**INFO 6205**

**Program Structures & Algorithms**

**Fall 2020**

**Assignment No 2**

* **Task**

Work with benchmark timer, by measuring sorting arrays to draw conclusions from the algorithms running.

* **Output**

2020-09-28 22:14:02 INFO Benchmark\_Timer - Begin run: Insertion sort with 10 runs

2020-09-28 22:14:04 INFO Benchmark\_Timer - Begin run: Insertion sort with 10 runs

2020-09-28 22:14:04 INFO Benchmark\_Timer - Begin run: Insertion sort with 10 runs

2020-09-28 22:14:06 INFO Benchmark\_Timer - Begin run: Insertion sort with 10 runs

Average time of 10 times of insertion sort (random) array of 5000 numbers in 125.1 ms

Average time of 10 times of insertion sort (ordered) array of 5000 numbers in 53.9 ms

Average time of 10 times of insertion sort (partially-ordered) array of 5000 numbers in 116.2 ms

Average time of 10 times of insertion sort (reverse) array of 5000 numbers in 40.4 ms

2020-09-28 22:14:06 INFO Benchmark\_Timer - Begin run: Insertion sort with 20 runs

2020-09-28 22:14:09 INFO Benchmark\_Timer - Begin run: Insertion sort with 20 runs

2020-09-28 22:14:10 INFO Benchmark\_Timer - Begin run: Insertion sort with 20 runs

2020-09-28 22:14:12 INFO Benchmark\_Timer - Begin run: Insertion sort with 20 runs

Average time of 20 times of insertion sort (random) array of 5000 numbers in 110.6 ms

Average time of 20 times of insertion sort (ordered) array of 5000 numbers in 40.7 ms

Average time of 20 times of insertion sort (partially-ordered) array of 5000 numbers in 112.65 ms

Average time of 20 times of insertion sort (reverse) array of 5000 numbers in 43.75 ms

2020-09-28 22:14:13 INFO Benchmark\_Timer - Begin run: Insertion sort with 40 runs

2020-09-28 22:14:18 INFO Benchmark\_Timer - Begin run: Insertion sort with 40 runs

2020-09-28 22:14:20 INFO Benchmark\_Timer - Begin run: Insertion sort with 40 runs

2020-09-28 22:14:25 INFO Benchmark\_Timer - Begin run: Insertion sort with 40 runs

Average time of 40 times of insertion sort (random) array of 5000 numbers in 111.1 ms

Average time of 40 times of insertion sort (ordered) array of 5000 numbers in 43.175 ms

Average time of 40 times of insertion sort (partially-ordered) array of 5000 numbers in 109.5 ms

Average time of 40 times of insertion sort (reverse) array of 5000 numbers in 61.525 ms

2020-09-28 22:14:27 INFO Benchmark\_Timer - Begin run: Insertion sort with 80 runs

2020-09-28 22:14:38 INFO Benchmark\_Timer - Begin run: Insertion sort with 80 runs

2020-09-28 22:14:43 INFO Benchmark\_Timer - Begin run: Insertion sort with 80 runs

2020-09-28 22:14:53 INFO Benchmark\_Timer - Begin run: Insertion sort with 80 runs

Average time of 80 times of insertion sort (random) array of 5000 numbers in 117.4125 ms

Average time of 80 times of insertion sort (ordered) array of 5000 numbers in 51.8625 ms

Average time of 80 times of insertion sort (partially-ordered) array of 5000 numbers in 117.425 ms

Average time of 80 times of insertion sort (reverse) array of 5000 numbers in 44.075 ms

2020-09-28 22:14:57 INFO Benchmark\_Timer - Begin run: Insertion sort with 160 runs

2020-09-28 22:15:16 INFO Benchmark\_Timer - Begin run: Insertion sort with 160 runs

2020-09-28 22:15:23 INFO Benchmark\_Timer - Begin run: Insertion sort with 160 runs

2020-09-28 22:15:42 INFO Benchmark\_Timer - Begin run: Insertion sort with 160 runs

Average time of 160 times of insertion sort (random) array of 5000 numbers in 111.15625 ms

Average time of 160 times of insertion sort (ordered) array of 5000 numbers in 42.14375 ms

Average time of 160 times of insertion sort (partially-ordered) array of 5000 numbers in 109.66875 ms

Average time of 160 times of insertion sort (reverse) array of 5000 numbers in 42.01875 ms

* **Conclusion**

When doing insertion sort, the running time of sort random or partially ordered array is longer, then the ordered array and reverse ordered array use less time to be sorted.

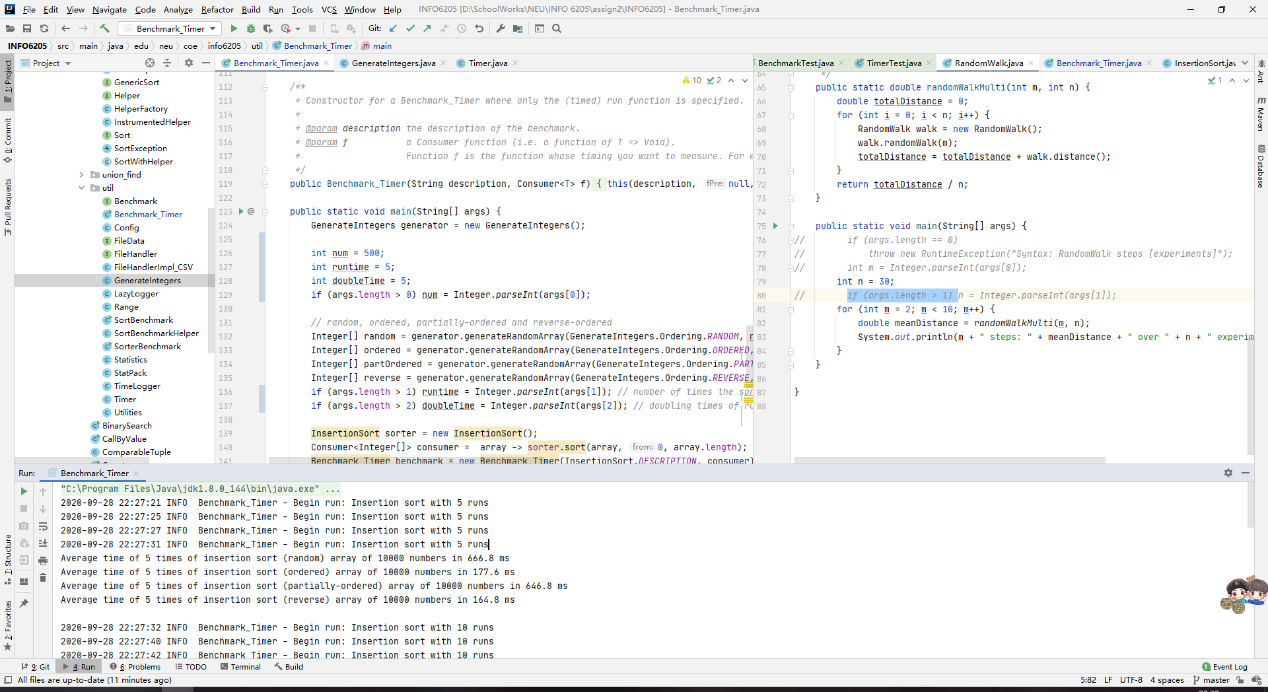
For random and partially ordered arrays, sometimes, it takes longer for sorting random. There is also opposite situation appears.

So are ordered and reverse arrays.

When the array is larger, the result of timing looks more stable in the same ordering situation of the array.

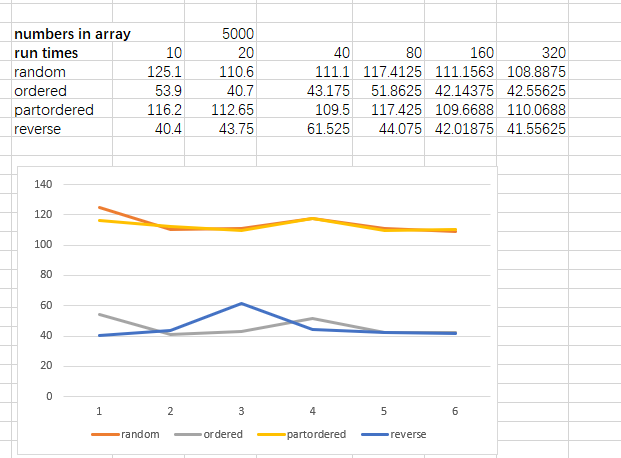
* **Evidence to support**

Screenshot of running the measuring method of insertion sorting (implement in Benchmark\_Timer as main):

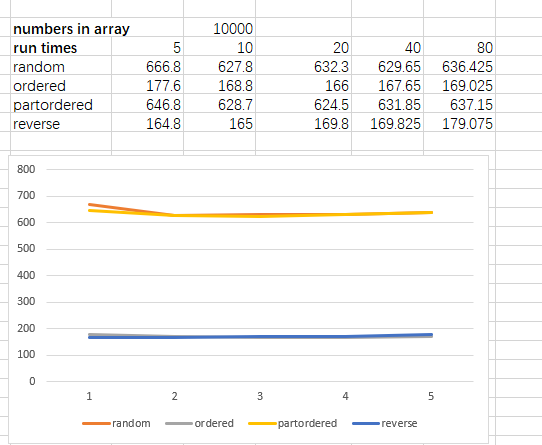


Spreadsheet locate at /report/asssign2/YueFang\_AssignmentNo2.xlsx

Sort array of 5000 numbers in four ordering (timing in ms):

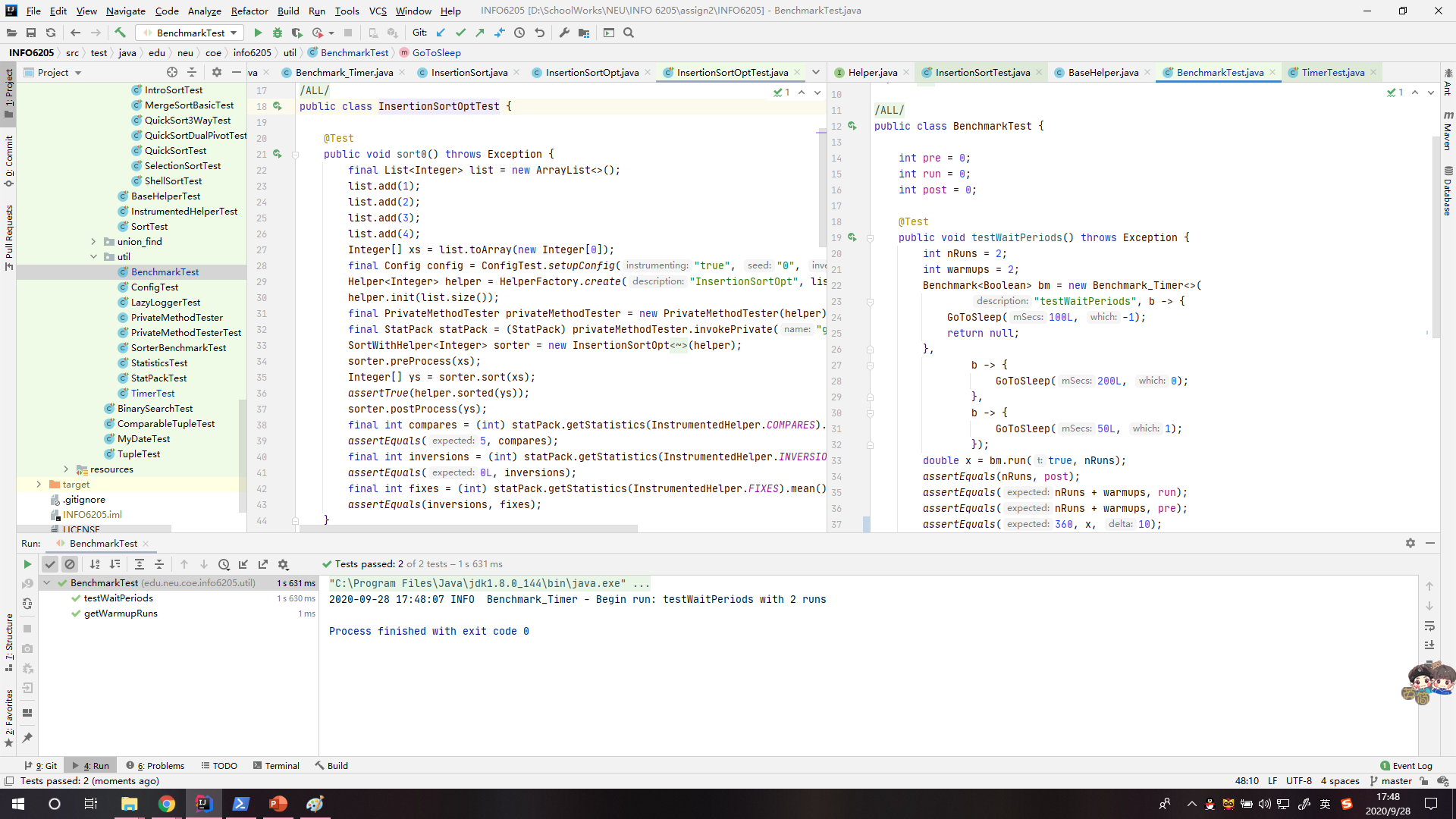


Sort array of 5000 numbers in four ordering (timing in ms):

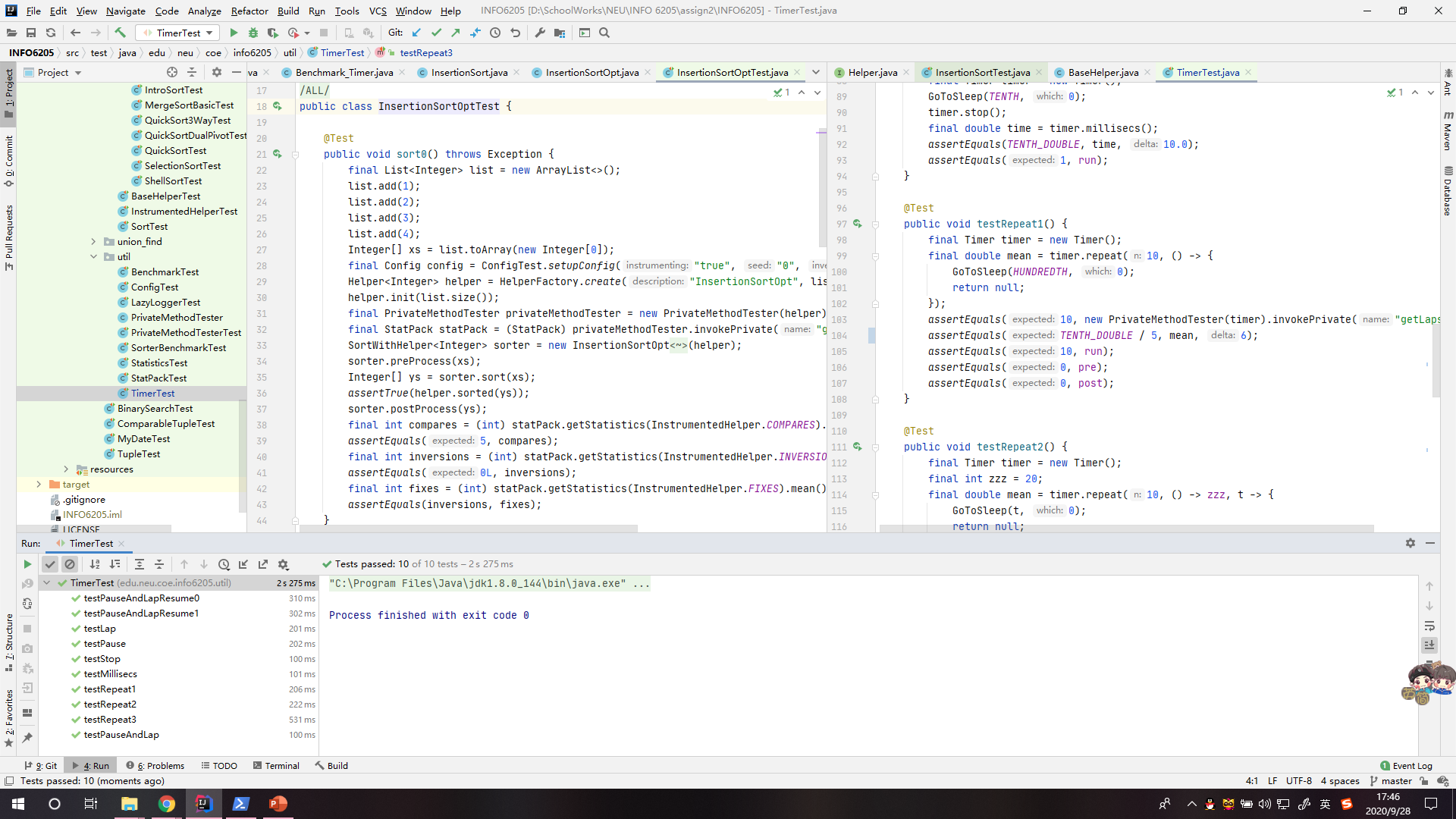


* **Screenshot of Unit test passing**

BenchmarkTest:



TimerTest:



InsertionSortTest:

