Program Structures and Algorithms

Spring 2023 (Section 3)

Assignment 3 – Insertion Sort

Name: Abhinav Choudhary

NUID: 002780326

Task

- To implement three methods of Timer.java class called repeat, getClock, and toMillisecs. These functions are to be implemented using Supplier and Consumer functions which take generic type T as input and U as output. Later, run benchmarks for timer functions called BenchmarkTest.java and TimerTest.java.
- To implement Insertion Sort in a way like described in Arrays.sort(). Helper classes may be used for the implementation if required. Lastly, run unit tests in InsertionSortTest.java file.
- To measure running time of Insertion Sort under different array ordering conditions. One, where array is already sorted and in correct order, next in which array is partially ordered, next in which array is randomly ordered and lastly where array is reverse ordered.

Relationship Conclusion

By running all the algorithms for increasing values on N multiple times, we found out that the time complexity of the Insertion Sort varies between O(n) and O(n^2). If the array is ordered, i.e., best case scenario, only one pass required to check and confirm whether the array is ordered or not. If the array is reverse ordered, i.e., worst case scenario, n^2 passes are required to sort the array. The other two implementations lie in between where partially ordered array generally performs better than random array.

Ordered < Partially Ordered < Random < Reverse Ordered

Evidence to support conclusion

After running all the four implementations for increasing values of N (size of input array) and for multiple runs, following raw run time (in milliseconds) were noted:

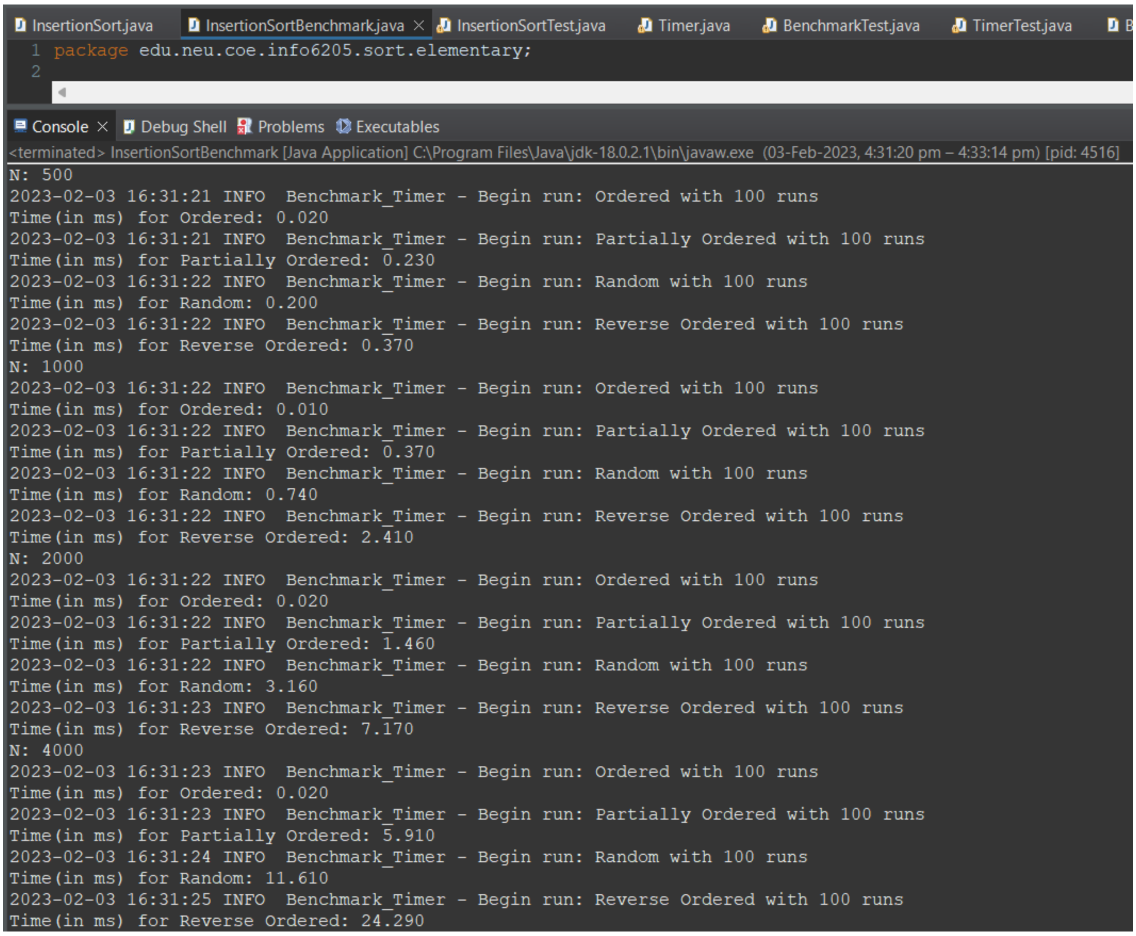
N	Runs	Time (in ms) - Ordered	Time (in ms) - Partially Ordered	Time (in ms) - Random	Time (in ms) - Reverse Ordered
500	100	0.015	0.224	0.208	0.375
1000	100	0.005	0.348	0.647	1.489
2000	100	0.01	1.402	3.1	5.958
4000	100	0.02	5.92	11.883	23.755
8000	100	0.03	22.765	47.334	95.584
16000	100	0.064	94.642	190.653	417.032

In this table, we can clearly see that in almost every value of N, the ordered array performs the best whereas reverse ordered array performs the worst. This gap further increases as N increases, to the point where for large values of N (8000+), run time complexity of ordered array is negligible in comparison to reverse ordered array.

Even for other implementations, partially ordered array's raw run time is towards the lower end when compared to reverse ordered array and random array lies somewhere in the middle.

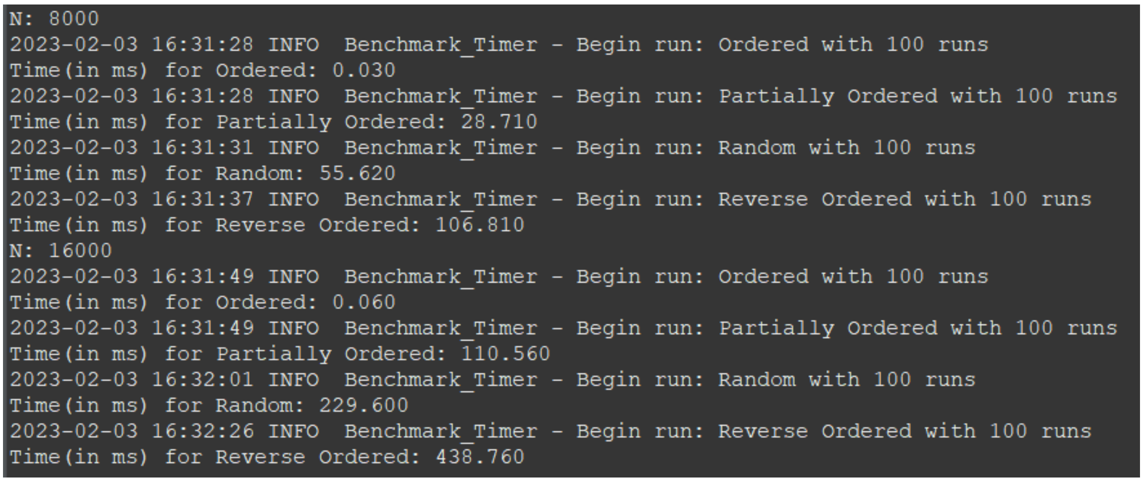
Following are the screenshots from the benchmark:

Note: Values in screenshot may differ than provided in table above. Separate instances were run for collecting data and taking screenshots.



```java
package edu.neu.coe.info6205.sort.elementary;
```

```
<terminated> InsertionSortBenchmark [Java Application] C:\Program Files\Java\jdk-18.0.2.1\bin\javaw.exe  (03-Feb-2023, 4:31:20 pm – 4:33:14 pm) [pid: 4516]
N: 500
2023-02-03 16:31:21 INFO  Benchmark_Timer - Begin run: Ordered with 100 runs
Time(in ms) for Ordered: 0.020
2023-02-03 16:31:21 INFO  Benchmark_Timer - Begin run: Partially Ordered with 100 runs
Time(in ms) for Partially Ordered: 0.230
2023-02-03 16:31:22 INFO  Benchmark_Timer - Begin run: Random with 100 runs
Time(in ms) for Random: 0.200
2023-02-03 16:31:22 INFO  Benchmark_Timer - Begin run: Reverse Ordered with 100 runs
Time(in ms) for Reverse Ordered: 0.370
N: 1000
2023-02-03 16:31:22 INFO  Benchmark_Timer - Begin run: Ordered with 100 runs
Time(in ms) for Ordered: 0.010
2023-02-03 16:31:22 INFO  Benchmark_Timer - Begin run: Partially Ordered with 100 runs
Time(in ms) for Partially Ordered: 0.370
2023-02-03 16:31:22 INFO  Benchmark_Timer - Begin run: Random with 100 runs
Time(in ms) for Random: 0.740
2023-02-03 16:31:22 INFO  Benchmark_Timer - Begin run: Reverse Ordered with 100 runs
Time(in ms) for Reverse Ordered: 2.410
N: 2000
2023-02-03 16:31:22 INFO  Benchmark_Timer - Begin run: Ordered with 100 runs
Time(in ms) for Ordered: 0.020
2023-02-03 16:31:22 INFO  Benchmark_Timer - Begin run: Partially Ordered with 100 runs
Time(in ms) for Partially Ordered: 1.460
2023-02-03 16:31:22 INFO  Benchmark_Timer - Begin run: Random with 100 runs
Time(in ms) for Random: 3.160
2023-02-03 16:31:23 INFO  Benchmark_Timer - Begin run: Reverse Ordered with 100 runs
Time(in ms) for Reverse Ordered: 7.170
N: 4000
2023-02-03 16:31:23 INFO  Benchmark_Timer - Begin run: Ordered with 100 runs
Time(in ms) for Ordered: 0.020
2023-02-03 16:31:23 INFO  Benchmark_Timer - Begin run: Partially Ordered with 100 runs
Time(in ms) for Partially Ordered: 5.910
2023-02-03 16:31:24 INFO  Benchmark_Timer - Begin run: Random with 100 runs
Time(in ms) for Random: 11.610
2023-02-03 16:31:25 INFO  Benchmark_Timer - Begin run: Reverse Ordered with 100 runs
Time(in ms) for Reverse Ordered: 24.290
```



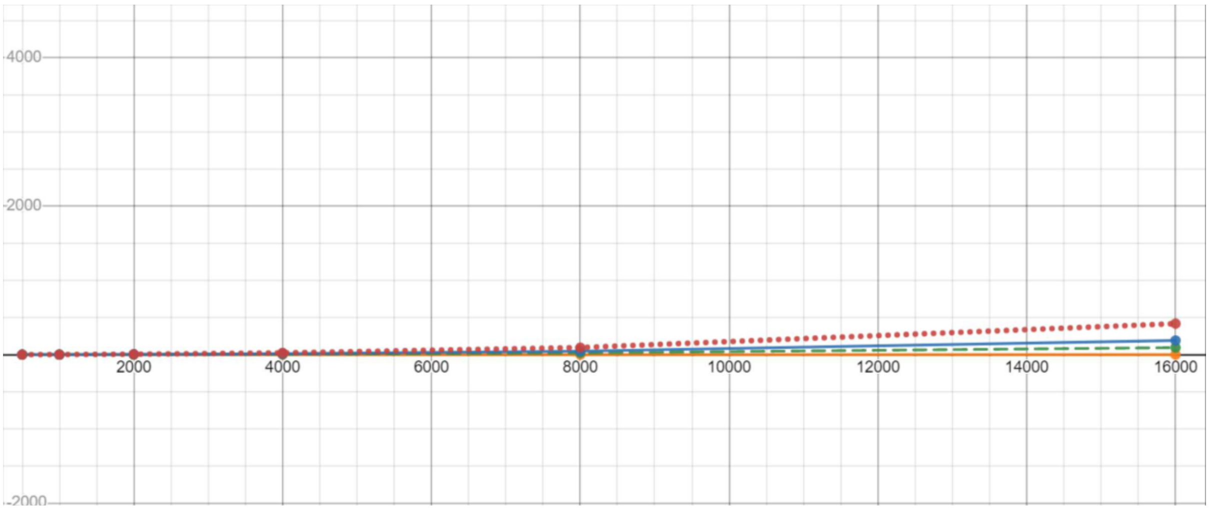
```
N: 8000
2023-02-03 16:31:28 INFO  Benchmark_Timer - Begin run: Ordered with 100 runs
Time(in ms) for Ordered: 0.030
2023-02-03 16:31:28 INFO  Benchmark_Timer - Begin run: Partially Ordered with 100 runs
Time(in ms) for Partially Ordered: 28.710
2023-02-03 16:31:31 INFO  Benchmark_Timer - Begin run: Random with 100 runs
Time(in ms) for Random: 55.620
2023-02-03 16:31:37 INFO  Benchmark_Timer - Begin run: Reverse Ordered with 100 runs
Time(in ms) for Reverse Ordered: 106.810
N: 16000
2023-02-03 16:31:49 INFO  Benchmark_Timer - Begin run: Ordered with 100 runs
Time(in ms) for Ordered: 0.060
2023-02-03 16:31:49 INFO  Benchmark_Timer - Begin run: Partially Ordered with 100 runs
Time(in ms) for Partially Ordered: 110.560
2023-02-03 16:32:01 INFO  Benchmark_Timer - Begin run: Random with 100 runs
Time(in ms) for Random: 229.600
2023-02-03 16:32:26 INFO  Benchmark_Timer - Begin run: Reverse Ordered with 100 runs
Time(in ms) for Reverse Ordered: 438.760
```

With these timing observations, we can conclude that for large values of N, ordered array performs the best followed by partially ordered, random, and reverse ordered array at the end.

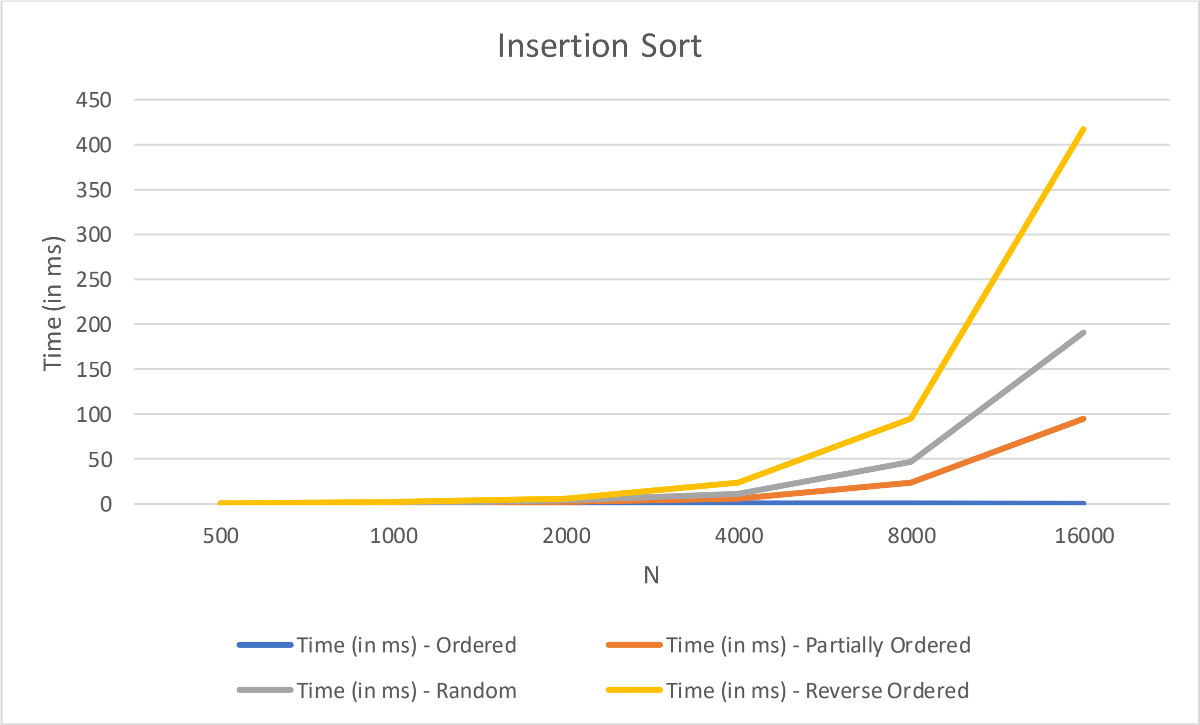
Graphical Representation

Following is the graph plotted using the timing observations, with time (in ms) along the y axis and N along the x axis. Solid Orange line represents ordered array, Green Dashed line represents partially ordered array, Blue Solid line represents random array, and Red Dotted line represents reverse ordered array.

(Desmos Graphing Calculator was used to plot the points and create the graph: Desmos | Graphing Calculator)



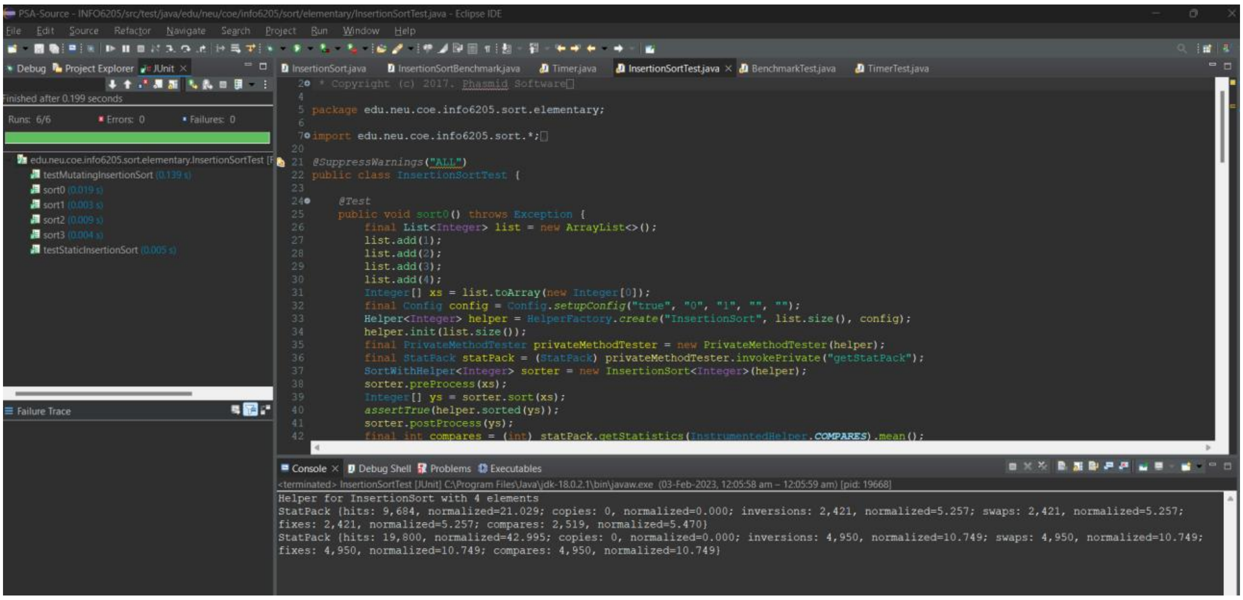
Graph created with Excel:



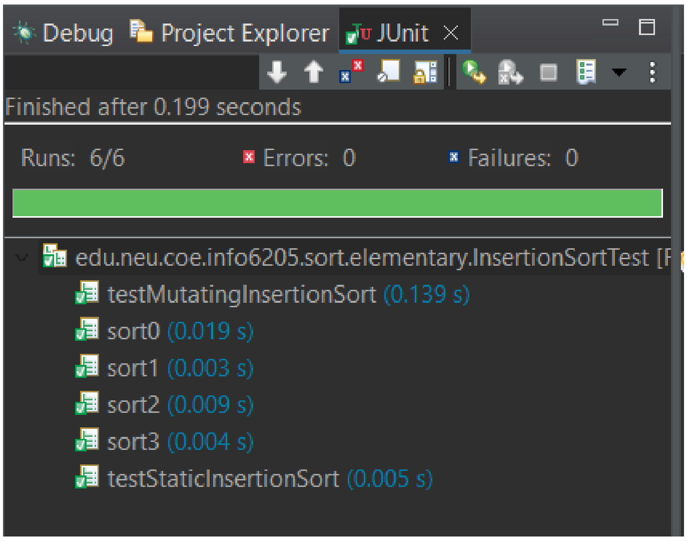
Unit Test

Below is the screenshot of runs of different test files:

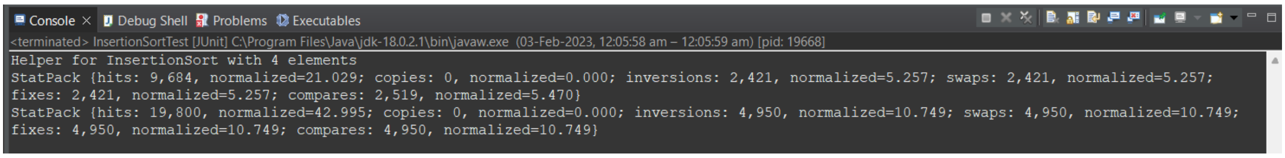
InsertionSortTest.java


PSA-Source - INFO6205/src/test/java/edu/neu/coe/info6205/sort/elementary/InsertionSortTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Debug Project Explorer JUnit
InsertionSort.java InsertionSortBenchmark.java Timer.java InsertionSortTest.java BenchmarkTest.java TimerTest.java
Failure Trace
Console Debug Shell Problems Executables

```java
 * Copyright (c) 2017. Phasmid Software

package edu.neu.coe.info6205.sort.elementary;

import edu.neu.coe.info6205.sort.*;

@SuppressWarnings("ALL")
public class InsertionSortTest {

    @Test
    public void sort0() throws Exception {
        final List<Integer> list = new ArrayList<>();
        list.add(1);
        list.add(2);
        list.add(3);
        list.add(4);
        Integer[] xs = list.toArray(new Integer[0]);
        final Config config = Config.setupConfig("true", "0", "1", "", "");
        Helper<Integer> helper = HelperFactory.create("InsertionSort", list.size(), config);
        helper.init(list.size());
        final PrivateMethodTester privateMethodTester = new PrivateMethodTester(helper);
        final StatPack statPack = (StatPack) privateMethodTester.invokePrivate("getStatPack");
        SortWithHelper<Integer> sorter = new InsertionSort<Integer>(helper);
        sorter.preProcess(xs);
        Integer[] ys = sorter.sort(xs);
        assertTrue(helper.sorted(ys));
        sorter.postProcess(ys);
        final int compares = (int) statPack.getStatistics(InstrumentedHelper.COMPARES).mean();
```

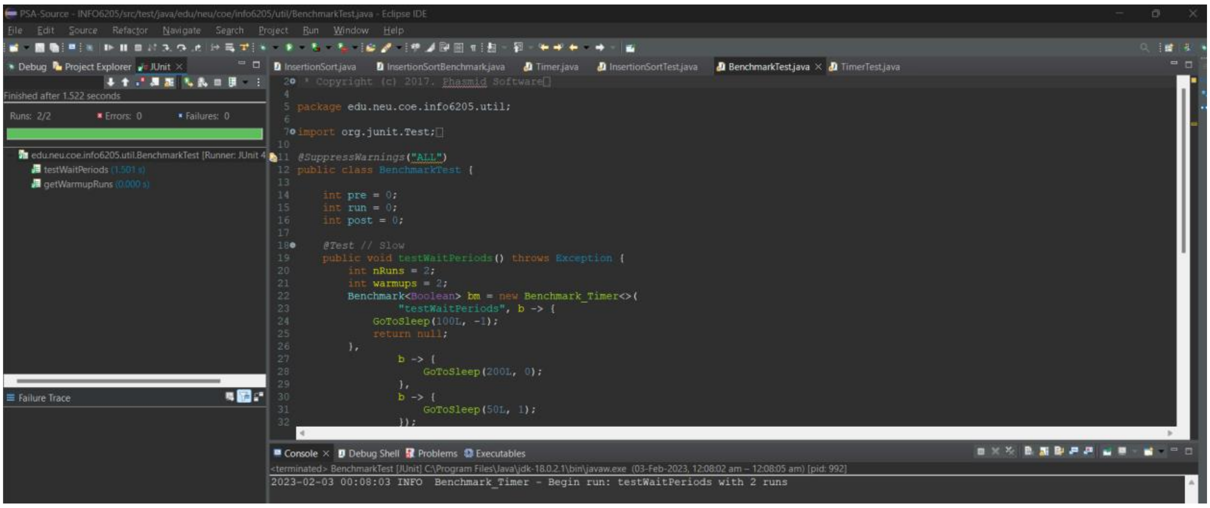


```
Finished after 0.199 seconds

Runs: 6/6        Errors: 0        Failures: 0

edu.neu.coe.info6205.sort.elementary.InsertionSortTest [F
    testMutatingInsertionSort (0.139 s)
    sort0 (0.019 s)
    sort1 (0.003 s)
    sort2 (0.009 s)
    sort3 (0.004 s)
    testStaticInsertionSort (0.005 s)
```



```
<terminated> InsertionSortTest [JUnit] C:\Program Files\Java\jdk-18.0.2.1\bin\javaw.exe (03-Feb-2023, 12:05:58 am – 12:05:59 am) [pid: 19668]
Helper for InsertionSort with 4 elements
StatPack {hits: 9,684, normalized=21.029; copies: 0, normalized=0.000; inversions: 2,421, normalized=5.257; swaps: 2,421, normalized=5.257;
fixes: 2,421, normalized=5.257; compares: 2,519, normalized=5.470}
StatPack {hits: 19,800, normalized=42.995; copies: 0, normalized=0.000; inversions: 4,950, normalized=10.749; swaps: 4,950, normalized=10.749;
fixes: 4,950, normalized=10.749; compares: 4,950, normalized=10.749}
```

BenchmarkTest.java



PSA-Source - INFO6205/src/test/java/edu/neu/coe/info6205/util/BenchmarkTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Debug Project Explorer JUnit
InsertionSort.java InsertionSortBenchmark.java Timer.java InsertionSortTest.java BenchmarkTest.java TimerTest.java
Failure Trace
Console Debug Shell Problems Executables

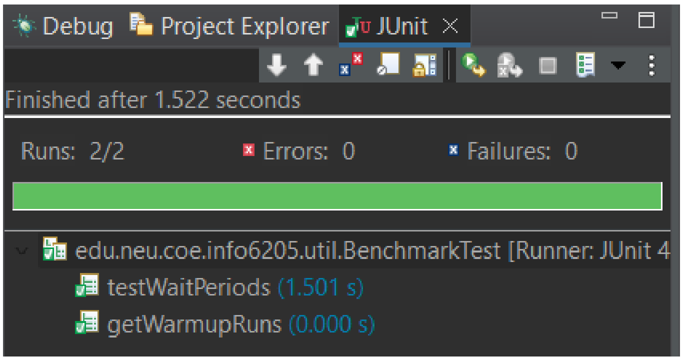
Finished after 1.522 seconds

Runs: 2/2 Errors: 0 Failures: 0

- edu.neu.coe.info6205.util.BenchmarkTest [Runner: JUnit 4
  - testWaitPeriods (1.501 s)
  - getWarmupRuns (0.000 s)

```java
 * Copyright (c) 2017. Phasmid Software

package edu.neu.coe.info6205.util;

import org.junit.Test;

@SuppressWarnings("ALL")
public class BenchmarkTest {

    int pre = 0;
    int run = 0;
    int post = 0;

    @Test // Slow
    public void testWaitPeriods() throws Exception {
        int nRuns = 2;
        int warmups = 2;
        Benchmark<Boolean> bm = new Benchmark_Timer<>(
                "testWaitPeriods", b -> {
            GoToSleep(100L, -1);
            return null;
        },
                b -> {
                    GoToSleep(200L, 0);
                },
                b -> {
                    GoToSleep(50L, 1);
                });
```

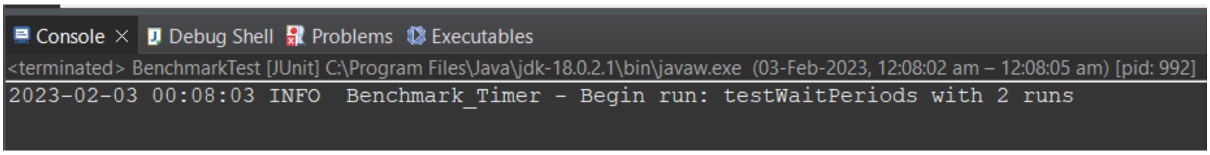
```
<terminated> BenchmarkTest [JUnit] C:\Program Files\Java\jdk-18.0.2.1\bin\javaw.exe (03-Feb-2023, 12:08:02 am – 12:08:05 am) [pid: 992]
2023-02-03 00:08:03 INFO  Benchmark_Timer - Begin run: testWaitPeriods with 2 runs
```



Finished after 1.522 seconds

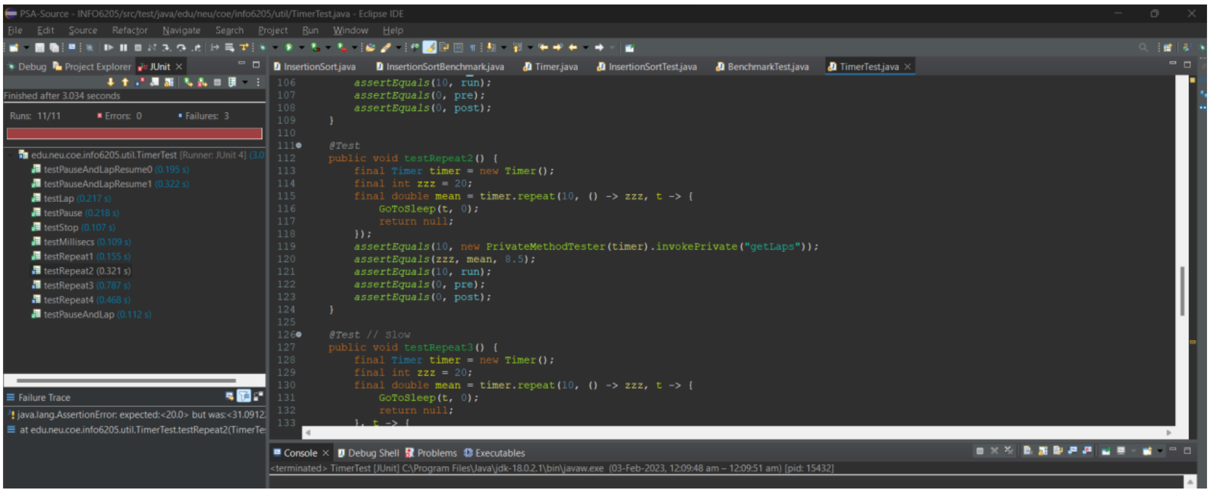
Runs: 2/2 Errors: 0 Failures: 0

- edu.neu.coe.info6205.util.BenchmarkTest [Runner: JUnit 4
  - testWaitPeriods (1.501 s)
  - getWarmupRuns (0.000 s)



```
<terminated> BenchmarkTest [JUnit] C:\Program Files\Java\jdk-18.0.2.1\bin\javaw.exe (03-Feb-2023, 12:08:02 am – 12:08:05 am) [pid: 992]
2023-02-03 00:08:03 INFO  Benchmark_Timer - Begin run: testWaitPeriods with 2 runs
```

TimerTest.java



PSA-Source - INFO6205/src/test/java/edu/neu/coe/info6205/util/TimerTest.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Debug Project Explorer JUnit
InsertionSort.java InsertionSortBenchmark.java Timer.java InsertionSortTest.java BenchmarkTest.java TimerTest.java
Failure Trace
Console Debug Shell Problems Executables

Finished after 3.034 seconds

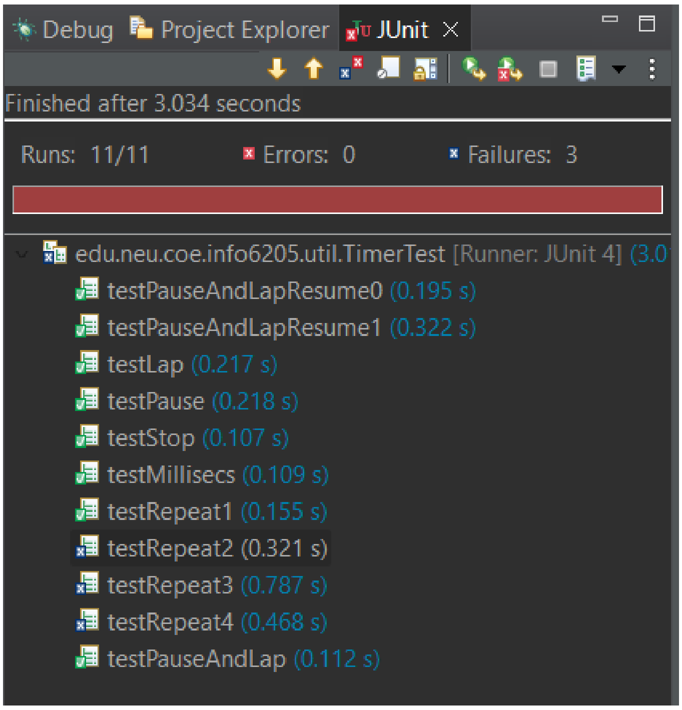
Runs: 11/11 Errors: 0 Failures: 3

- edu.neu.coe.info6205.util.TimerTest [Runner: JUnit 4] (3.0
  - testPauseAndLapResume0 (0.195 s)
  - testPauseAndLapResume1 (0.322 s)
  - testLap (0.217 s)
  - testPause (0.218 s)
  - testStop (0.107 s)
  - testMillisecs (0.109 s)
  - testRepeat1 (0.155 s)
  - testRepeat2 (0.321 s)
  - testRepeat3 (0.787 s)
  - testRepeat4 (0.468 s)
  - testPauseAndLap (0.112 s)

```
java.lang.AssertionError: expected:<20.0> but was:<31.0912
at edu.neu.coe.info6205.util.TimerTest.testRepeat2(TimerTe
```

```java
        assertEquals(10, run);
        assertEquals(0, pre);
        assertEquals(0, post);
    }

    @Test
    public void testRepeat2() {
        final Timer timer = new Timer();
        final int zzz = 20;
        final double mean = timer.repeat(10, () -> zzz, t -> {
            GoToSleep(t, 0);
            return null;
        });
        assertEquals(10, new PrivateMethodTester(timer).invokePrivate("getLaps"));
        assertEquals(zzz, mean, 8.5);
        assertEquals(10, run);
        assertEquals(0, pre);
        assertEquals(0, post);
    }

    @Test // Slow
    public void testRepeat3() {
        final Timer timer = new Timer();
        final int zzz = 20;
        final double mean = timer.repeat(10, () -> zzz, t -> {
            GoToSleep(t, 0);
            return null;
        }, t -> {
```

```
<terminated> TimerTest [JUnit] C:\Program Files\Java\jdk-18.0.2.1\bin\javaw.exe (03-Feb-2023, 12:09:48 am – 12:09:51 am) [pid: 15432]
```



```
Finished after 3.034 seconds

Runs:  11/11          Errors:  0          Failures:  3

edu.neu.coe.info6205.util.TimerTest [Runner: JUnit 4] (3.0
    testPauseAndLapResume0 (0.195 s)
    testPauseAndLapResume1 (0.322 s)
    testLap (0.217 s)
    testPause (0.218 s)
    testStop (0.107 s)
    testMillisecs (0.109 s)
    testRepeat1 (0.155 s)
    testRepeat2 (0.321 s)
    testRepeat3 (0.787 s)
    testRepeat4 (0.468 s)
    testPauseAndLap (0.112 s)
```