**Sai Saketh Kosuri (002199401)**

**Program Structures & Algorithms Fall 202**

**Assignment No. 02**

**Task:**

1. Implement 3 methods in Timer class and to run the unit tests in TimerTest and BenchmarkTest
2. Implement insertion sort and verify the implementation by running unit tests in InsertionSortTest.
3. Implement a main method to run the following benchmarks:
   1. Random array generation
   2. Sorted array generation
   3. Reverse Ordered array generation
   4. Partially Ordered array generation

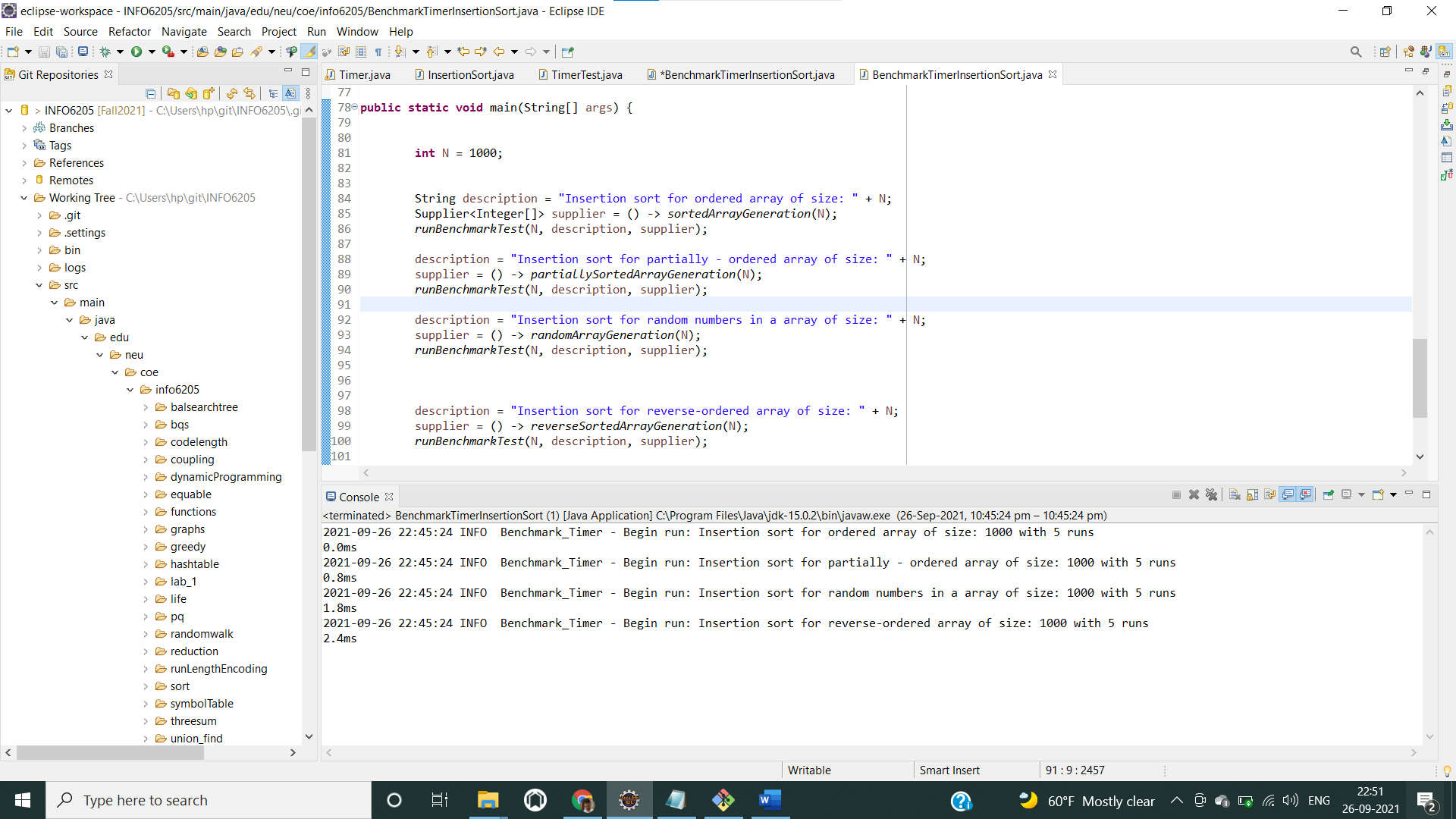
Also use the doubling method for choosing ‘n’ and test for at least five values of ‘n’

**Relationship Conclusion for Task 3:**

From the observations of the graph i.e. relationship between N and T for different ordered arrays, following conclusions are drawn:

1. reverse ordered array elements took maximum amount of time for sorting as it is worst case. Hence the time complexity is O(N^2) which is a quadratic.
2. Best case scenario corresponds to the ordered array elements with complexity of 0(N).
3. Partially ordered array elements corresponds to the time complexity is O(N log(N)) which is a linearithmic.

Output for Benchmark Timer Insertion sort:



For 1000 Array input size:

Benchmark\_Timer - Begin run: Insertion sort for ordered array of size: 1000 with 5 runs

0.0ms

Benchmark\_Timer - Begin run: Insertion sort for partially - ordered array of size: 1000 with 5 runs

0.8ms

Benchmark\_Timer - Begin run: Insertion sort for random numbers in a array of size: 1000 with 5 runs

0.8ms

Benchmark\_Timer - Begin run: Insertion sort for reverse-ordered array of size: 1000 with 5 runs

1.6ms

For 10000 array input size:

Benchmark\_Timer - Begin run: Insertion sort for ordered array of size: 10000 with 5 runs

0.8ms

Benchmark\_Timer - Begin run: Insertion sort for partially - ordered array of size: 10000 with 5 runs

22.6ms

Benchmark\_Timer - Begin run: Insertion sort for random numbers in a array of size: 10000 with 5 runs

81.6ms

Benchmark\_Timer - Begin run: Insertion sort for reverse-ordered array of size: 10000 with 5 runs

148.6ms

For 5000 array input size:

Benchmark\_Timer - Begin run: Insertion sort for ordered array of size: 5000 with 5 runs

0.2ms

Benchmark\_Timer - Begin run: Insertion sort for partially - ordered array of size: 5000 with 5 runs

5.2ms

Benchmark\_Timer - Begin run: Insertion sort for random numbers in a array of size: 5000 with 5 runs

19.8ms

Benchmark\_Timer - Begin run: Insertion sort for reverse-ordered array of size: 5000 with 5 runs

35.8ms

For 20000 array input sze:

2021-09-26 23:49:01 INFO Benchmark\_Timer - Begin run: Insertion sort for ordered array of size: 5000 with 5 runs

0.2ms

2021-09-26 23:49:01 INFO Benchmark\_Timer - Begin run: Insertion sort for partially - ordered array of size: 5000 with 5 runs

5.2ms

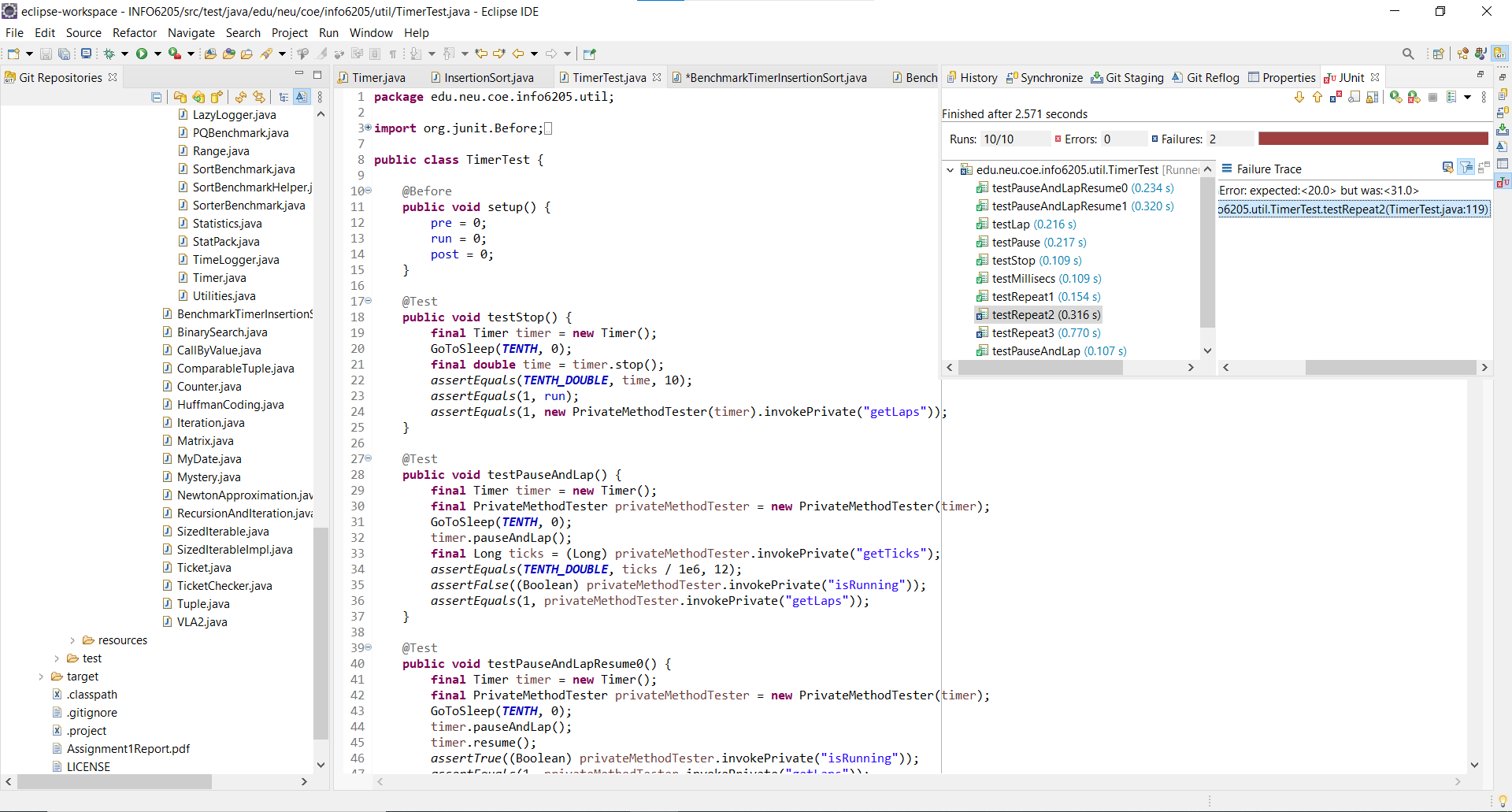
2021-09-26 23:49:01 INFO Benchmark\_Timer - Begin run: Insertion sort for random numbers in a array of size: 5000 with 5 runs

19.8ms

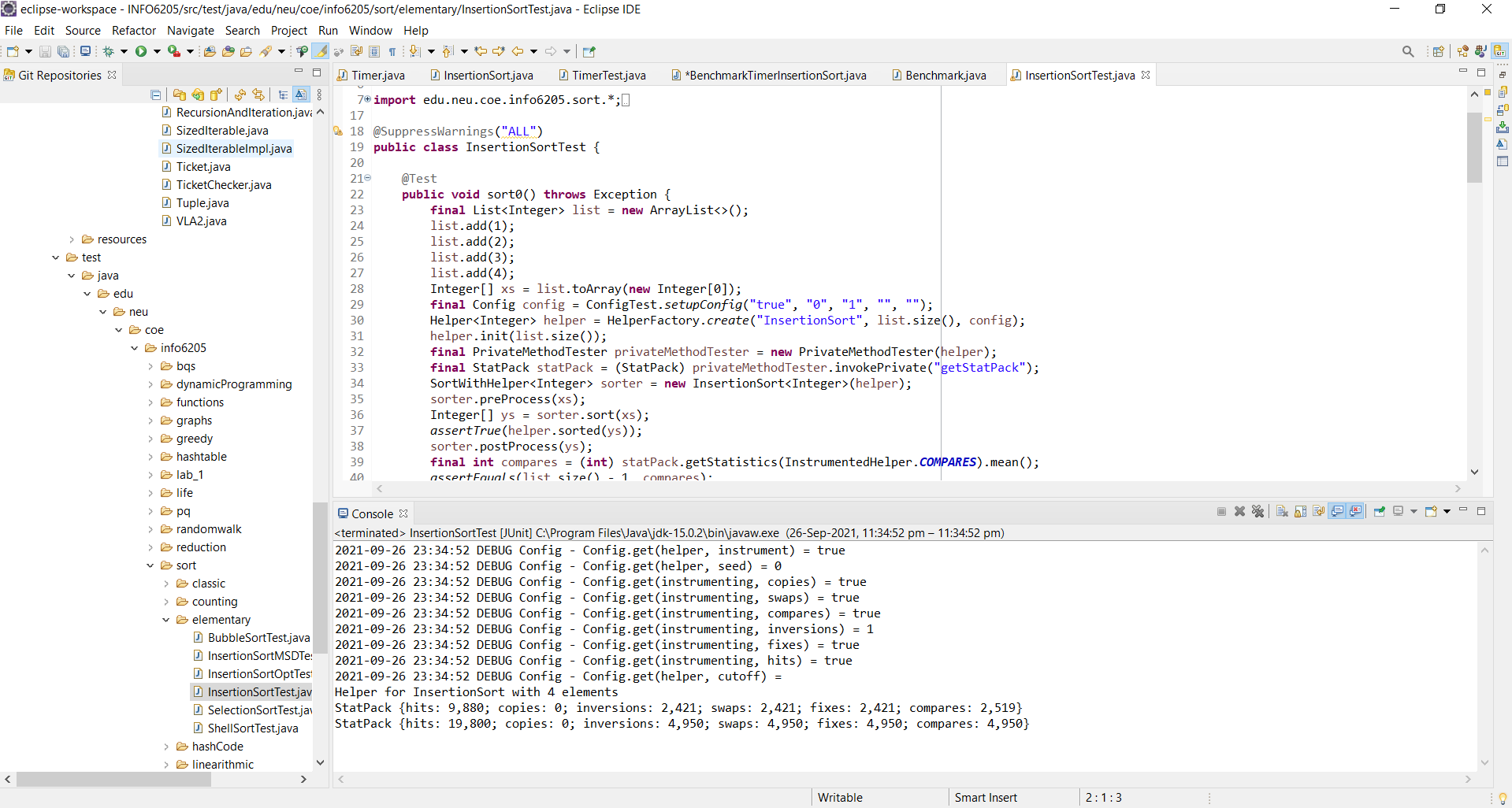
2021-09-26 23:49:01 INFO Benchmark\_Timer - Begin run: Insertion sort for reverse-ordered array of size: 5000 with 5 runs

35.8ms

Output for Timer Test java:



Insertion Sort test:



Helper for InsertionSort with 4 elements

StatPack {hits: 9,880; copies: 0; inversions: 2,421; swaps: 2,421; fixes: 2,421; compares: 2,519}

StatPack {hits: 19,800; copies: 0; inversions: 4,950; swaps: 4,950; fixes: 4,950; compares: 4,950}