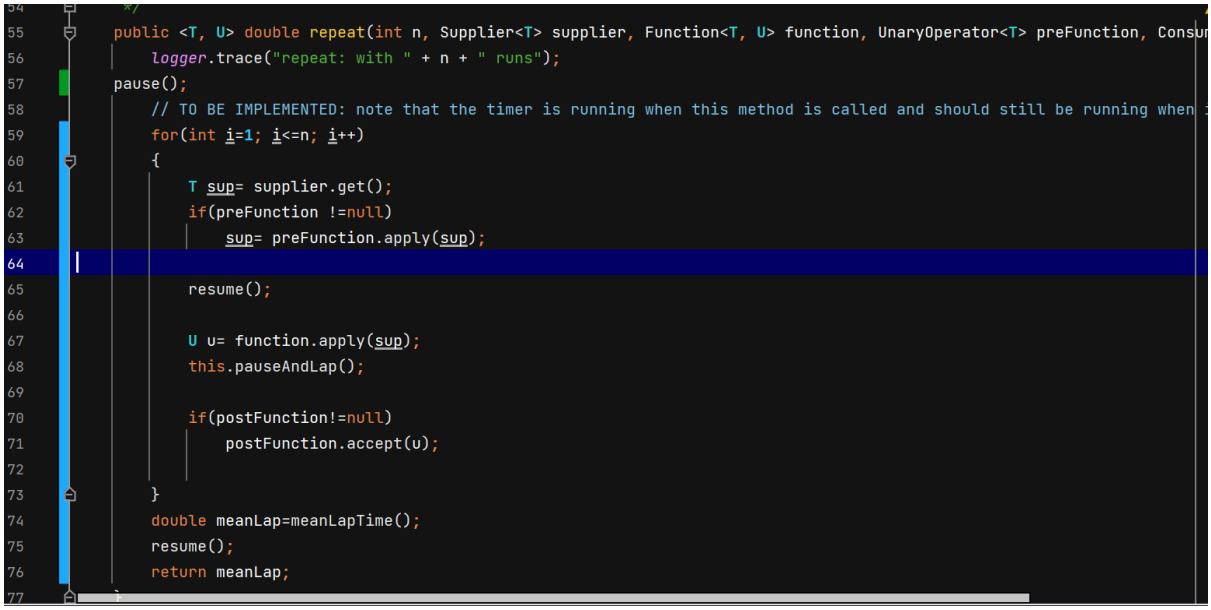
Assignment 2- Insertion Sort Benchmarking

Part 1: Implement 3 methods in Timer class-



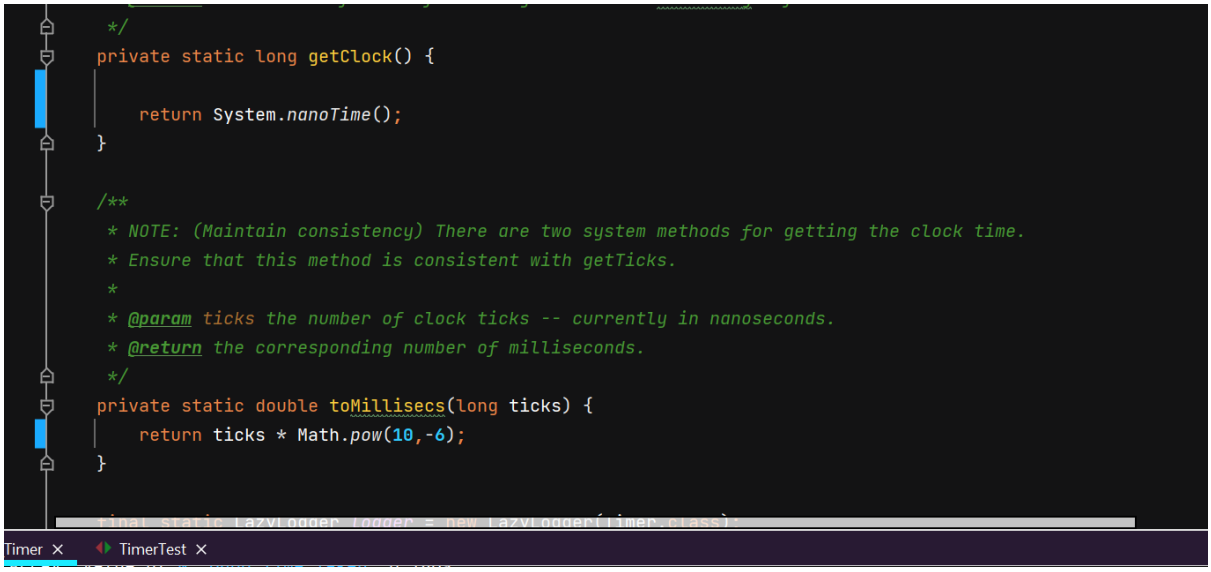
```
*/
public <T, U> double repeat(int n, Supplier<T> supplier, Function<T, U> function, UnaryOperator<T> preFunction, Consu
    logger.trace("repeat: with " + n + " runs");
pause();
    // TO BE IMPLEMENTED: note that the timer is running when this method is called and should still be running when
    for(int i=1; i<=n; i++)
    {
        T sup= supplier.get();
        if(preFunction !=null)
            sup= preFunction.apply(sup);

        resume();

        U u= function.apply(sup);
        this.pauseAndLap();

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

    }
    double meanLap=meanLapTime();
    resume();
    return meanLap;
}
```



```
 */
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.
 *
 * @param ticks the number of clock ticks -- currently in nanoseconds.
 * @return the corresponding number of milliseconds.
 */
private static double toMillisecs(long ticks) {
    return ticks * Math.pow(10,-6);
}

final static LazyLogger logger = new LazyLogger(Timer.class);
```

Timer × TimerTest ×

Part 2: Implement Insertion Sort-

```java
    */
    public void sort(X[] xs, int from, int to) {
        final Helper<X> helper = getHelper();
        for(int i= from; i< to; i++)
        {
            for(int j=i; j>from && helper.compare(xs[j-1],xs[j]) > 0;j--)
            {
                helper.swap(xs, j-1,j);
            }
        }
        // TO BE IMPLEMENTED
    }

    public static final String DESCRIPTION = "Insertion sort";

    public static <T extends Comparable<T>> void sort(T[] ts) { new InsertionSort<T>().mutatingSort(ts); }
}
```

Part 3: Implement main method to perform Insertion Sort analysis-

Implemented main() in Benchamark_Timer.java class to perform the Insertion Sort analysis.

The analysis was run for- Random, Reverse, Ordered and Partially Ordered Arrays for 20 runs.

```java
final static LazyLogger logger = new LazyLogger(Benchmark_Timer.class);

public static void main(String[] args)
{
    InsertionSort insertionSort=new InsertionSort();
    Benchmark_Timer<Integer[]> benchmarkTimer=new Benchmark_Timer<>
            (description: "Benchmarking", fPre: null,(i)->insertionSort.sort(i, from: 0,i.length), fPost: null);

    System.out.println("--------Randomly Ordered Array-------");
    for(int i=200;i<=7000;i*=2)
    {
        int j=i;
        Supplier<Integer[]> supplier=new Supplier<Integer[]>() {
            @Override
            public Integer[] get() {
                Random random=new Random();
                Integer arr[]=new Integer[j];
                for(int k=0; k<j;k++)
                {
                    arr[k]= random.nextInt();
                }
                return arr;
            }
```

Analysis Conclusion:

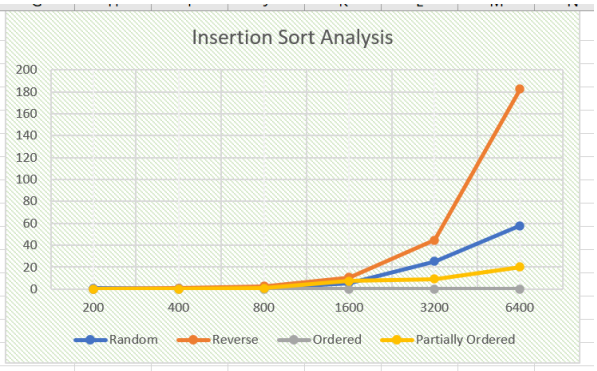
After analysing the results for 20 runs on differently ordered arrays the below conclusion can be drawn based on the run time of the insertion sort:

Ordered Array< Partially Ordered< Randomly Ordered< Reverse Ordered

For Ordered Array the run time is O(1).

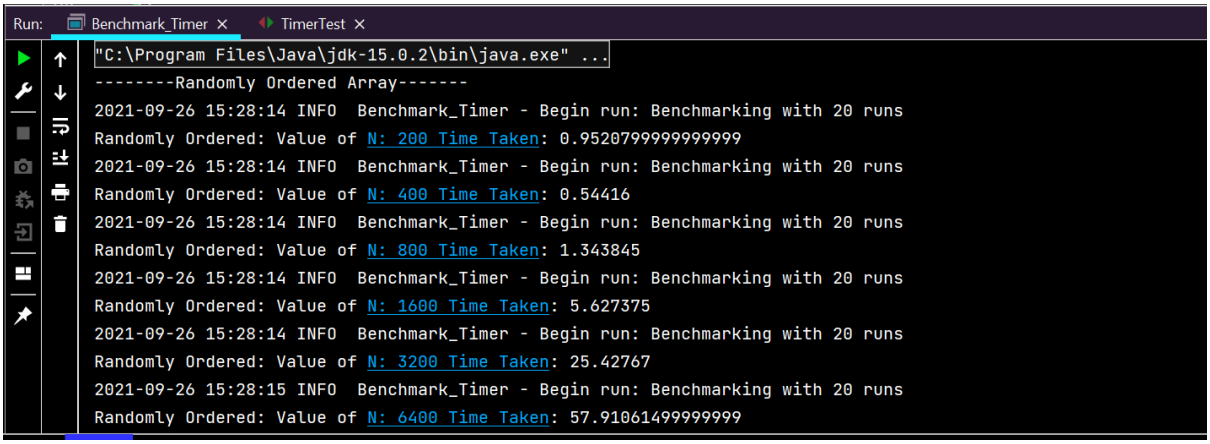
Graphical Analysis:

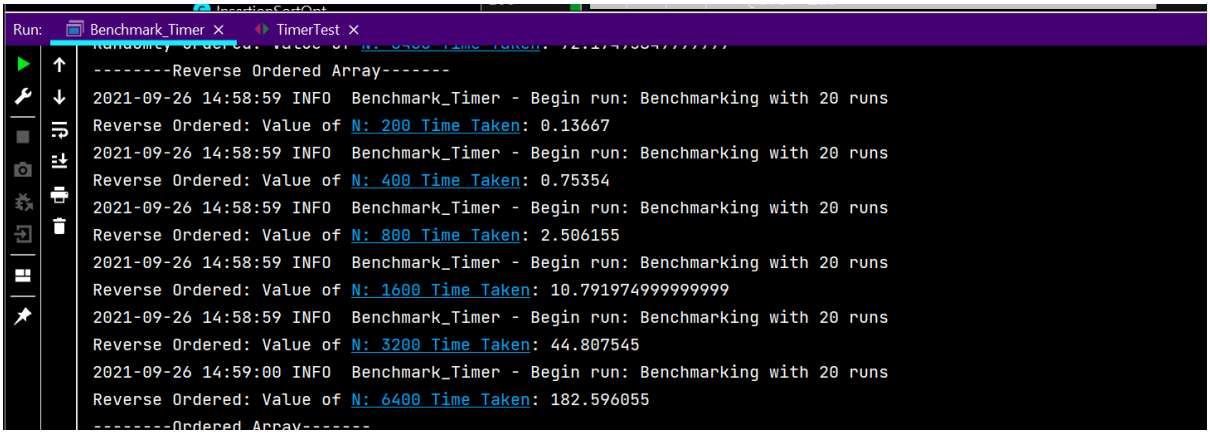
Benchmarking with 20 runs				
N	Random	Reverse	Ordered	Partially Ordered
200	0.95	0.136	0.078	0.046
400	0.544	0.753	0.016	0.196
800	1.34	2.506	0.041	0.9508
1600	5.62	10.79	0.084	7.26
3200	25.42	44.8	0.225	9.051
6400	57.91	182.59	0.186	20.18

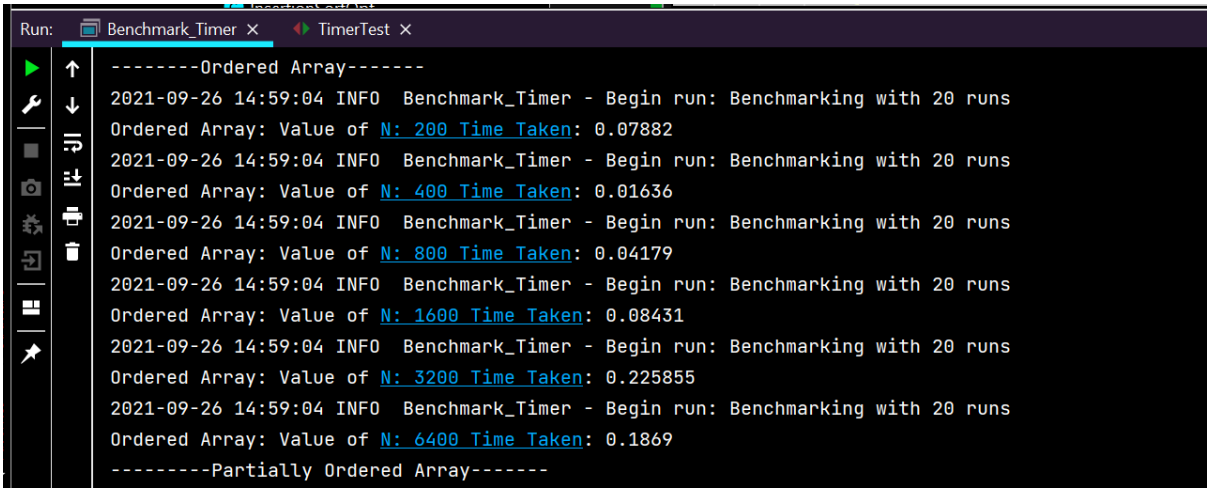


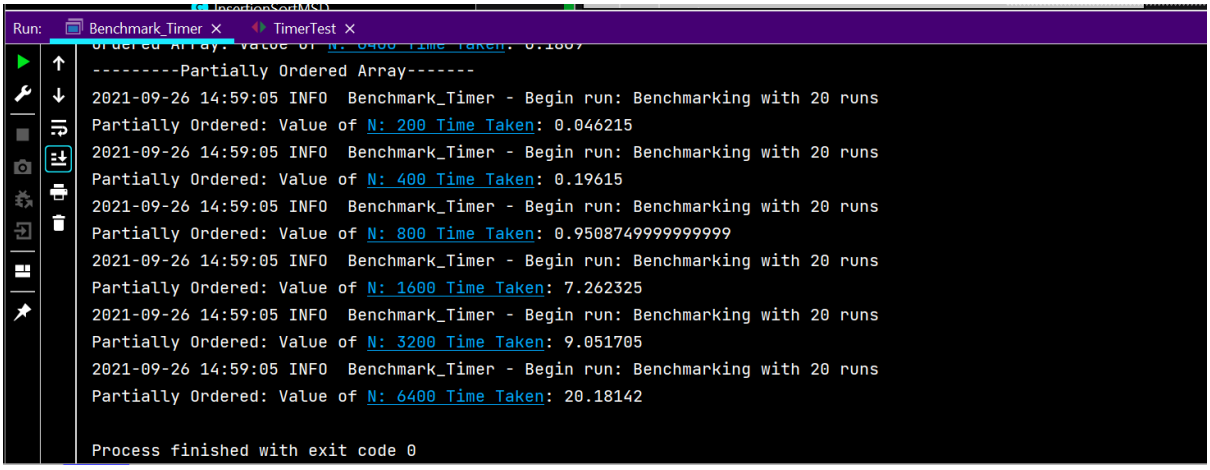
Evidence:

1. Console Output for Benchmark_Timer run for differently ordered arrays:



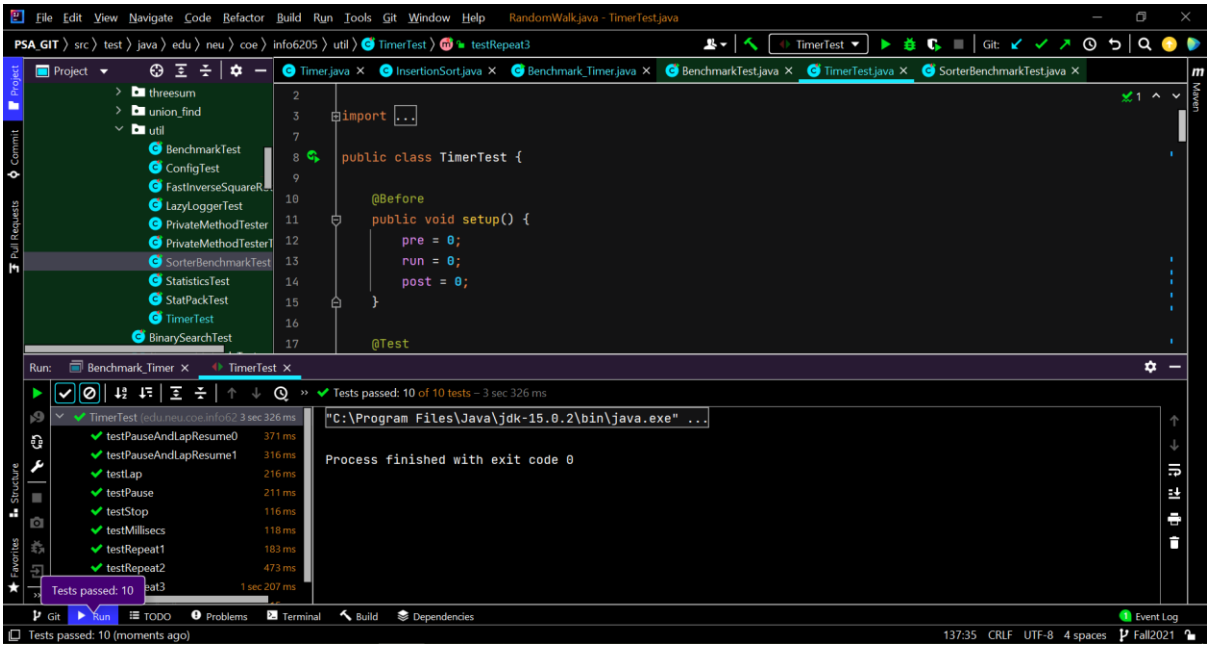




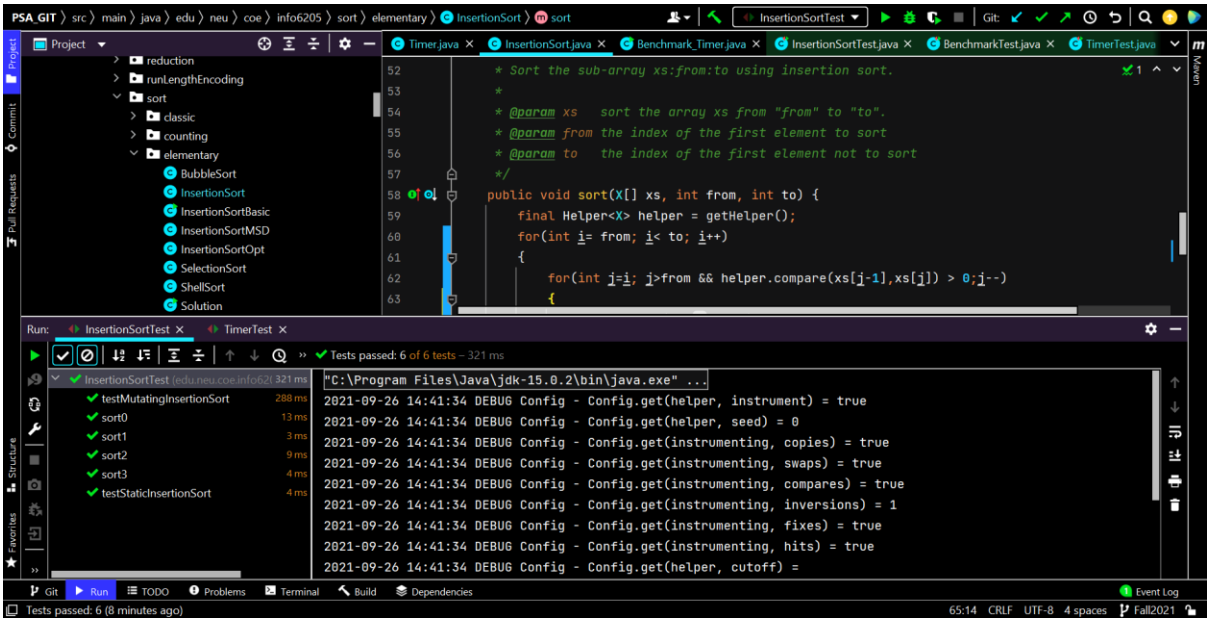


2. Unit Test Evidence

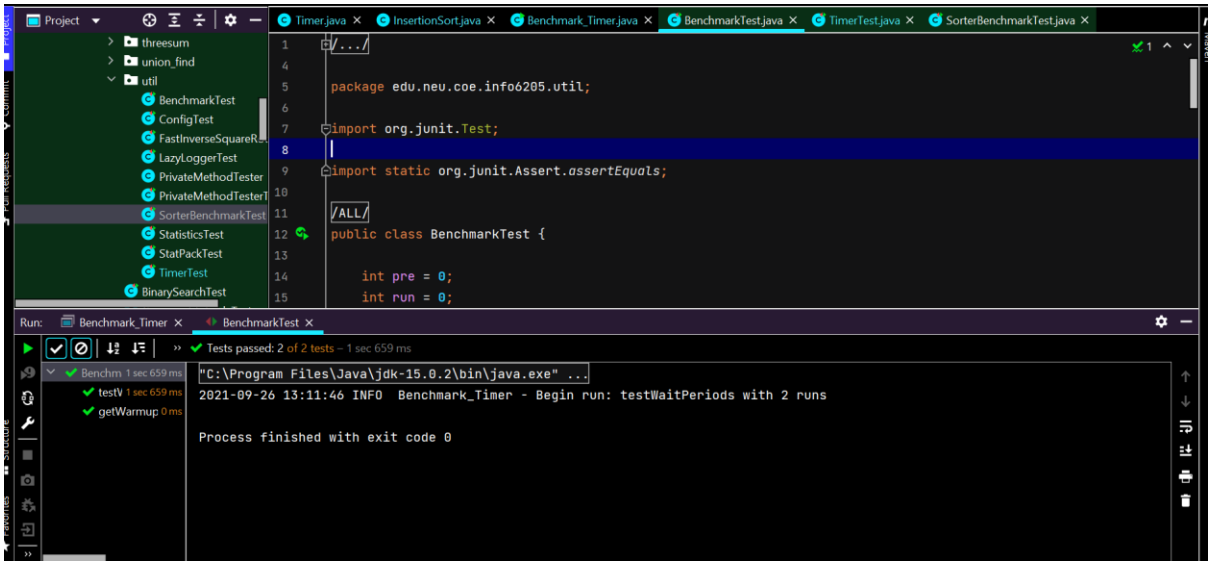
- **TimerTest**



- **InsertionSortTest**



- **BenchmarkTest**


Project
threesum
union_find
util
BenchmarkTest
ConfigTest
FastInverseSquareR
LazyLoggerTest
PrivateMethodTester
PrivateMethodTesterT
SorterBenchmarkTest
StatisticsTest
StatPackTest
TimerTest
BinarySearchTest
Timer.java
InsertionSort.java
Benchmark_Timer.java
BenchmarkTest.java
TimerTest.java
SorterBenchmarkTest.java
/.../
package edu.neu.coe.info6205.util;
import org.junit.Test;
import static org.junit.Assert.assertEquals;
/ALL/
public class BenchmarkTest {
int pre = 0;
int run = 0;
Run:
Benchmark_Timer
BenchmarkTest
Tests passed: 2 of 2 tests – 1 sec 659 ms
Benchm 1 sec 659 ms
testV 1 sec 659 ms
getWarmup 0 ms
"C:\Program Files\Java\jdk-15.0.2\bin\java.exe" ...
2021-09-26 13:11:46 INFO Benchmark_Timer - Begin run: testWaitPeriods with 2 runs
Process finished with exit code 0