Narendra Raj R K (001553969)

Program Structures & Algorithms

Spring 21

Assignment No. 2

Tasks:

Part 1: In this task I am to implement three methods of a class called Timer - **<T, U> double repeat, getClock() & toMillisecs(long ticks)**.

- ⇨ Which I was successfully able to implement as all the test cases under **TimerTest** and **BenchmarkTest** have passed.

Part 2: Implement **Insertion Sort** under the sort() method available under the InsertionSort.java file only with the help of methods already defined under BaseHelper.java file**.**

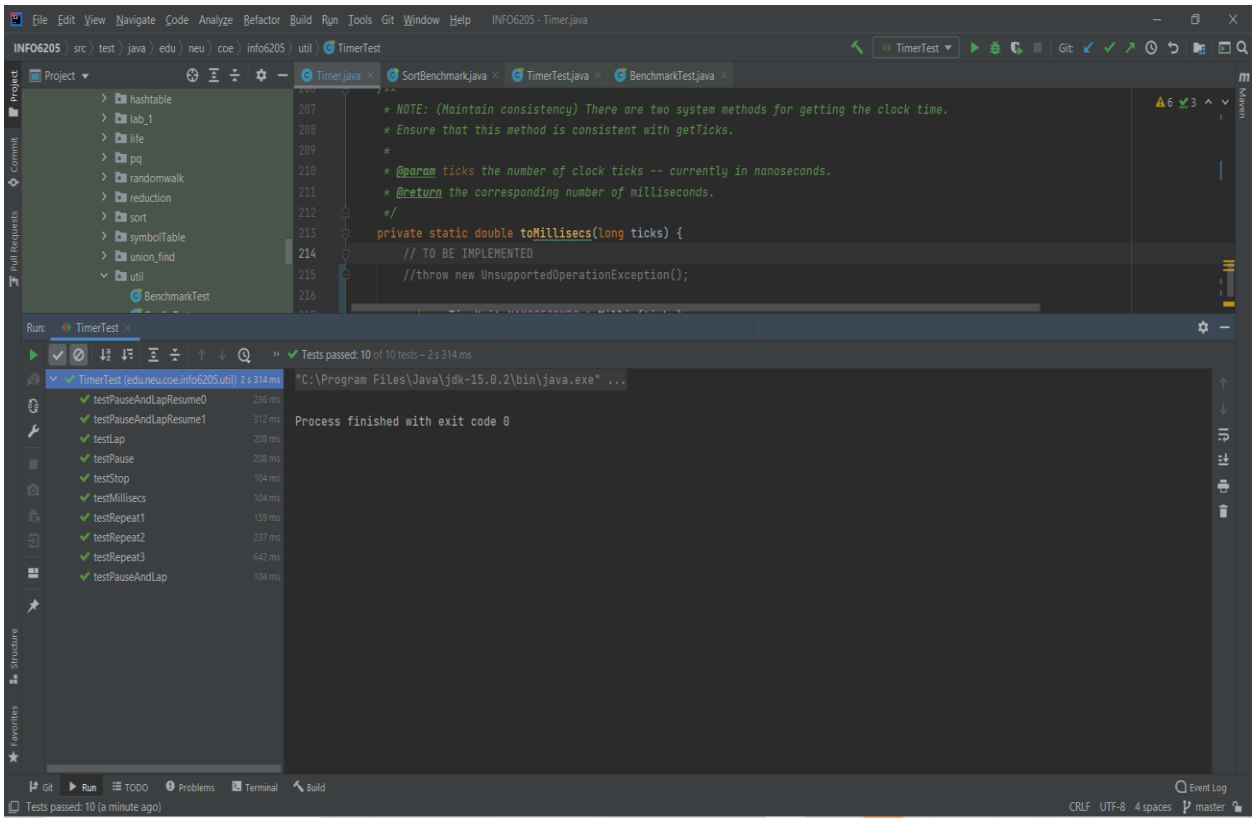
- ⇨ Successfully implemented the Insertion sort logic only using the methods available under BaseHelper file.

Part 3: Implement a **main method in Benchmark_Timer.java** to benchmark the Insertion sort algorithm against different array inputs.

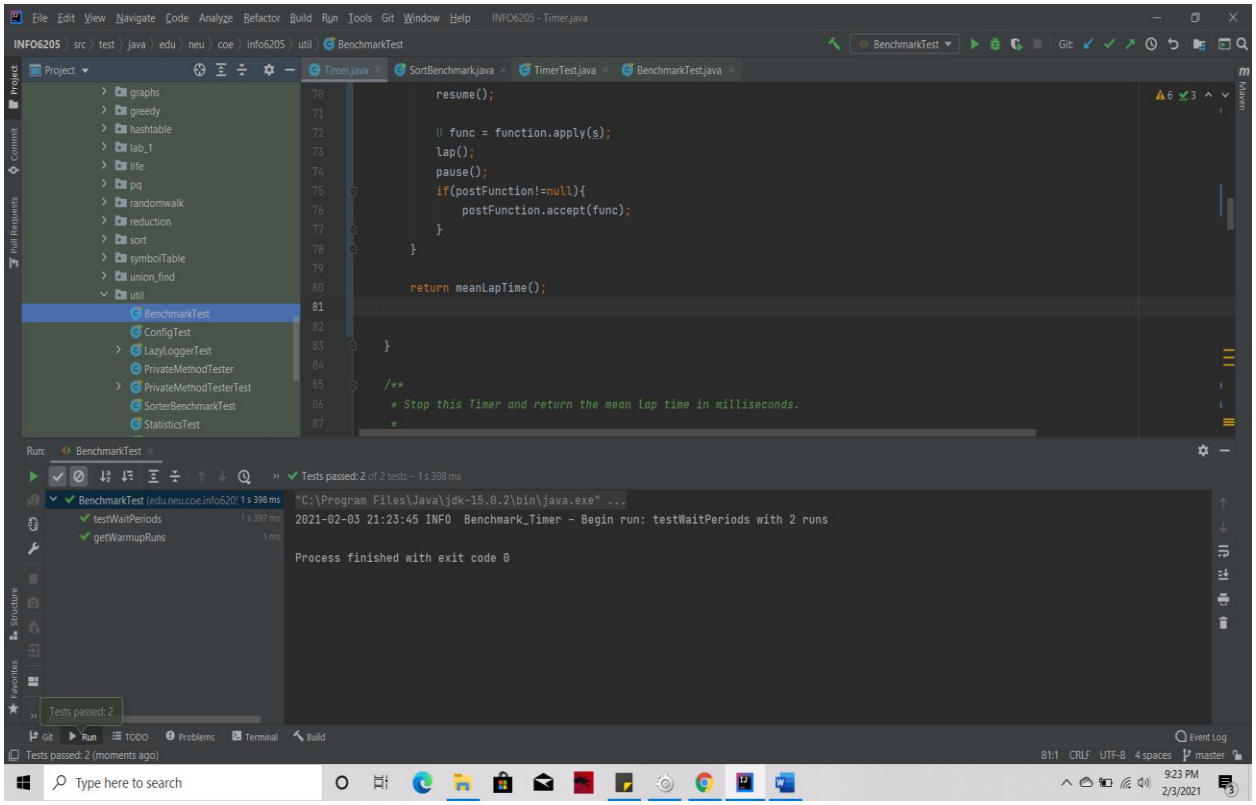
⇨ Implemented a main method to successfully benchmark against four distinctly sorted arrays of type Integer (ordered, random, partial & reversed).

Unit Test Results:

i) TimerTest.java (All 10 test cases passed)



ii) BenchmarkTest.java (All 2 test cases passed)



iii) InsertionSortTest.java (All 4 test cases passed)

Evidence to Support Conclusion:

I have created four different arrays as per the requirement of this assignment,

1. Sorted Array
2. Reversed Array
3. Random Array
4. Partial Array

Some useful Abbreviations

- n – Number of elements in the Array

I ran the experiment for various “n” values like 1000 & 10000.

- m – number of repetitions to arrive at the mean time.

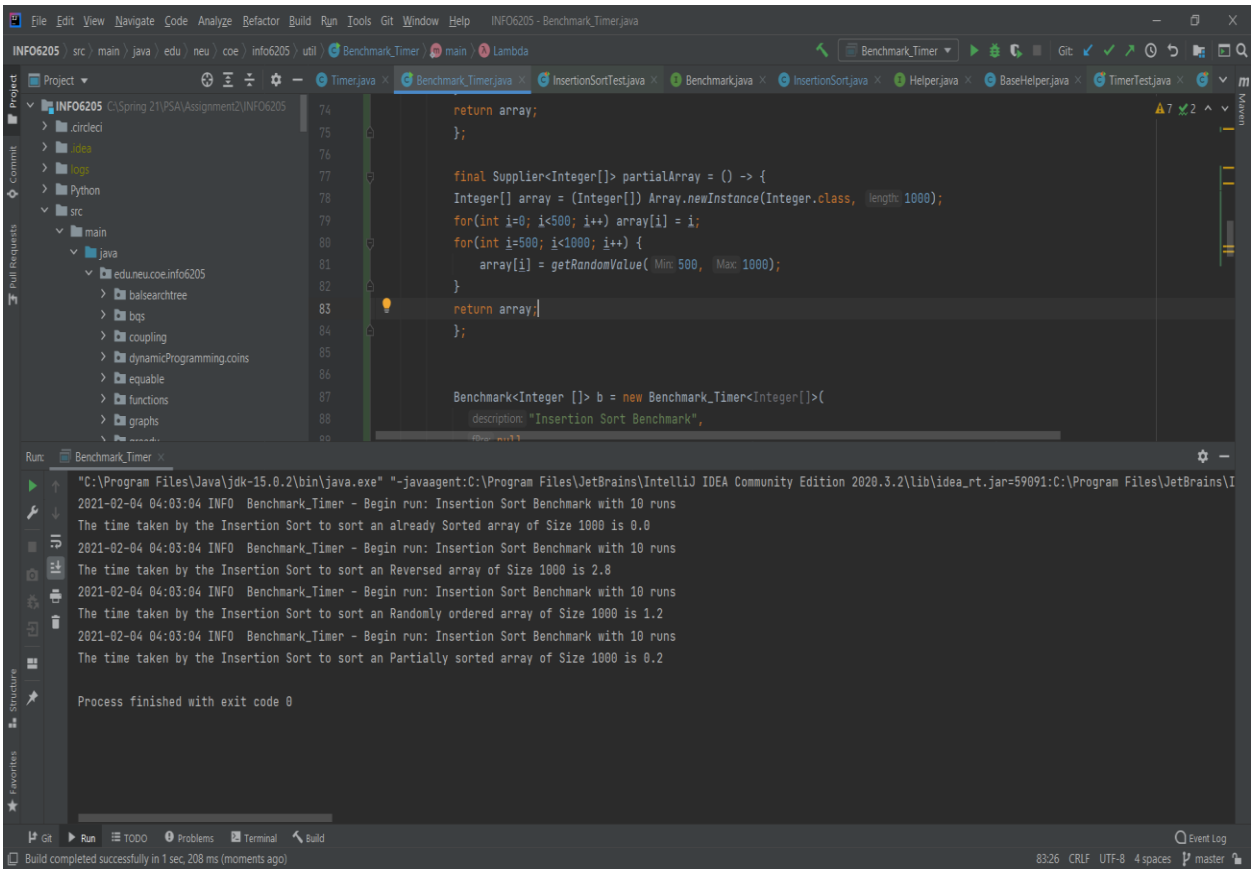
Ran the experiment for different values of m like 10,50 & 100.

Please find observations, screenshots & graphs etc., supporting the conclusion

Run 1: m – 10 ; n – 1000 (10 Reps & 1000 elements in the array)

Type of Array	M – Repetitions	N – No. of Elements	Mean Time (Milliseconds)
Sorted	10	1000	**0.0**
Reversed	10	1000	**2.8**
Random	10	1000	**1.2**
Partial	10	1000	**0.2**

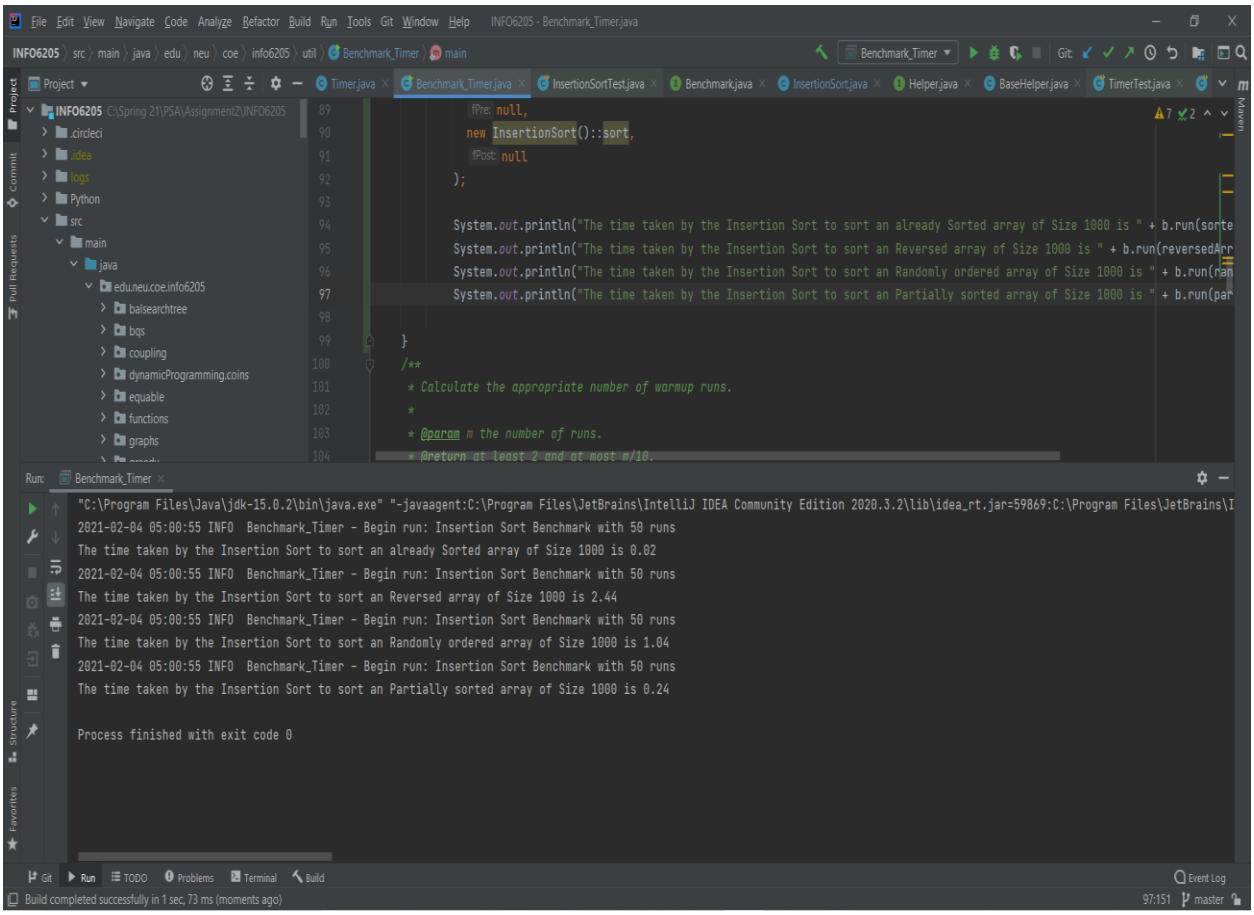
Screenshot of the Output



Run 2: **m – 50 ; n – 1000** (50 Reps & 1000 elements in the array)

Type of Array	M – Repetitions	N – No. of Elements	Mean Time (Milliseconds)
Sorted	50	1000	**0.02**
Reversed	50	1000	**2.44**
Random	50	1000	**1.04**
Partial	50	1000	**0.24**

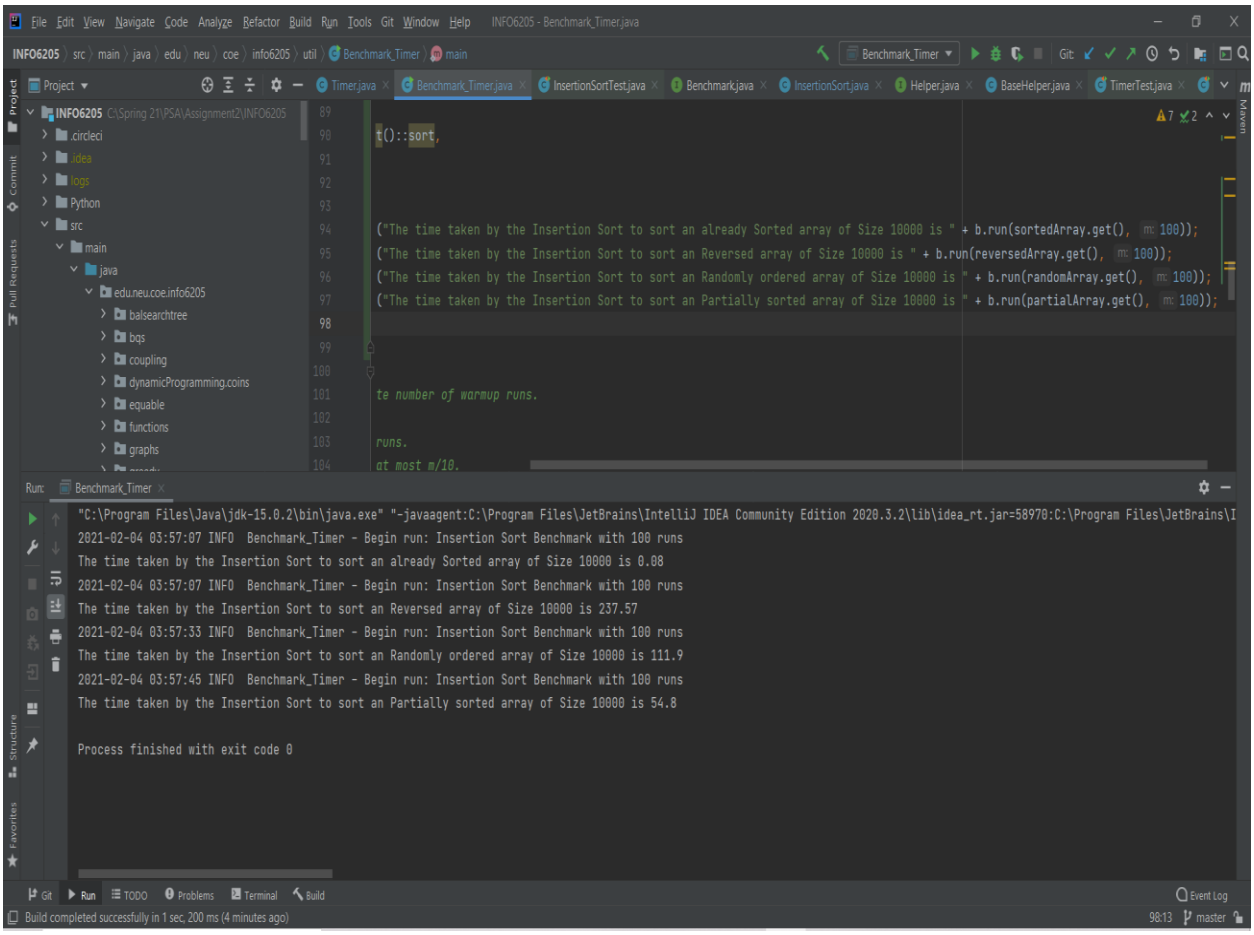
Screenshot of the Output



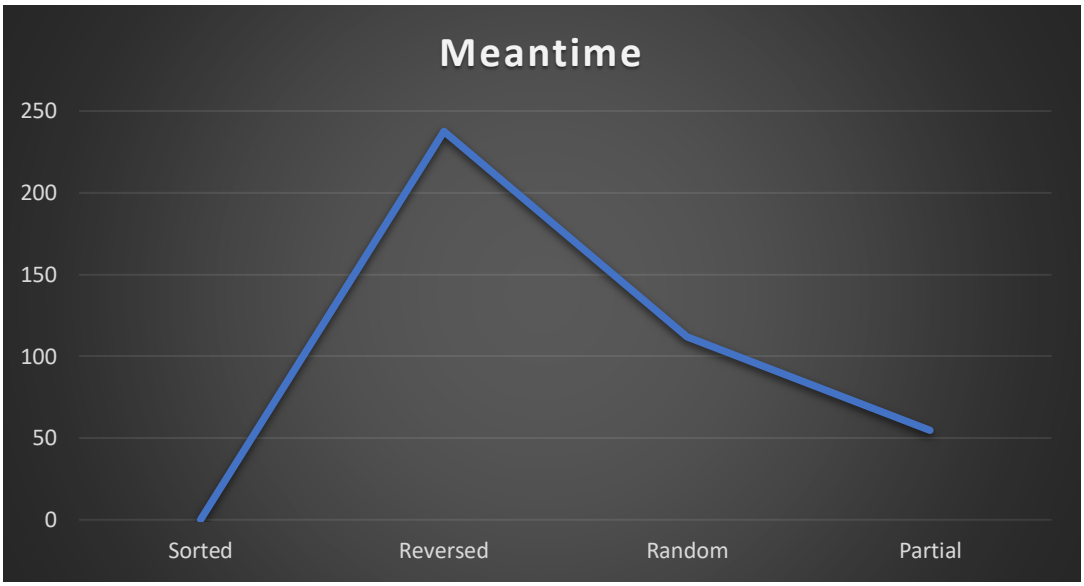
Run 3: **m – 100 ; n – 10000** (100 Reps & 10000 elements in the array)

Type of Array	M – Repetitions	N – No. of Elements	Mean Time (Milliseconds)
Sorted	100	10000	**0.08**
Reversed	100	10000	**237.57**
Random	100	10000	**111.9**
Partial	100	10000	**54.8**

Screenshot of the Output



Graphical Representation:



Final Conclusion:

- After benchmarking the Insertion Sort Algorithm against the different types of Arrays – Sorted, Reversed, Random & Partially ordered for multiple runs. I have arrived at the following conclusion, that time taken to sort an already sorted array is the least and to sort an reversed array consumes the most time.

Sorted Array < Partial Array < Random Array < Reversed Array