Program Structures & Algorithms
Spring 2022
Assignment No. 2

Name: Shashank Siripragada
NUID: 002193773

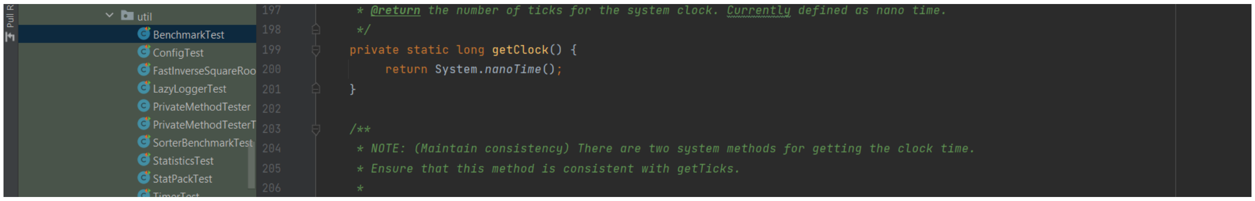
Task:

- (Part 1) You are to implement three (3) methods (*repeat*, *getClock*, and *toMillisecs*) of a class called *Timer*. Don't forget to check your implementation by running the unit tests in *BenchmarkTest* and *TimerTest*.
- (Part 2) Implement *InsertionSort* (in the *InsertionSort* class) by simply looking up the insertion code used by *Arrays.sort*. Either way, you must run the unit tests in *InsertionSortTest*.
- (Part 3) Implement the 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. Draw any conclusions from your observations regarding the order of growth.

Part1:

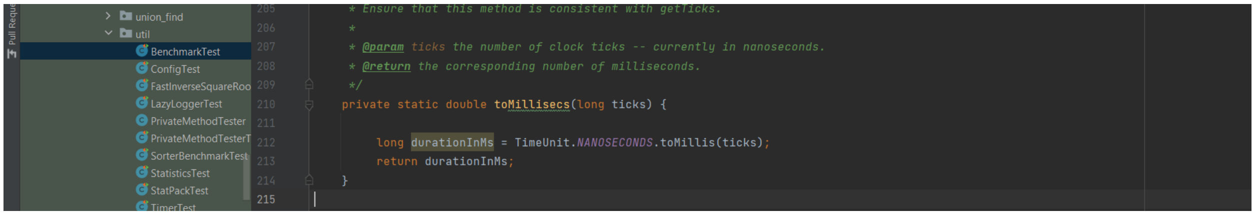
Implemented repeat, getClock, and toMillisecs in Timer Class.

Code Changes: getClock



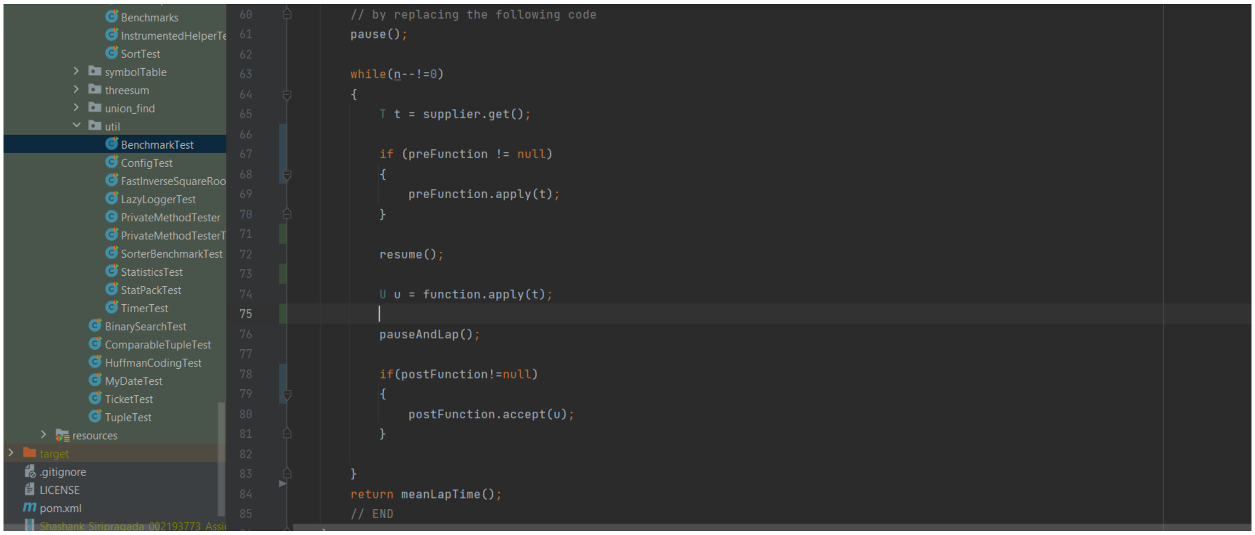
```
 * @return the number of ticks for the system clock. Currently defined as nano time.
 */
private static long getClock() {
    return System.nanoTime();
}

/**
 * NOTE: (Maintain consistency) There are two system methods for getting the clock time.
 * Ensure that this method is consistent with getTicks.
 *
```

Code Changes: toMillisecs



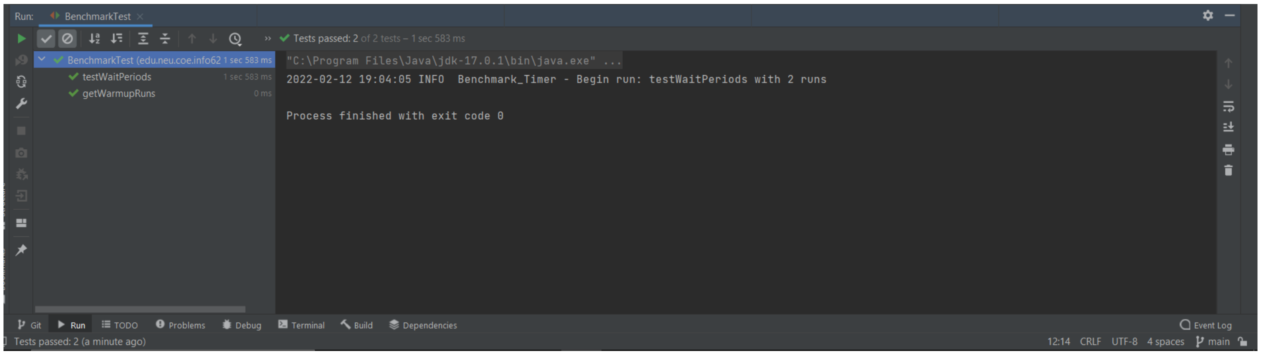
```
 * Ensure that this method is consistent with getTicks.
 *
 * @param ticks the number of clock ticks -- currently in nanoseconds.
 * @return the corresponding number of milliseconds.
 */
private static double toMillisecs(long ticks) {

    long durationInMs = TimeUnit.NANOSECONDS.toMillis(ticks);
    return durationInMs;
}

```

Code Changes: repeat



```java
        // by replacing the following code
        pause();

        while(n--!=0)
        {
            T t = supplier.get();

            if (preFunction != null)
            {
                preFunction.apply(t);
            }

            resume();

            U u = function.apply(t);

            pauseAndLap();

            if(postFunction!=null)
            {
                postFunction.accept(u);
            }

        }
        return meanLapTime();
        // END
```

Unit Test Results: BenchmarkTest



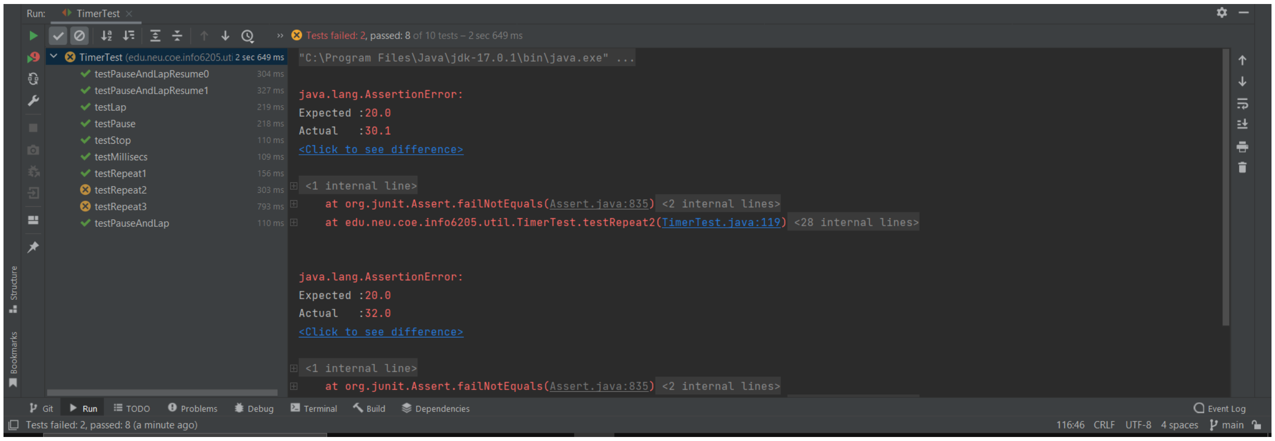
Tests passed: 2 of 2 tests – 1 sec 583 ms

```
✓ BenchmarkTest (edu.neu.coe.info62 1 sec 583 ms
    ✓ testWaitPeriods                1 sec 583 ms
    ✓ getWarmupRuns                         0 ms
```

```
"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" ...
2022-02-12 19:04:05 INFO  Benchmark_Timer - Begin run: testWaitPeriods with 2 runs

Process finished with exit code 0
```

Tests passed: 2 (a minute ago)

Unit Test Results: TimerTest



Tests failed: 2, passed: 8 of 10 tests – 2 sec 649 ms

```
✗ TimerTest (edu.neu.coe.info6205.uti 2 sec 649 ms
    ✓ testPauseAndLapResume0          304 ms
    ✓ testPauseAndLapResume1          327 ms
    ✓ testLap                         219 ms
    ✓ testPause                       218 ms
    ✓ testStop                        110 ms
    ✓ testMillisecs                   109 ms
    ✓ testRepeat1                     156 ms
    ✗ testRepeat2                     303 ms
    ✗ testRepeat3                     793 ms
    ✓ testPauseAndLap                 110 ms
```

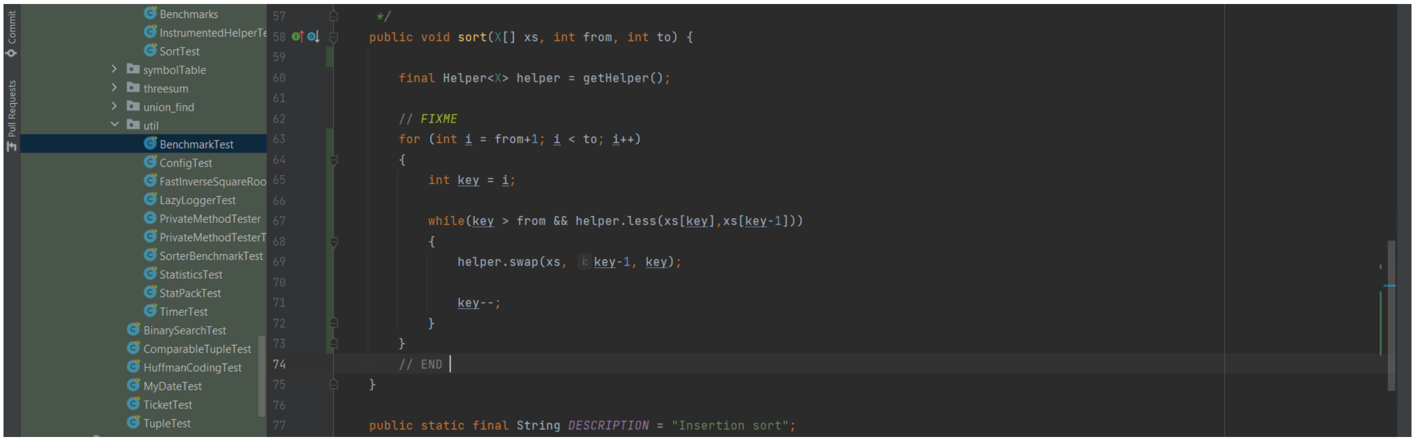
```
"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" ...

java.lang.AssertionError:
Expected :20.0
Actual   :30.1
<Click to see difference>

<1 internal line>
    at org.junit.Assert.failNotEquals(Assert.java:835) <2 internal lines>
    at edu.neu.coe.info6205.util.TimerTest.testRepeat2(TimerTest.java:119) <28 internal lines>


java.lang.AssertionError:
Expected :20.0
Actual   :32.0
<Click to see difference>

<1 internal line>
    at org.junit.Assert.failNotEquals(Assert.java:835) <2 internal lines>
```

Tests failed: 2, passed: 8 (a minute ago)

Part2:

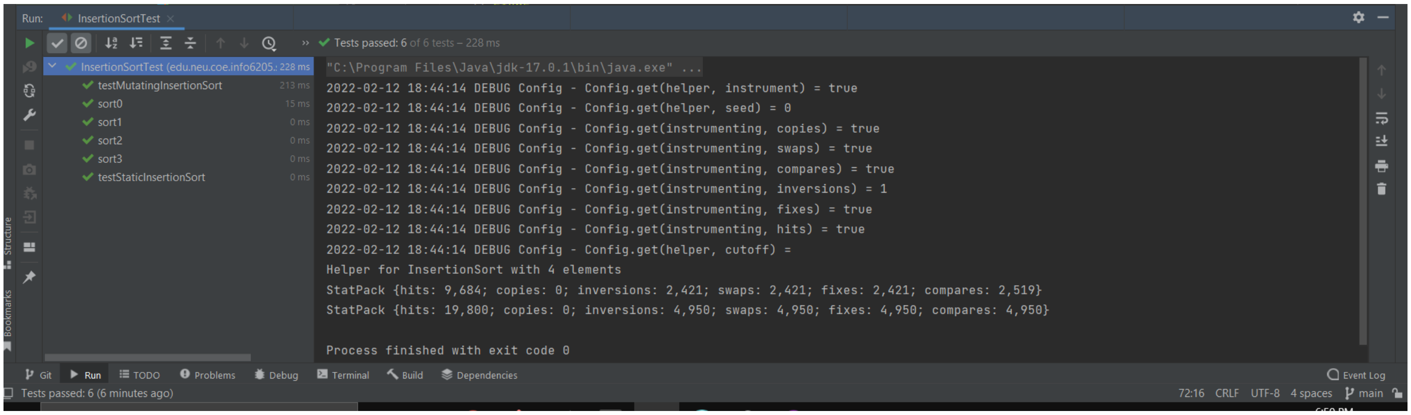
Filled in *InsertionSort* code. The test cases have successfully run.

Code Changes: InsertionSort



```java
     */
    public void sort(X[] xs, int from, int to) {

        final Helper<X> helper = getHelper();

        // FIXME
        for (int i = from+1; i < to; i++)
        {
            int key = i;

            while(key > from && helper.less(xs[key],xs[key-1]))
            {
                helper.swap(xs, key-1, key);

                key--;
            }
        }
        // END
    }

    public static final String DESCRIPTION = "Insertion sort";
```

UnitTest results: InsertionSort



Tests passed: 6 of 6 tests – 228 ms

```
InsertionSortTest (edu.neu.coe.info6205.: 228 ms
    testMutatingInsertionSort           213 ms
    sort0                                15 ms
    sort1                                 0 ms
    sort2                                 0 ms
    sort3                                 0 ms
    testStaticInsertionSort               0 ms
```

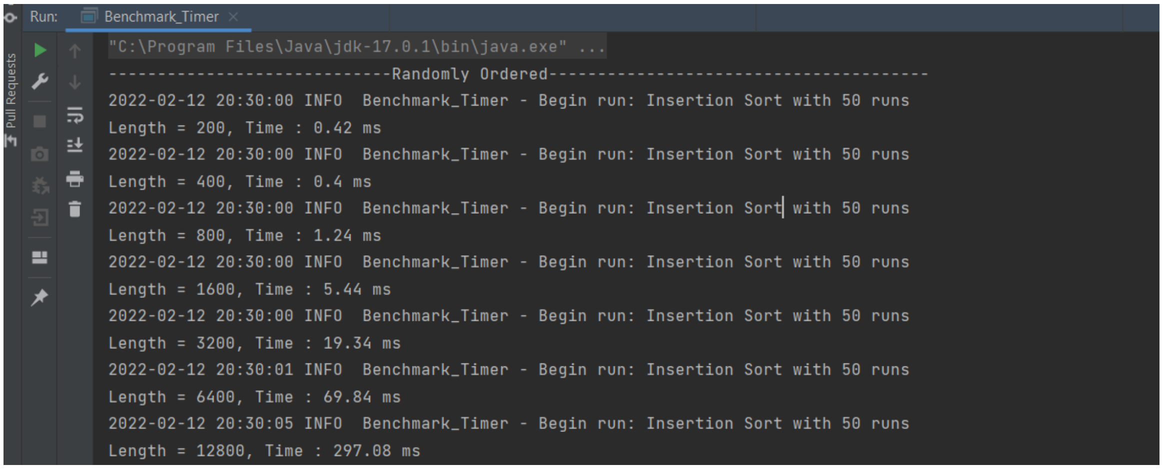
```
"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" ...
2022-02-12 18:44:14 DEBUG Config - Config.get(helper, instrument) = true
2022-02-12 18:44:14 DEBUG Config - Config.get(helper, seed) = 0
2022-02-12 18:44:14 DEBUG Config - Config.get(instrumenting, copies) = true
2022-02-12 18:44:14 DEBUG Config - Config.get(instrumenting, swaps) = true
2022-02-12 18:44:14 DEBUG Config - Config.get(instrumenting, compares) = true
2022-02-12 18:44:14 DEBUG Config - Config.get(instrumenting, inversions) = 1
2022-02-12 18:44:14 DEBUG Config - Config.get(instrumenting, fixes) = true
2022-02-12 18:44:14 DEBUG Config - Config.get(instrumenting, hits) = true
2022-02-12 18:44:14 DEBUG Config - Config.get(helper, cutoff) =
Helper for InsertionSort with 4 elements
StatPack {hits: 9,684; 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}

Process finished with exit code 0
```

Tests passed: 6 (6 minutes ago)

Part3:

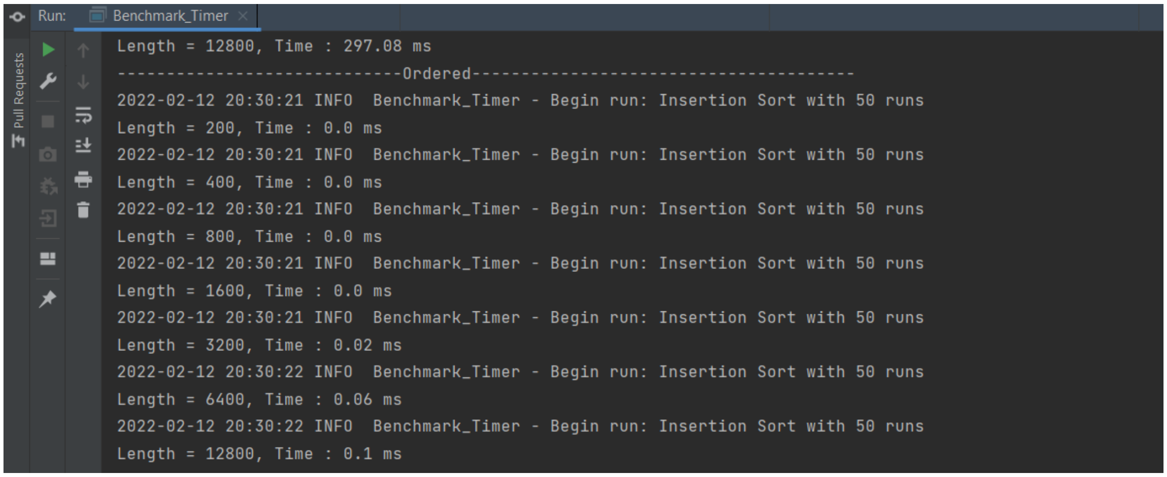
Implemented the main program in Benchmark_Timer.java to run the benchmarks: measure the running times of sort, using four different initial array ordering situations: random, ordered, partially-ordered, and reverse-ordered.

Run Results: Randomly Ordered



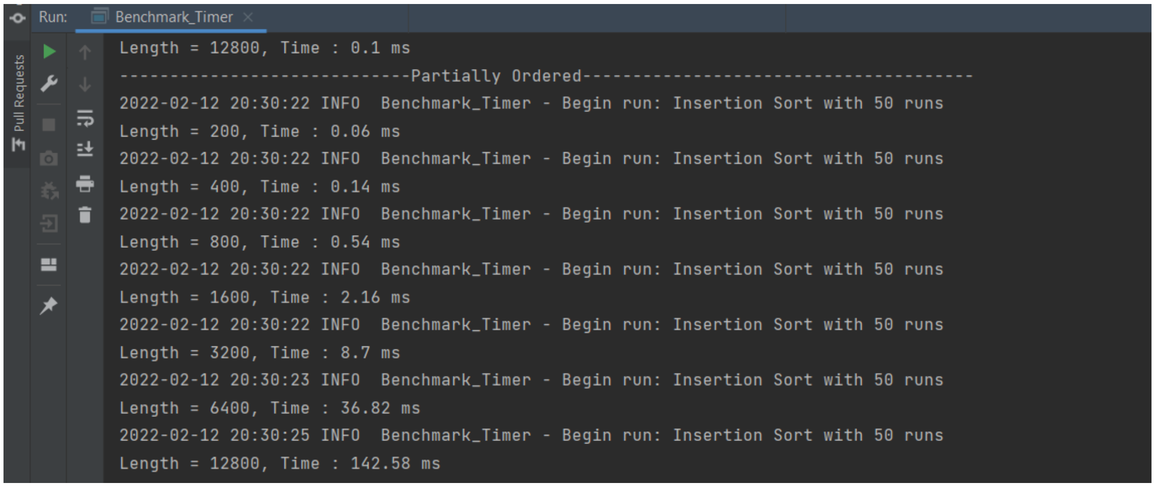
```
"C:\Program Files\Java\jdk-17.0.1\bin\java.exe" ...
-----------------------------Randomly Ordered---------------------------------------
2022-02-12 20:30:00 INFO  Benchmark_Timer - Begin run: Insertion Sort with 50 runs
Length = 200, Time : 0.42 ms
2022-02-12 20:30:00 INFO  Benchmark_Timer - Begin run: Insertion Sort with 50 runs
Length = 400, Time : 0.4 ms
2022-02-12 20:30:00 INFO  Benchmark_Timer - Begin run: Insertion Sort with 50 runs
Length = 800, Time : 1.24 ms
2022-02-12 20:30:00 INFO  Benchmark_Timer - Begin run: Insertion Sort with 50 runs
Length = 1600, Time : 5.44 ms
2022-02-12 20:30:00 INFO  Benchmark_Timer - Begin run: Insertion Sort with 50 runs
Length = 3200, Time : 19.34 ms
2022-02-12 20:30:01 INFO  Benchmark_Timer - Begin run: Insertion Sort with 50 runs
Length = 6400, Time : 69.84 ms
2022-02-12 20:30:05 INFO  Benchmark_Timer - Begin run: Insertion Sort with 50 runs
Length = 12800, Time : 297.08 ms
```

Run Results: Ordered



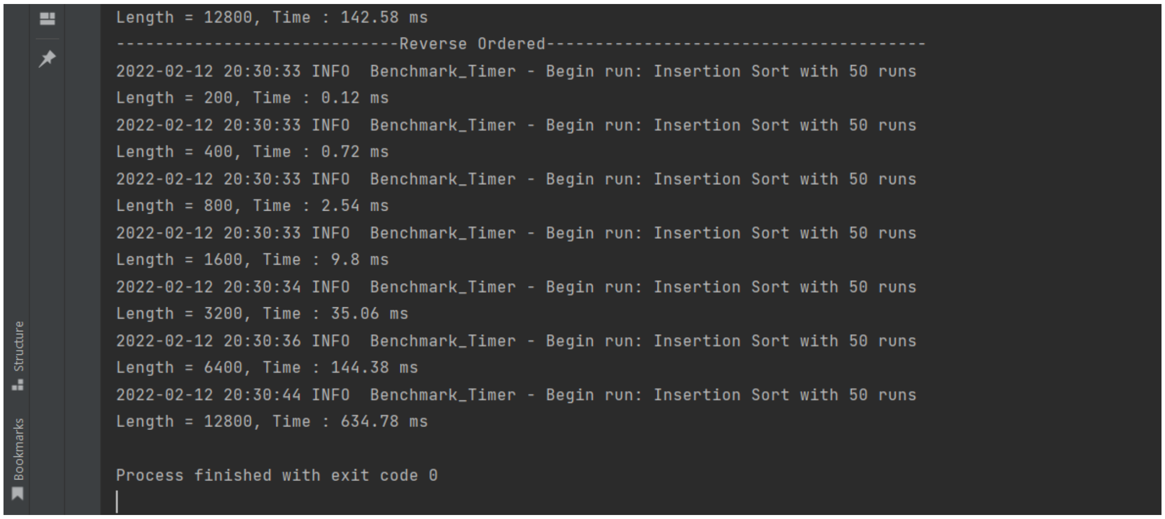
```
Length = 12800, Time : 297.08 ms
-----------------------------Ordered---------------------------------------
2022-02-12 20:30:21 INFO  Benchmark_Timer - Begin run: Insertion Sort with 50 runs
Length = 200, Time : 0.0 ms
2022-02-12 20:30:21 INFO  Benchmark_Timer - Begin run: Insertion Sort with 50 runs
Length = 400, Time : 0.0 ms
2022-02-12 20:30:21 INFO  Benchmark_Timer - Begin run: Insertion Sort with 50 runs
Length = 800, Time : 0.0 ms
2022-02-12 20:30:21 INFO  Benchmark_Timer - Begin run: Insertion Sort with 50 runs
Length = 1600, Time : 0.0 ms
2022-02-12 20:30:21 INFO  Benchmark_Timer - Begin run: Insertion Sort with 50 runs
Length = 3200, Time : 0.02 ms
2022-02-12 20:30:22 INFO  Benchmark_Timer - Begin run: Insertion Sort with 50 runs
Length = 6400, Time : 0.06 ms
2022-02-12 20:30:22 INFO  Benchmark_Timer - Begin run: Insertion Sort with 50 runs
Length = 12800, Time : 0.1 ms
```

Run Results: Partially Ordered



```
Length = 12800, Time : 0.1 ms
-----------------------------Partially Ordered---------------------------------------
2022-02-12 20:30:22 INFO  Benchmark_Timer - Begin run: Insertion Sort with 50 runs
Length = 200, Time : 0.06 ms
2022-02-12 20:30:22 INFO  Benchmark_Timer - Begin run: Insertion Sort with 50 runs
Length = 400, Time : 0.14 ms
2022-02-12 20:30:22 INFO  Benchmark_Timer - Begin run: Insertion Sort with 50 runs
Length = 800, Time : 0.54 ms
2022-02-12 20:30:22 INFO  Benchmark_Timer - Begin run: Insertion Sort with 50 runs
Length = 1600, Time : 2.16 ms
2022-02-12 20:30:22 INFO  Benchmark_Timer - Begin run: Insertion Sort with 50 runs
Length = 3200, Time : 8.7 ms
2022-02-12 20:30:23 INFO  Benchmark_Timer - Begin run: Insertion Sort with 50 runs
Length = 6400, Time : 36.82 ms
2022-02-12 20:30:25 INFO  Benchmark_Timer - Begin run: Insertion Sort with 50 runs
Length = 12800, Time : 142.58 ms
```

Run Results: Reverse Ordered



```
Length = 12800, Time : 142.58 ms
-----------------------------Reverse Ordered---------------------------------------
2022-02-12 20:30:33 INFO  Benchmark_Timer - Begin run: Insertion Sort with 50 runs
Length = 200, Time : 0.12 ms
2022-02-12 20:30:33 INFO  Benchmark_Timer - Begin run: Insertion Sort with 50 runs
Length = 400, Time : 0.72 ms
2022-02-12 20:30:33 INFO  Benchmark_Timer - Begin run: Insertion Sort with 50 runs
Length = 800, Time : 2.54 ms
2022-02-12 20:30:33 INFO  Benchmark_Timer - Begin run: Insertion Sort with 50 runs
Length = 1600, Time : 9.8 ms
2022-02-12 20:30:34 INFO  Benchmark_Timer - Begin run: Insertion Sort with 50 runs
Length = 3200, Time : 35.06 ms
2022-02-12 20:30:36 INFO  Benchmark_Timer - Begin run: Insertion Sort with 50 runs
Length = 6400, Time : 144.38 ms
2022-02-12 20:30:44 INFO  Benchmark_Timer - Begin run: Insertion Sort with 50 runs
Length = 12800, Time : 634.78 ms

Process finished with exit code 0
```

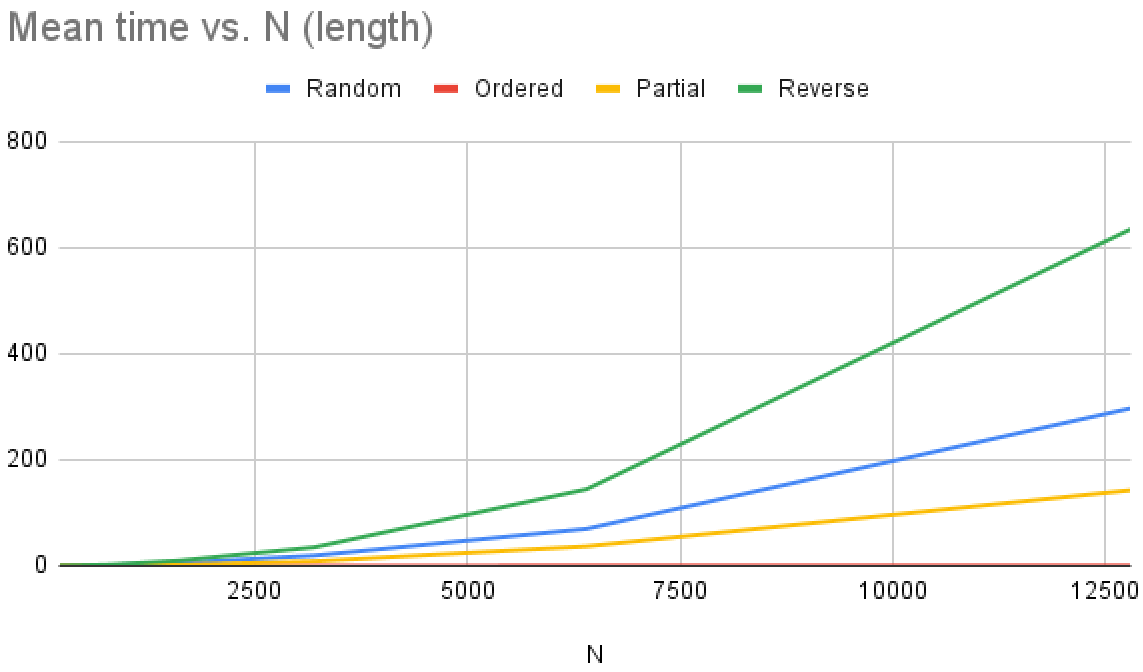
Observations:

Let us take the runtime of all the above four methods and plot it against the length of an array. Let’s start with an array of size 200 and got to 12800 by doubling and running for 50 runs. We get the following values.

N	Random	Ordered	Partial	Reverse
200	0.42 ms	0.0 ms	0.06 ms	0.12 ms
400	0.4 ms	0.0 ms	0.14 ms	0.72 ms
800	1.24 ms	0.0 ms	0.54 ms	2.54 ms
1600	5.44 ms	0.0 ms	2.16 ms	9.8 ms
3200	19.34 ms	0.02 ms	8.7 ms	35.06 ms
6400	69.84 ms	0.06 ms	36.82 ms	144.38 ms
12800	297.08 ms	0.1 ms	142.58 ms	634.78 ms

From the above values let us construct a graph of Mean time taken for 50 runs vs. Array Length (N) and we can observe the following relationship.

Relationship:



In terms of order of growth for running time of InsertionSort, from the above line plot it can be inferred that:

Ordered < Partially Ordered < Randomly Ordered < Reverse Ordered