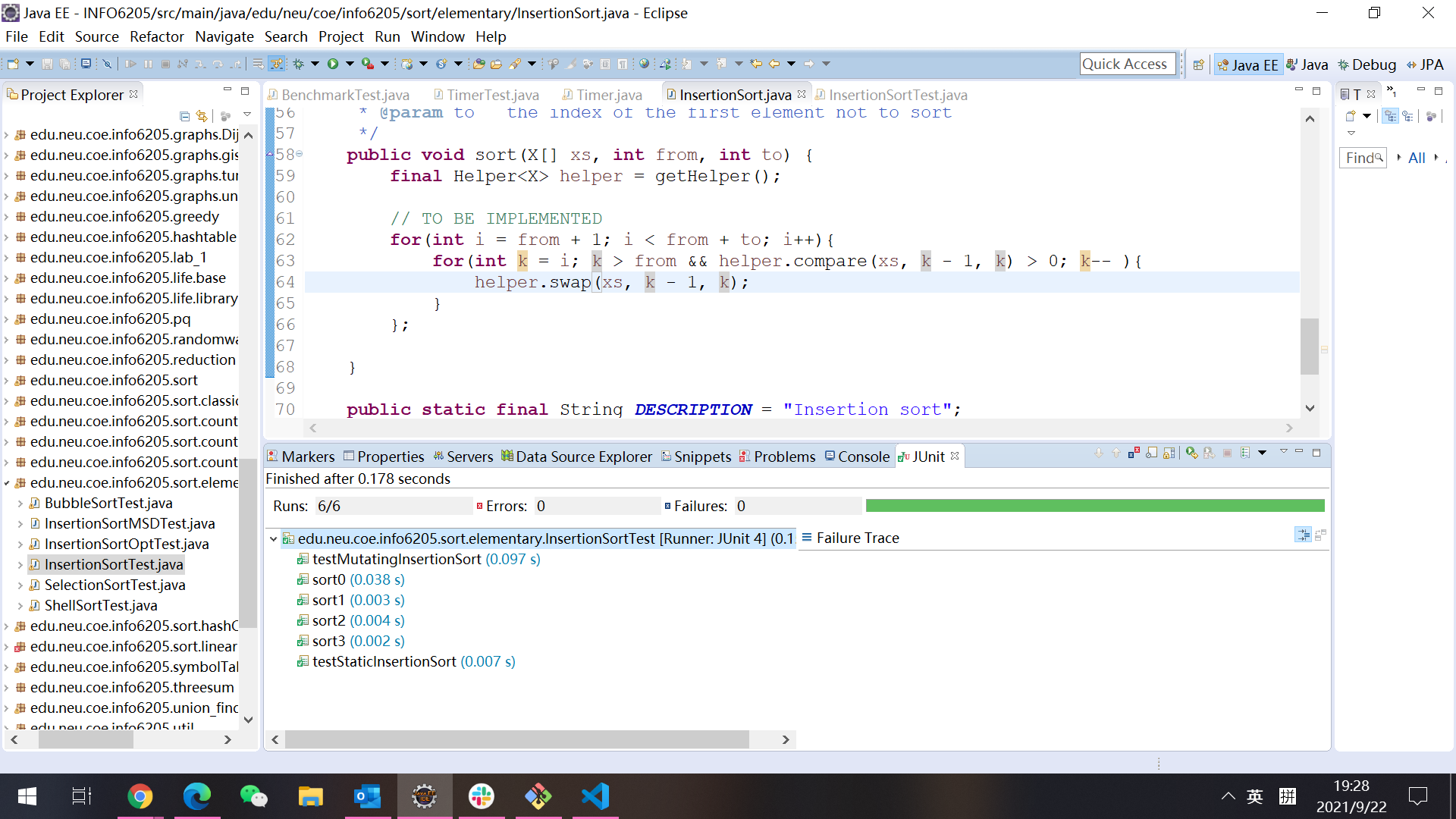
The order of run time of insertion sort for different order situation in arrays are:

**Ordered < Partially Ordered < Randomly Ordered < Reverse Ordered**

* + **Evidence to support the conclusion:**

1. **Output (Snapshot of Code output in the terminal)**
2. **Graphical Representation** **(Observations from experiments should be tabulated and analyzed by plotting graphs(usually in excel) to arrive on the relationship conclusion)**

I put all the output into the Excel, then put the running time and eight different values of N to drawn in a broken line diagram, from which can clearly see the speed of these four sorts in running time.

* + **Unit tests result:(Snapshot of successful unit test run)**
  + TimerTest
  + BenchmarkTest
  + InsertionSortTest