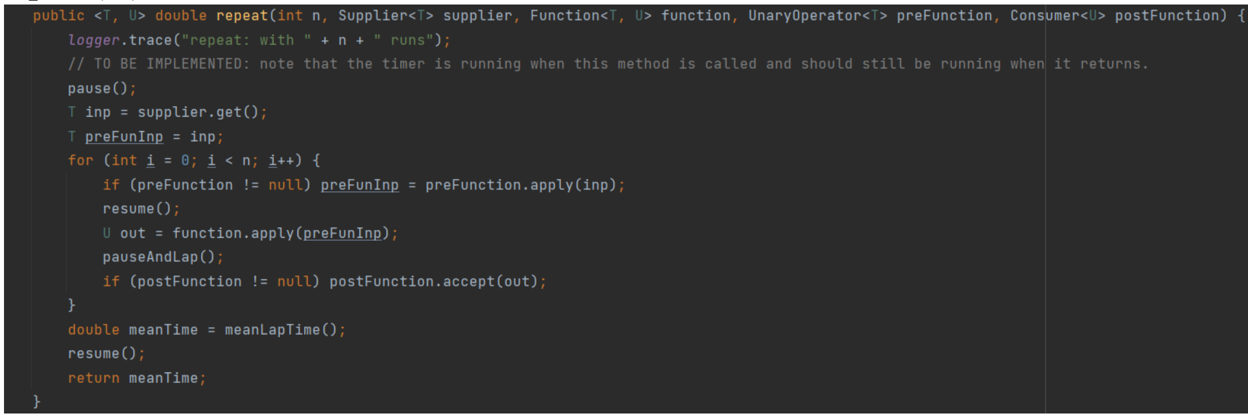
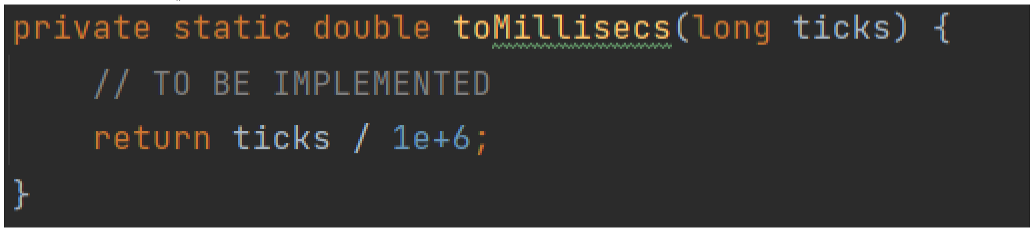
# Tanay Saxena (001586302)
# Program Structures & Algorithms
# Fall 2021
# Assignment No. 2

- Task (List down the tasks performed in the Assignment)
  - Added missing code in 4 methods -
    Class Timer.java
    repeat(...) -



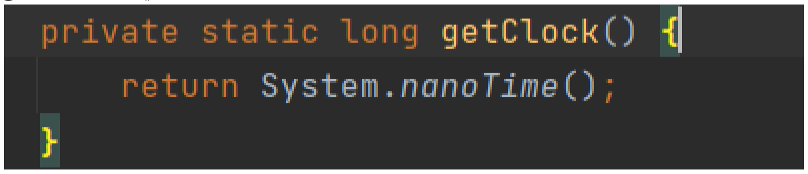
```
public <T, U> double repeat(int n, Supplier<T> supplier, Function<T, U> function, UnaryOperator<T> preFunction, Consumer<U> postFunction) {
    logger.trace("repeat: with " + n + " runs");
    // TO BE IMPLEMENTED: note that the timer is running when this method is called and should still be running when it returns.
    pause();
    T inp = supplier.get();
    T preFunInp = inp;
    for (int i = 0; i < n; i++) {
        if (preFunction != null) preFunInp = preFunction.apply(inp);
        resume();
        U out = function.apply(preFunInp);
        pauseAndLap();
        if (postFunction != null) postFunction.accept(out);
    }
    double meanTime = meanLapTime();
    resume();
    return meanTime;
}
```

toMillisecs() -



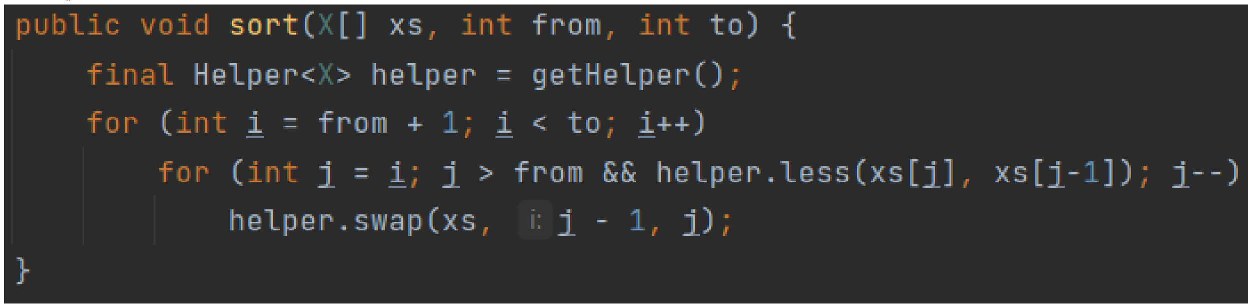
```
private static double toMillisecs(long ticks) {
    // TO BE IMPLEMENTED
    return ticks / 1e+6;
}
```

getClock()



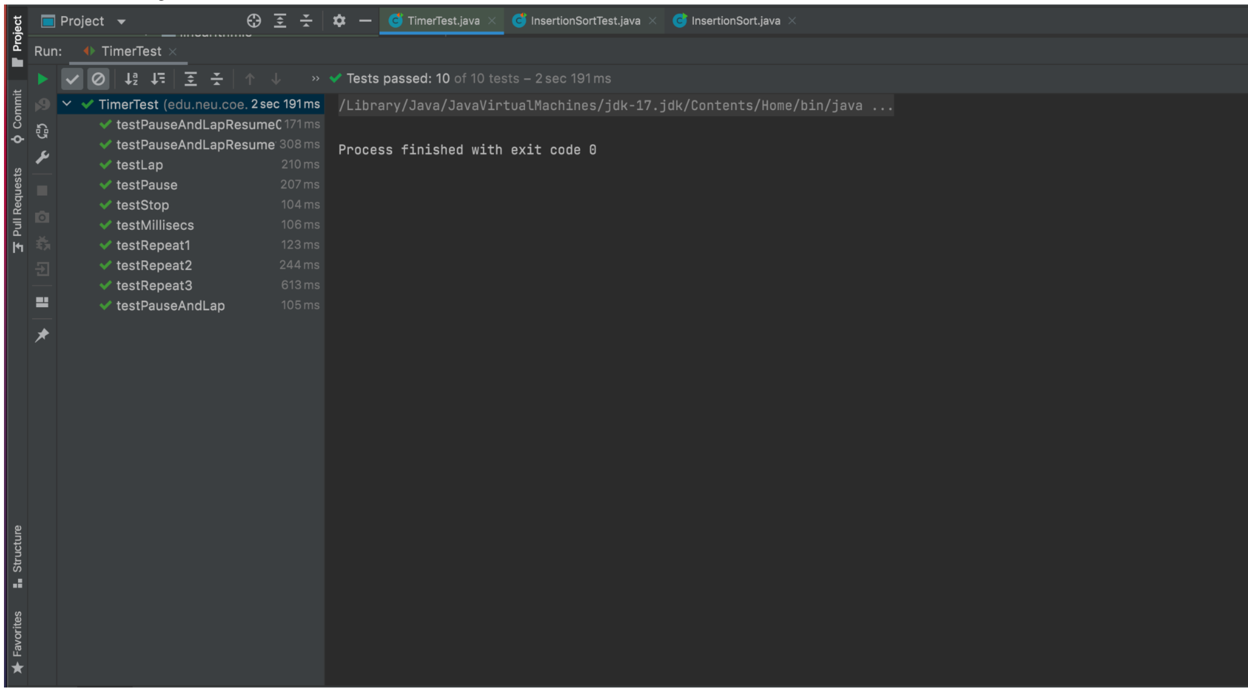
```
private static long getClock() {
    return System.nanoTime();
}
```

Class InsertionSort
sort() -



```
public void sort(X[] xs, int from, int to) {
    final Helper<X> helper = getHelper();
    for (int i = from + 1; i < to; i++)
        for (int j = i; j > from && helper.less(xs[j], xs[j-1]); j--)
            helper.swap(xs, i: j - 1, j);
}
```

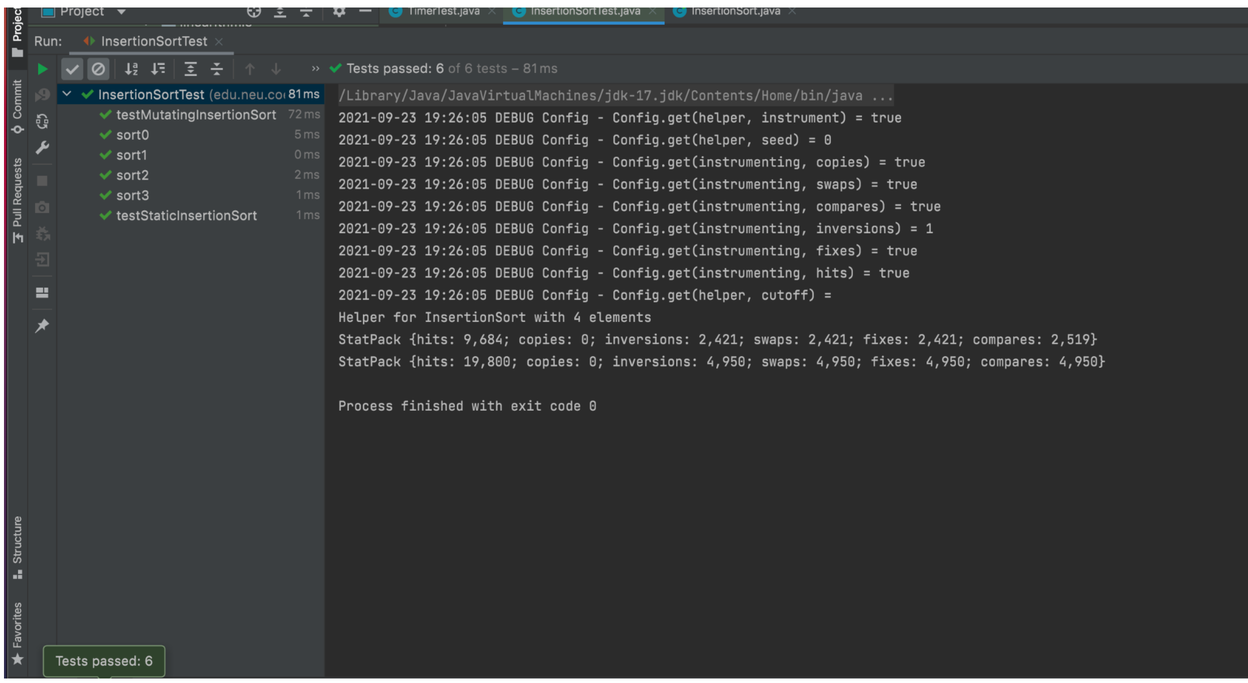
- Ran tests for InsertionSort class and Timer class - TimerTest.java -



```
Run: TimerTest
Tests passed: 10 of 10 tests – 2 sec 191 ms
TimerTest (edu.neu.coe. 2 sec 191 ms
  testPauseAndLapResumeC 171 ms
  testPauseAndLapResume 308 ms
  testLap 210 ms
  testPause 207 ms
  testStop 104 ms
  testMillisecs 106 ms
  testRepeat1 123 ms
  testRepeat2 244 ms
  testRepeat3 613 ms
  testPauseAndLap 105 ms
```

```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java ...

Process finished with exit code 0
```

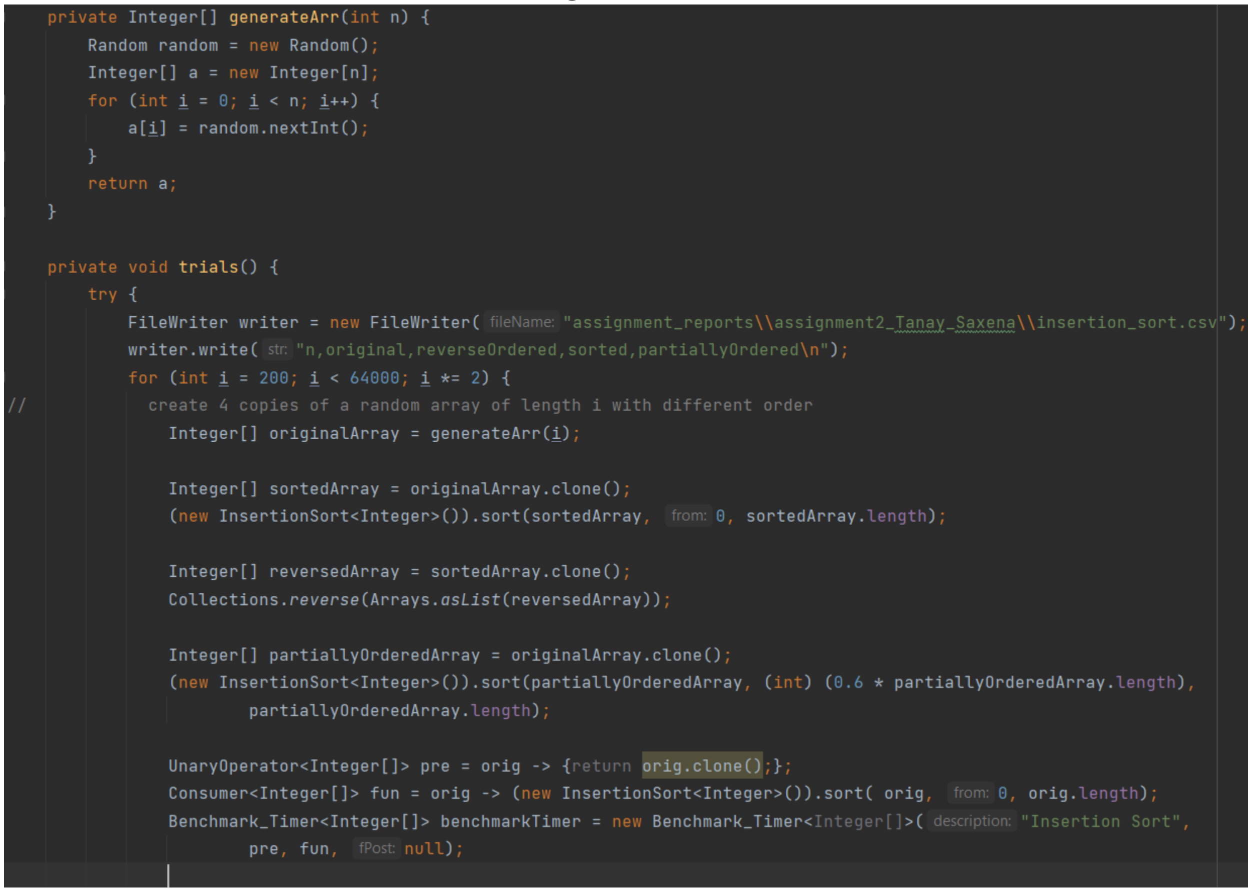
InsertionSortTest -



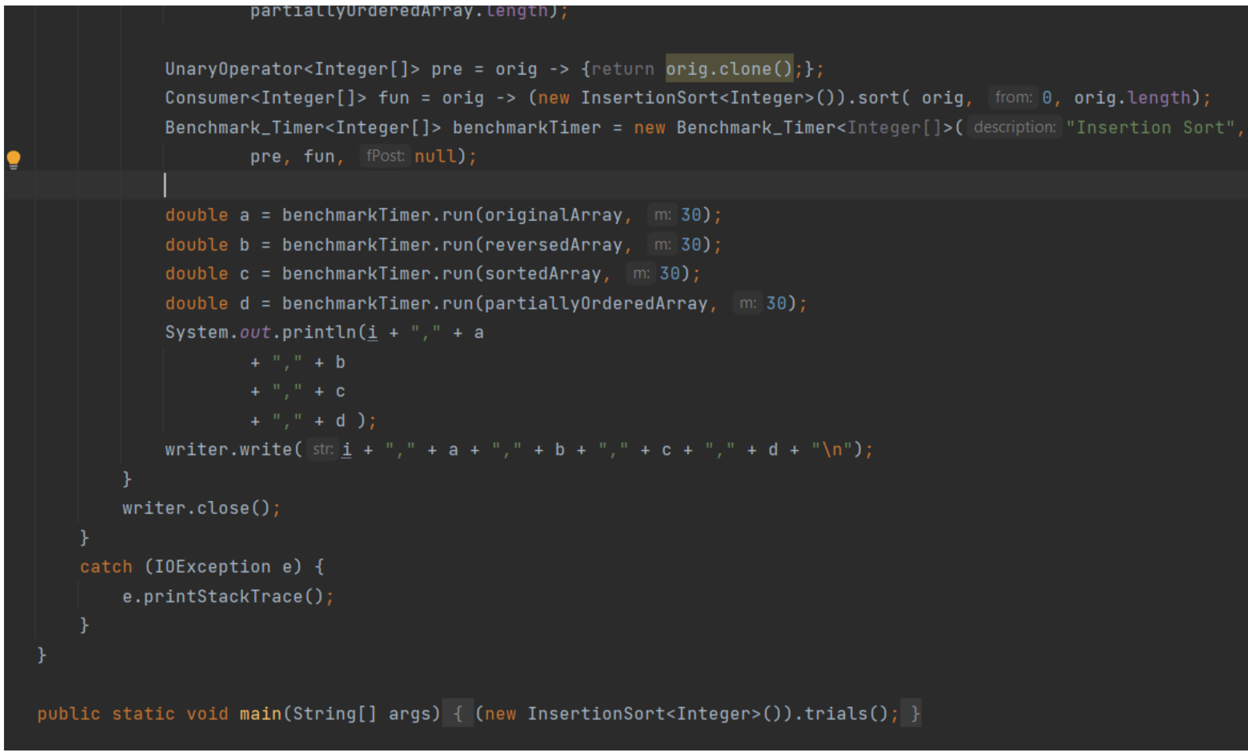
```
Run: InsertionSortTest
Tests passed: 6 of 6 tests – 81 ms
InsertionSortTest (edu.neu.co 81 ms
  testMutatingInsertionSort 72 ms
  sort0 5 ms
  sort1 0 ms
  sort2 2 ms
  sort3 1 ms
  testStaticInsertionSort 1 ms
Tests passed: 6
```

```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java ...
2021-09-23 19:26:05 DEBUG Config - Config.get(helper, instrument) = true
2021-09-23 19:26:05 DEBUG Config - Config.get(helper, seed) = 0
2021-09-23 19:26:05 DEBUG Config - Config.get(instrumenting, copies) = true
2021-09-23 19:26:05 DEBUG Config - Config.get(instrumenting, swaps) = true
2021-09-23 19:26:05 DEBUG Config - Config.get(instrumenting, compares) = true
2021-09-23 19:26:05 DEBUG Config - Config.get(instrumenting, inversions) = 1
2021-09-23 19:26:05 DEBUG Config - Config.get(instrumenting, fixes) = true
2021-09-23 19:26:05 DEBUG Config - Config.get(instrumenting, hits) = true
2021-09-23 19:26:05 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
```

- Created methods for the benchmarking -

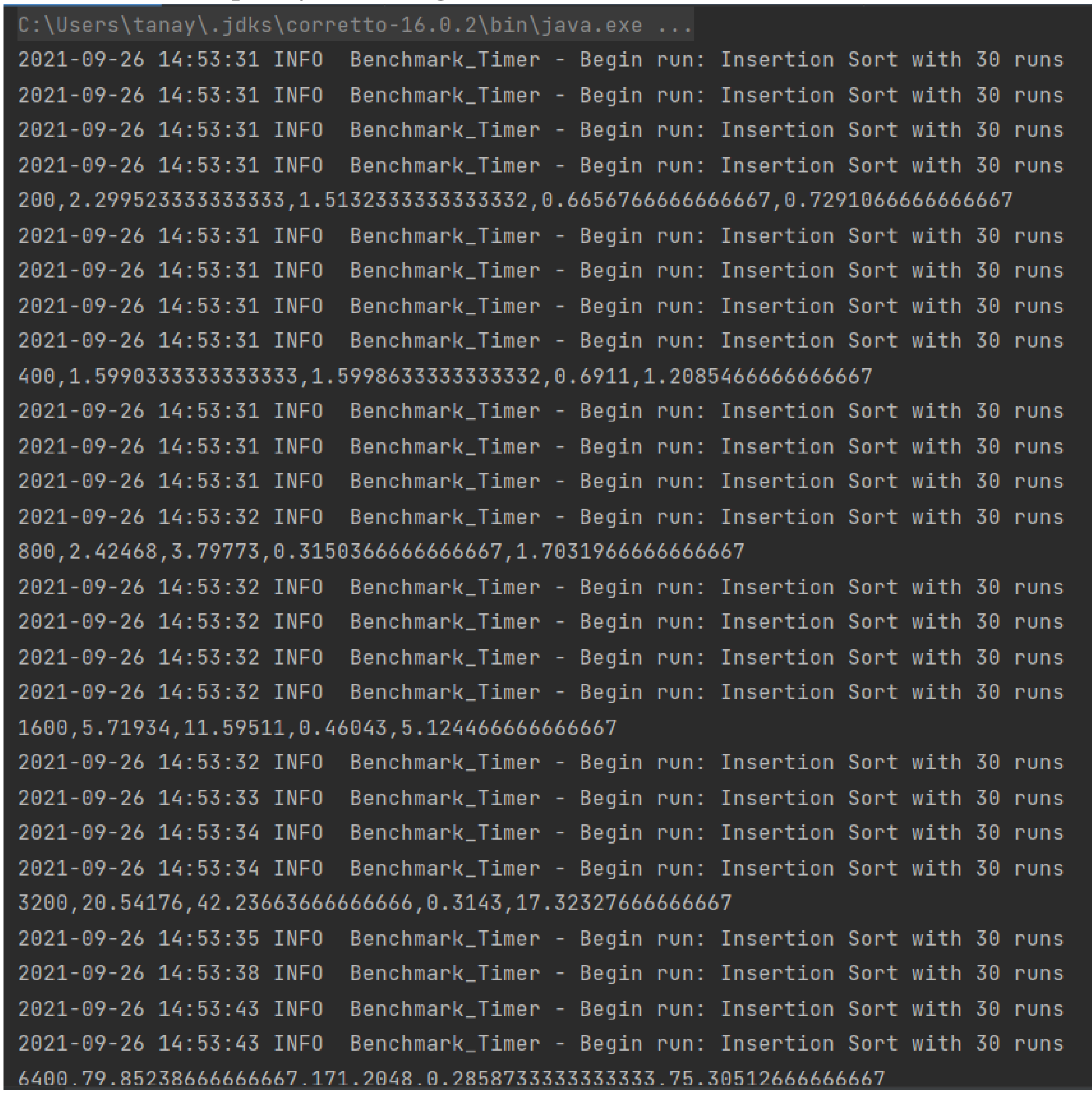


```
private Integer[] generateArr(int n) {
    Random random = new Random();
    Integer[] a = new Integer[n];
    for (int i = 0; i < n; i++) {
        a[i] = random.nextInt();
    }
    return a;
}

private void trials() {
    try {
        FileWriter writer = new FileWriter( fileName: "assignment_reports\\assignment2_Tanay_Saxena\\insertion_sort.csv");
        writer.write( str: "n,original,reverseOrdered,sorted,partiallyOrdered\n");
        for (int i = 200; i < 64000; i *= 2) {
//              create 4 copies of a random array of length i with different order
            Integer[] originalArray = generateArr(i);

            Integer[] sortedArray = originalArray.clone();
            (new InsertionSort<Integer>()).sort(sortedArray,  from: 0, sortedArray.length);

            Integer[] reversedArray = sortedArray.clone();
            Collections.reverse(Arrays.asList(reversedArray));

            Integer[] partiallyOrderedArray = originalArray.clone();
            (new InsertionSort<Integer>()).sort(partiallyOrderedArray, (int) (0.6 * partiallyOrderedArray.length),
                    partiallyOrderedArray.length);

            UnaryOperator<Integer[]> pre = orig -> {return orig.clone();};
            Consumer<Integer[]> fun = orig -> (new InsertionSort<Integer>()).sort( orig,  from: 0, orig.length);
            Benchmark_Timer<Integer[]> benchmarkTimer = new Benchmark_Timer<Integer[]>( description: "Insertion Sort",
                    pre, fun,  fPost: null);
```

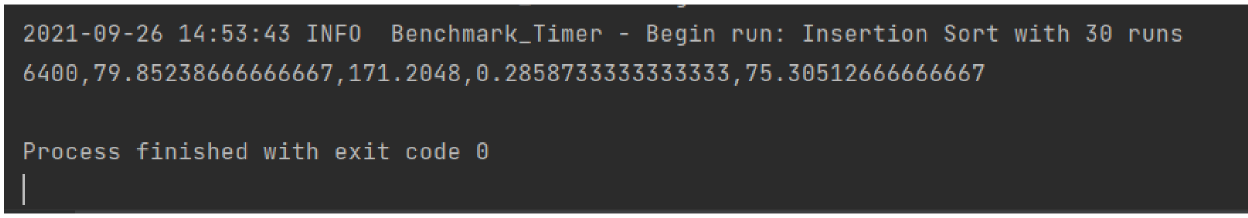


```
                    partiallyOrderedArray.length);

            UnaryOperator<Integer[]> pre = orig -> {return orig.clone();};
            Consumer<Integer[]> fun = orig -> (new InsertionSort<Integer>()).sort( orig,  from: 0, orig.length);
            Benchmark_Timer<Integer[]> benchmarkTimer = new Benchmark_Timer<Integer[]>( description: "Insertion Sort",
                    pre, fun,  fPost: null);

            double a = benchmarkTimer.run(originalArray,  m: 30);
            double b = benchmarkTimer.run(reversedArray,  m: 30);
            double c = benchmarkTimer.run(sortedArray,  m: 30);
            double d = benchmarkTimer.run(partiallyOrderedArray,  m: 30);
            System.out.println(i + "," + a
                    + "," + b
                    + "," + c
                    + "," + d );
            writer.write( str: i + "," + a + "," + b + "," + c + "," + d + "\n");
        }
        writer.close();
    }
    catch (IOException e) {
        e.printStackTrace();
    }
}

public static void main(String[] args) { (new InsertionSort<Integer>()).trials(); }
```

○ Generated the output by executing the main method -



```
C:\Users\tanay\.jdks\corretto-16.0.2\bin\java.exe ...
2021-09-26 14:53:31 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:31 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:31 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:31 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
200,2.299523333333333,1.5132333333333332,0.6656766666666667,0.7291066666666667
2021-09-26 14:53:31 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:31 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:31 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:31 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
400,1.5990333333333333,1.5998633333333332,0.6911,1.2085466666666667
2021-09-26 14:53:31 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:31 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:31 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:32 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
800,2.42468,3.79773,0.3150366666666667,1.7031966666666667
2021-09-26 14:53:32 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:32 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:32 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:32 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
1600,5.71934,11.59511,0.46043,5.124466666666667
2021-09-26 14:53:32 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:33 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:34 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:34 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
3200,20.54176,42.23663666666666,0.3143,17.32327666666667
2021-09-26 14:53:35 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:38 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:43 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:43 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
6400,79.85238666666667,171.2048,0.2858733333333333,75.30512666666667
```

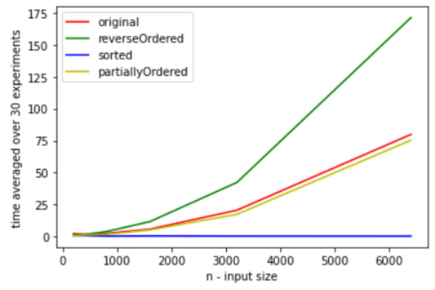


```
2021-09-26 14:53:43 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
6400,79.85238666666667,171.2048,0.2858733333333333,75.30512666666667

Process finished with exit code 0
```

- Created a jupyter notebook and performed analysis on the output (csv format)

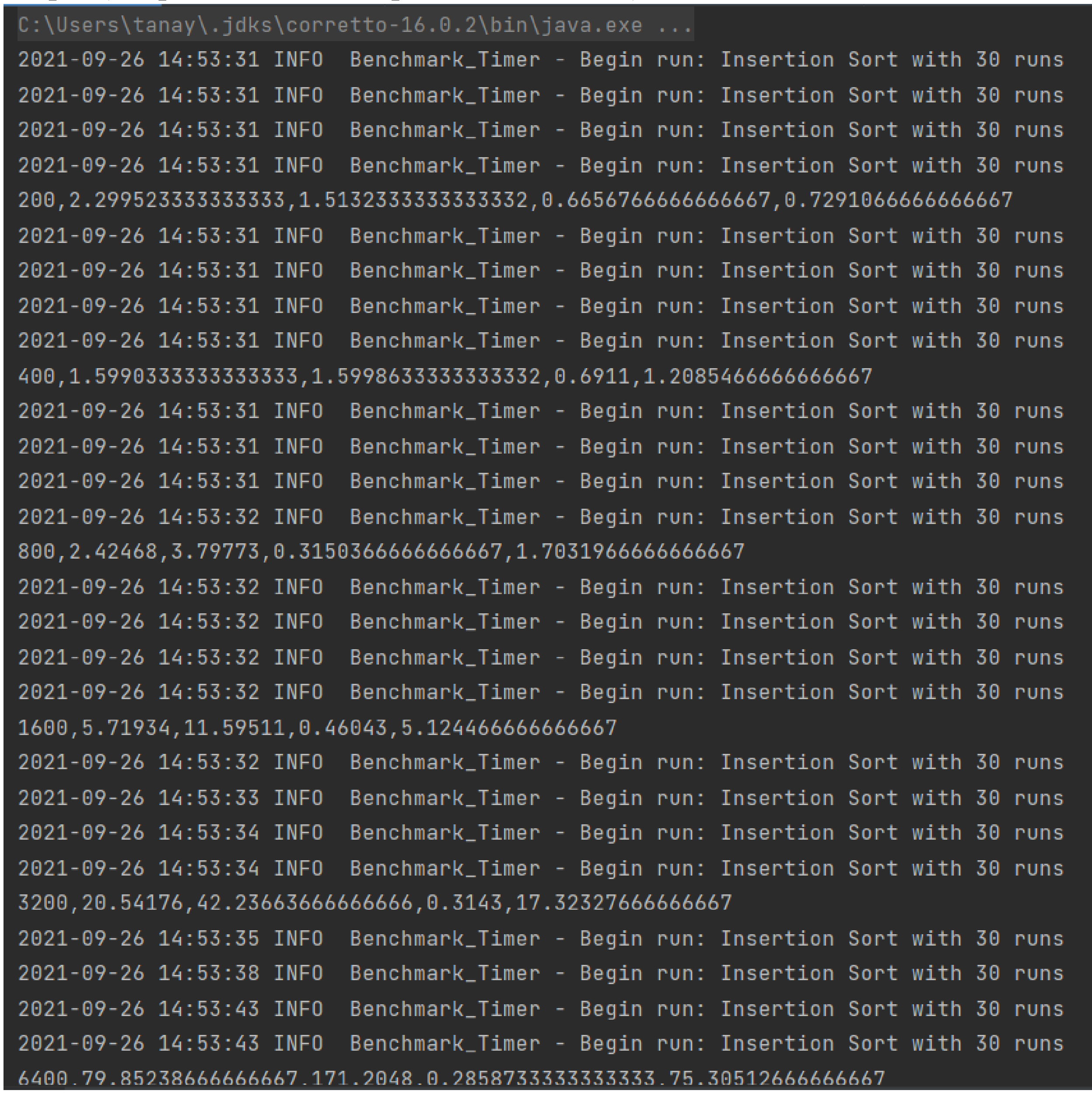
```
In [1]: import pandas as pd
        import matplotlib.pyplot as plt
```

```
In [2]: insertion_sort_df = pd.read_csv('insertion_sort.csv')
        insertion_sort_df

        plt.plot(insertion_sort_df['n'], insertion_sort_df['original'], "r", label = 'original')
        plt.plot(insertion_sort_df['n'], insertion_sort_df['reverseOrdered'], "g", label = 'reverseOrdered')
        plt.plot(insertion_sort_df['n'], insertion_sort_df['sorted'], "b", label = 'sorted')
        plt.plot(insertion_sort_df['n'], insertion_sort_df['partiallyOrdered'], "y", label = 'partiallyOrdered')
        plt.xlabel('n - input size')
        plt.ylabel('time averaged over 30 experiments')
        plt.legend()
        plt.show()
```

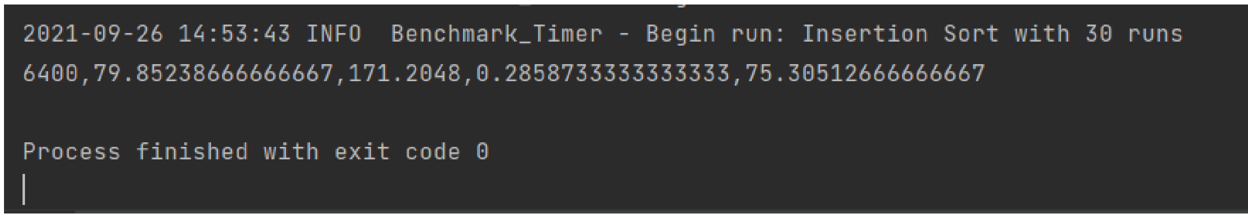


- Relationship Conclusion:
  - The graph appears to be in line with the time complexity of insertion algorithm O(n^2) hence the relationship t ∝ n^2 (where t is time and n is the input size)
  - Furthermore, we can see from the graphs that the reversed array performs worst and the sorted array the best.

- Evidence to support the conclusion:
    1. Output (Snapshot of Code output in the terminal)

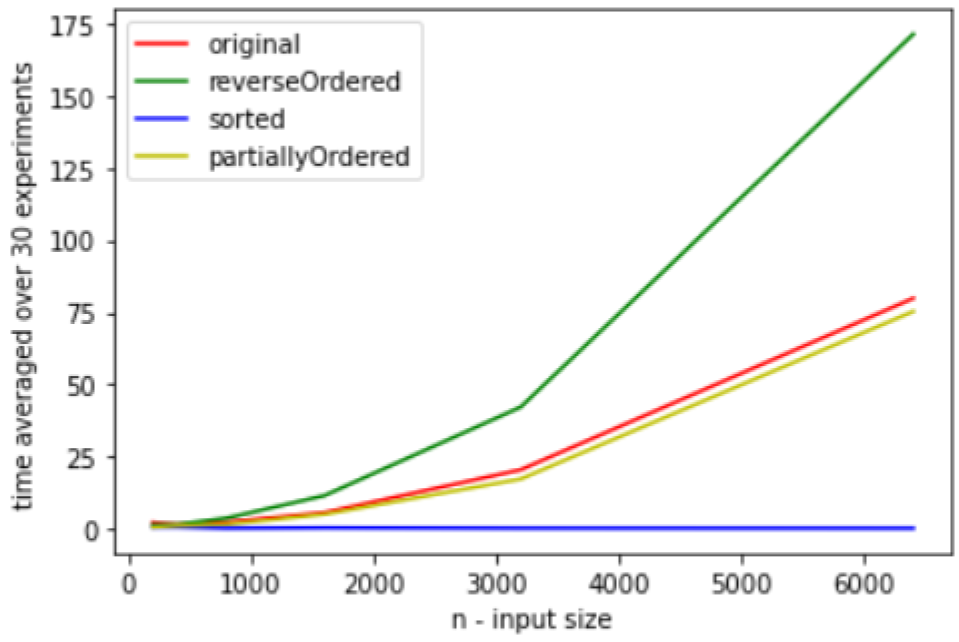


```
C:\Users\tanay\.jdks\corretto-16.0.2\bin\java.exe ...
2021-09-26 14:53:31 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:31 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:31 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:31 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
200,2.2995233333333333,1.5132333333333332,0.6656766666666667,0.7291066666666667
2021-09-26 14:53:31 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:31 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:31 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:31 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
400,1.5990333333333333,1.5998633333333332,0.6911,1.2085466666666667
2021-09-26 14:53:31 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:31 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:31 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:32 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
800,2.42468,3.79773,0.3150366666666667,1.7031966666666667
2021-09-26 14:53:32 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:32 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:32 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:32 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
1600,5.71934,11.59511,0.46043,5.124466666666667
2021-09-26 14:53:32 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:33 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:34 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:34 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
3200,20.54176,42.23663666666666,0.3143,17.32327666666667
2021-09-26 14:53:35 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:38 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:43 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
2021-09-26 14:53:43 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
6400,79.85238666666667,171.2048,0.2858733333333333,75.30512666666667
```

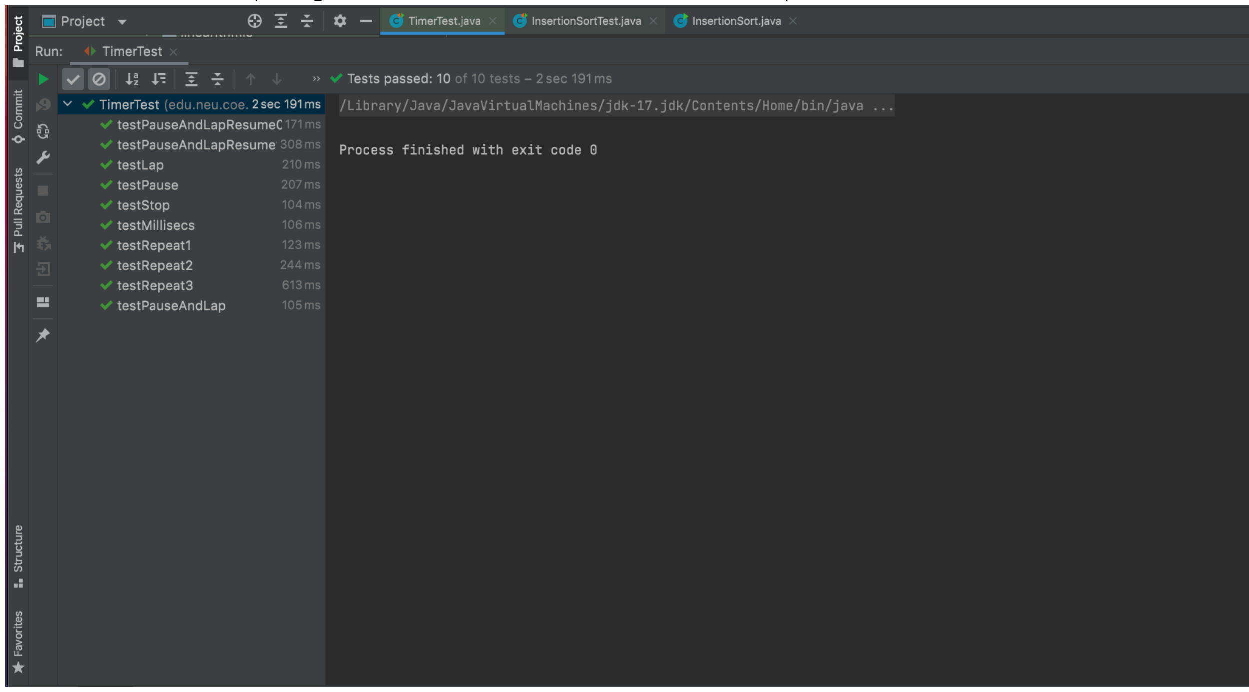


```
2021-09-26 14:53:43 INFO  Benchmark_Timer - Begin run: Insertion Sort with 30 runs
6400,79.85238666666667,171.2048,0.2858733333333333,75.30512666666667

Process finished with exit code 0
```

2. Graphical Representation(Observations)

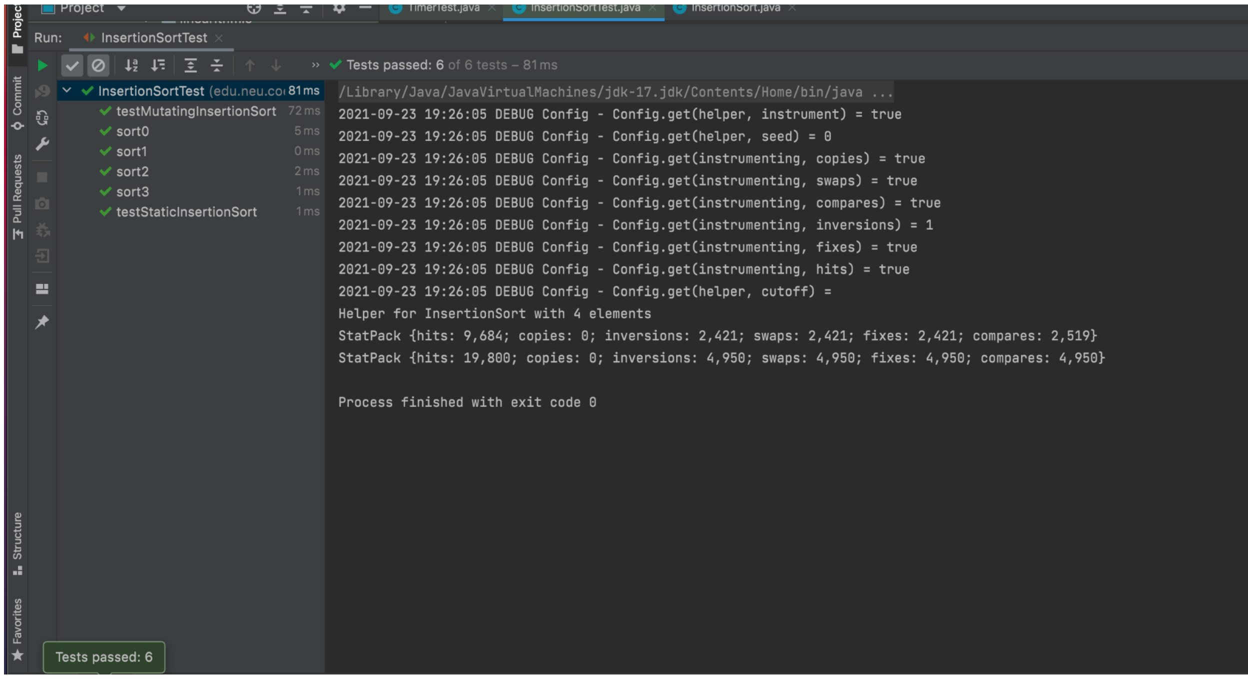


3. Unit tests result:(Snapshot of successful unit test run)



```
Tests passed: 10 of 10 tests – 2 sec 191 ms

TimerTest (edu.neu.coe. 2 sec 191 ms
    testPauseAndLapResumeC 171 ms
    testPauseAndLapResume  308 ms
    testLap                210 ms
    testPause              207 ms
    testStop               104 ms
    testMillisecs          106 ms
    testRepeat1            123 ms
    testRepeat2            244 ms
    testRepeat3            613 ms
    testPauseAndLap        105 ms
```

```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java ...

Process finished with exit code 0
```



Run: InsertionSortTest
Tests passed: 6 of 6 tests – 81 ms
InsertionSortTest (edu.neu.co 81 ms
testMutatingInsertionSort 72 ms
sort0 5 ms
sort1 0 ms
sort2 2 ms
sort3 1 ms
testStaticInsertionSort 1 ms
Tests passed: 6

```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java ...
2021-09-23 19:26:05 DEBUG Config - Config.get(helper, instrument) = true
2021-09-23 19:26:05 DEBUG Config - Config.get(helper, seed) = 0
2021-09-23 19:26:05 DEBUG Config - Config.get(instrumenting, copies) = true
2021-09-23 19:26:05 DEBUG Config - Config.get(instrumenting, swaps) = true
2021-09-23 19:26:05 DEBUG Config - Config.get(instrumenting, compares) = true
2021-09-23 19:26:05 DEBUG Config - Config.get(instrumenting, inversions) = 1
2021-09-23 19:26:05 DEBUG Config - Config.get(instrumenting, fixes) = true
2021-09-23 19:26:05 DEBUG Config - Config.get(instrumenting, hits) = true
2021-09-23 19:26:05 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
```