Vedant Jain NUID: 001305281

INFO 6205

Program Structures & Algorithms

Fall 2020

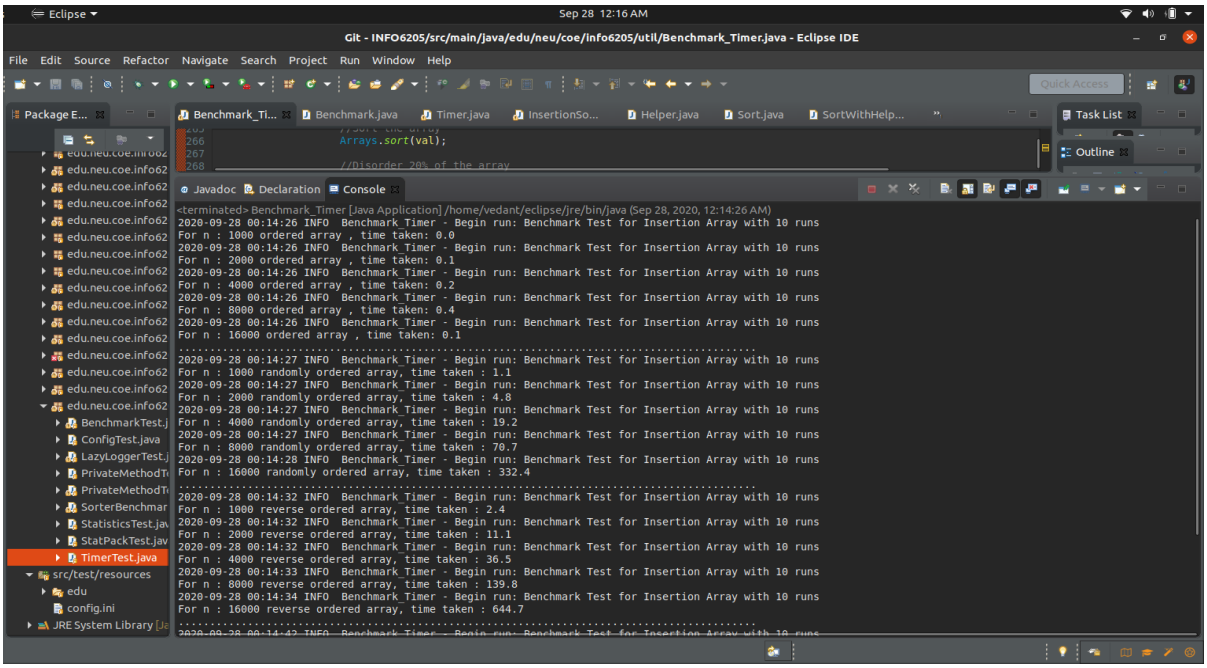
Assignment No: 2

- **Task:**

  1) To implement the class Timer.java and Benchmark_Timer.java class
  2) Complete the implementation of Insertion sort in InsertionSort.java class
  3) Measure the running time complexity of different array ordering situations - Random, Ordered, Partially-Ordered and Reverse-ordered array.

- **Output:**

  Carried out benchmark test for 5 different values of n starting from 1000 and getting doubled for each n fulfilling all 4 initial array ordering conditions



Console Output-

```
2020-09-28 00:14:26 INFO  Benchmark_Timer - Begin run: Benchmark Test for Insertion Array with 10 runs
For n : 1000 ordered array , time taken: 0.0

2020-09-28 00:14:26 INFO  Benchmark_Timer - Begin run: Benchmark Test for Insertion Array with 10 runs

For n : 2000 ordered array , time taken: 0.1

2020-09-28 00:14:26 INFO  Benchmark_Timer - Begin run: Benchmark Test for Insertion Array with 10 runs

For n : 4000 ordered array , time taken: 0.2

2020-09-28 00:14:26 INFO  Benchmark_Timer - Begin run: Benchmark Test for Insertion Array with 10 runs

For n : 8000 ordered array , time taken: 0.4

2020-09-28 00:14:26 INFO  Benchmark_Timer - Begin run: Benchmark Test for Insertion Array with 10 runs

For n : 16000 ordered array , time taken: 0.8

.....................................................................................

2020-09-28 00:14:27 INFO  Benchmark_Timer - Begin run: Benchmark Test for Insertion Array with 10 runs

For n : 1000 randomly ordered array, time taken : 1.1

2020-09-28 00:14:27 INFO  Benchmark_Timer - Begin run: Benchmark Test for Insertion Array with 10 runs

For n : 2000 randomly ordered array, time taken : 4.8

2020-09-28 00:14:27 INFO  Benchmark_Timer - Begin run: Benchmark Test for Insertion Array with 10 runs

For n : 4000 randomly ordered array, time taken : 19.2

2020-09-28 00:14:27 INFO  Benchmark_Timer - Begin run: Benchmark Test for Insertion Array with 10 runs

For n : 8000 randomly ordered array, time taken : 70.7

2020-09-28 00:14:28 INFO  Benchmark_Timer - Begin run: Benchmark Test for Insertion Array with 10 runs

For n : 16000 randomly ordered array, time taken : 332.4
```

```
.....................................................................................
2020-09-28 00:14:32 INFO  Benchmark_Timer - Begin run: Benchmark Test for Insertion Array with 10 runs
For n : 1000 reverse ordered array, time taken : 2.4
2020-09-28 00:14:32 INFO  Benchmark_Timer - Begin run: Benchmark Test for Insertion Array with 10 runs
For n : 2000 reverse ordered array, time taken : 11.1
2020-09-28 00:14:32 INFO  Benchmark_Timer - Begin run: Benchmark Test for Insertion Array with 10 runs
For n : 4000 reverse ordered array, time taken : 36.5
2020-09-28 00:14:33 INFO  Benchmark_Timer - Begin run: Benchmark Test for Insertion Array with 10 runs
For n : 8000 reverse ordered array, time taken : 139.8
2020-09-28 00:14:34 INFO  Benchmark_Timer - Begin run: Benchmark Test for Insertion Array with 10 runs
For n : 16000 reverse ordered array, time taken : 644.7
.....................................................................................
2020-09-28 00:14:42 INFO  Benchmark_Timer - Begin run: Benchmark Test for Insertion Array with 10 runs
For n: 1000 partially ordered array, time taken: 0.2
2020-09-28 00:14:42 INFO  Benchmark_Timer - Begin run: Benchmark Test for Insertion Array with 10 runs
For n: 2000 partially ordered array, time taken: 1.2
2020-09-28 00:14:42 INFO  Benchmark_Timer - Begin run: Benchmark Test for Insertion Array with 10 runs
For n: 4000 partially ordered array, time taken: 3.7
2020-09-28 00:14:42 INFO  Benchmark_Timer - Begin run: Benchmark Test for Insertion Array with 10 runs
For n: 8000 partially ordered array, time taken: 14.6
2020-09-28 00:14:42 INFO  Benchmark_Timer - Begin run: Benchmark Test for Insertion Array with 10 runs
For n: 16000 partially ordered array, time taken: 74.7
```

- **Relationship conclusion:**

1) When the elements are doubled , we can see that time taken by algorithm increases in a quadratic sense with average time taken of $O(n^2)$.

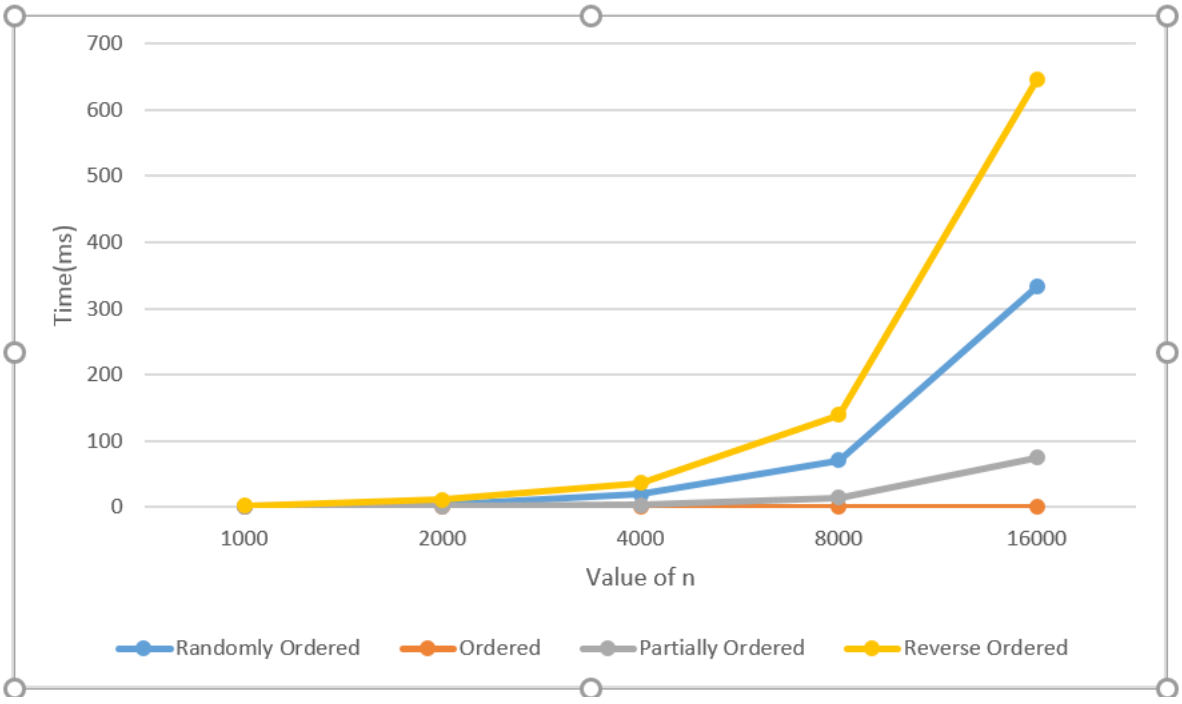
2) Insertion sort works fastest for the ordered arrays , best case time complexity in this case is O(n) as it does not perform any swap.

3) Meanwhile the worst case time complexity of $O(n^2)$ is where it performs the maximum swaps. Also the partially ordered array is relatively better but also performs many swaps with time complexity of $O(n^2)$.

- **Evidence to support relationship**:

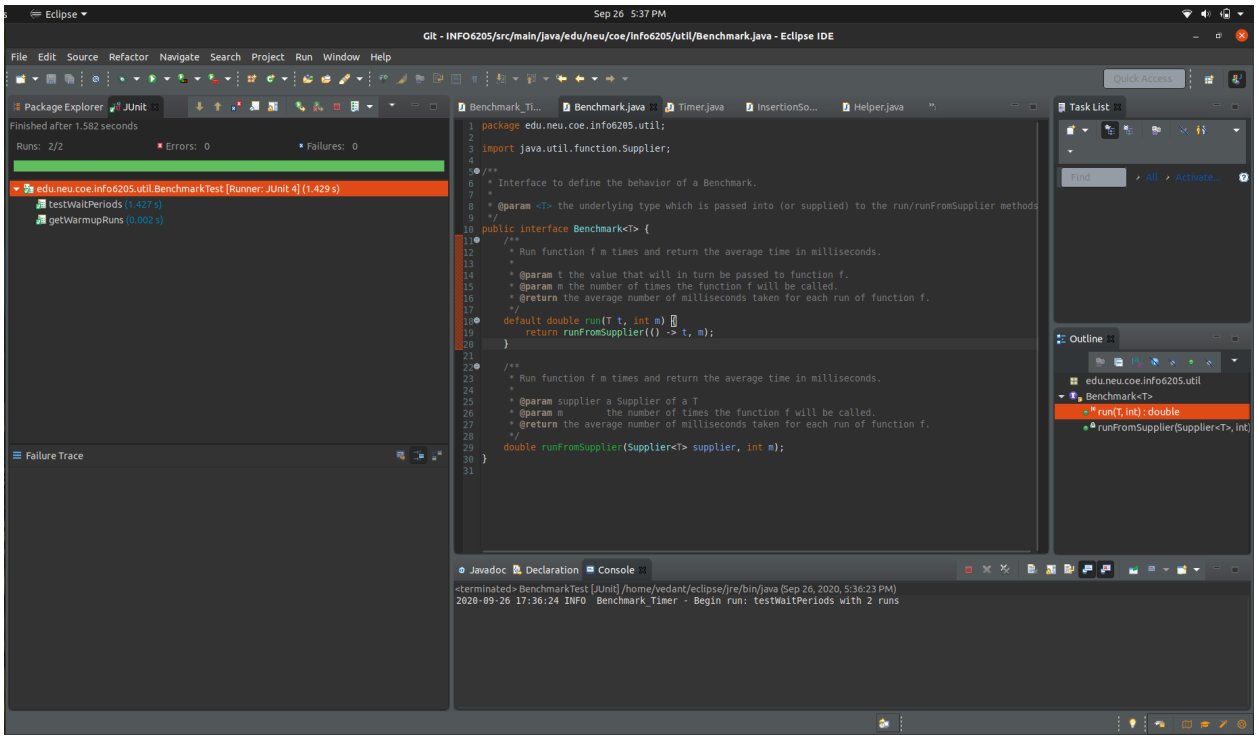
Vedant Jain NUID: 001305281

Value of n	Randomly Ordered(ms)	Ordered(ms)	Partially Ordered(ms)	Reverse Ordered(ms)
1000	1.1	0.0	0.2	2.4
2000	4.8	0.1	1.2	11.1
4000	19.2	0.2	3.7	36.5
8000	70.7	0.4	14.6	139.8
16000	332.4	0.8	74.7	644.7



- **Screenshot of Unit test passing:** Below is the screenshot of all the unit tests which ran successfully

Benchmark Test



Vedant Jain NUID: 001305281

TimerTest