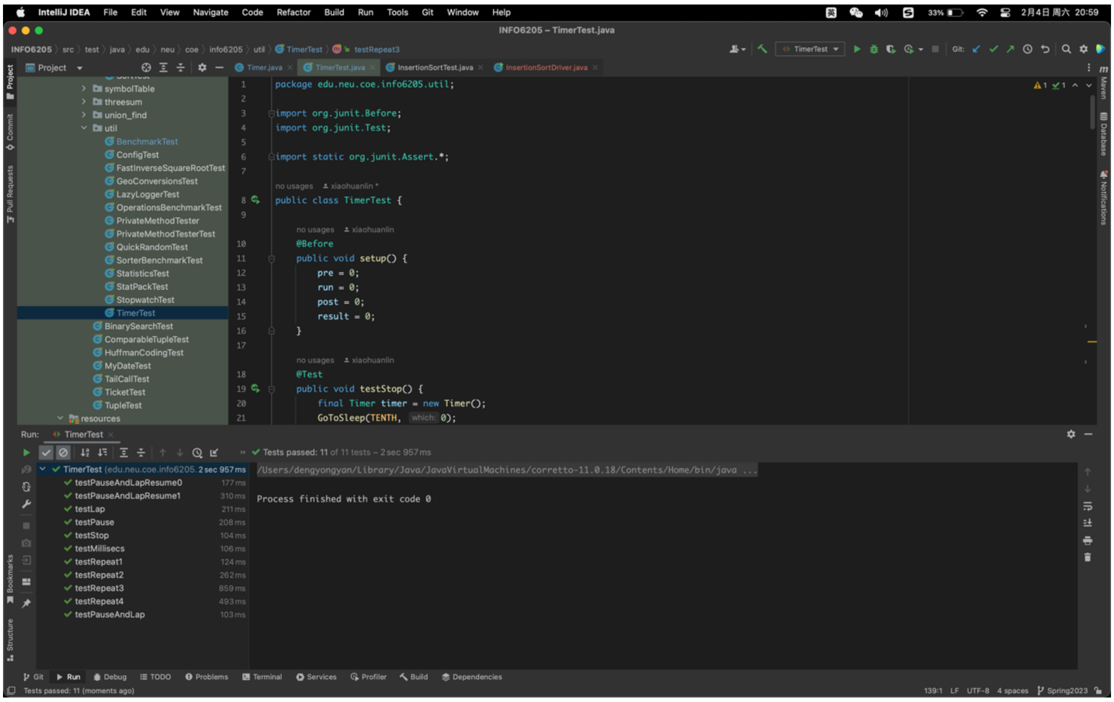
Assignment03 - Benchmark Timer for InsertionSort

Here is the result of unit tests for Timer.class and Benchmark_Timer.class:



INFO6205 – TimerTest.java

```java
1   package edu.neu.coe.info6205.util;
2
3   import org.junit.Before;
4   import org.junit.Test;
5
6   import static org.junit.Assert.*;
7
8   public class TimerTest {
9
10      @Before
11      public void setup() {
12          pre = 0;
13          run = 0;
14          post = 0;
15          result = 0;
16      }
17
18      @Test
19      public void testStop() {
20          final Timer timer = new Timer();
21          GoToSleep(TENTH, 0);
```

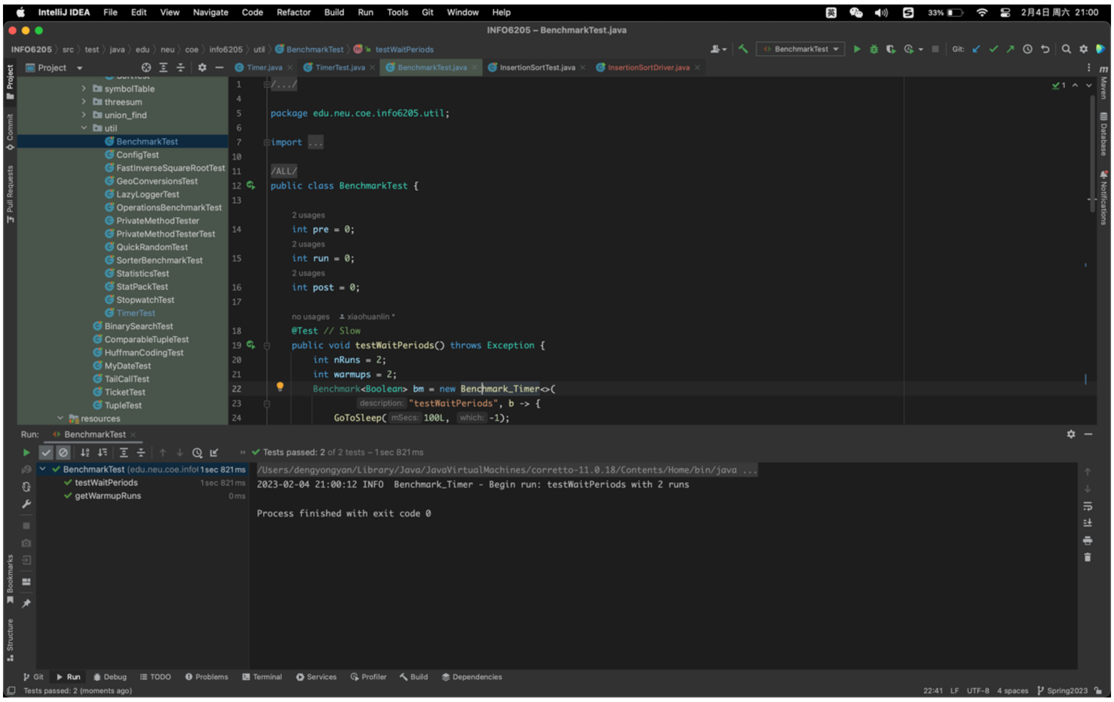
Tests passed: 11 of 11 tests – 2 sec 957 ms

```
TimerTest (edu.neu.coe.info6205. 2 sec 957 ms
  testPauseAndLapResume0   177 ms
  testPauseAndLapResume1   310 ms
  testLap                  211 ms
  testPause                208 ms
  testStop                 104 ms
  testMillisecs            106 ms
  testRepeat1              124 ms
  testRepeat2              262 ms
  testRepeat3              859 ms
  testRepeat4              493 ms
  testPauseAndLap          103 ms
```

```
/Users/dengyongyan/Library/Java/JavaVirtualMachines/corretto-11.0.18/Contents/Home/bin/java ...

Process finished with exit code 0
```

Tests passed: 11 (moments ago)

For Timertest, variable changed in **line 139** & **161**: *assertEquals(10, run - 10);*



INFO6205 – BenchmarkTest.java

```java
1   /.../
4
5   package edu.neu.coe.info6205.util;
6
7   import ...
10
11  /ALL/
12  public class BenchmarkTest {
13
14      int pre = 0;
15      int run = 0;
16      int post = 0;
17
18      @Test // Slow
19      public void testWaitPeriods() throws Exception {
20          int nRuns = 2;
21          int warmups = 2;
22          Benchmark<Boolean> bm = new Benchmark_Timer<>(
23                  "testWaitPeriods", b -> {
24              GoToSleep(100L, -1);
```

Tests passed: 2 of 2 tests – 1 sec 821 ms

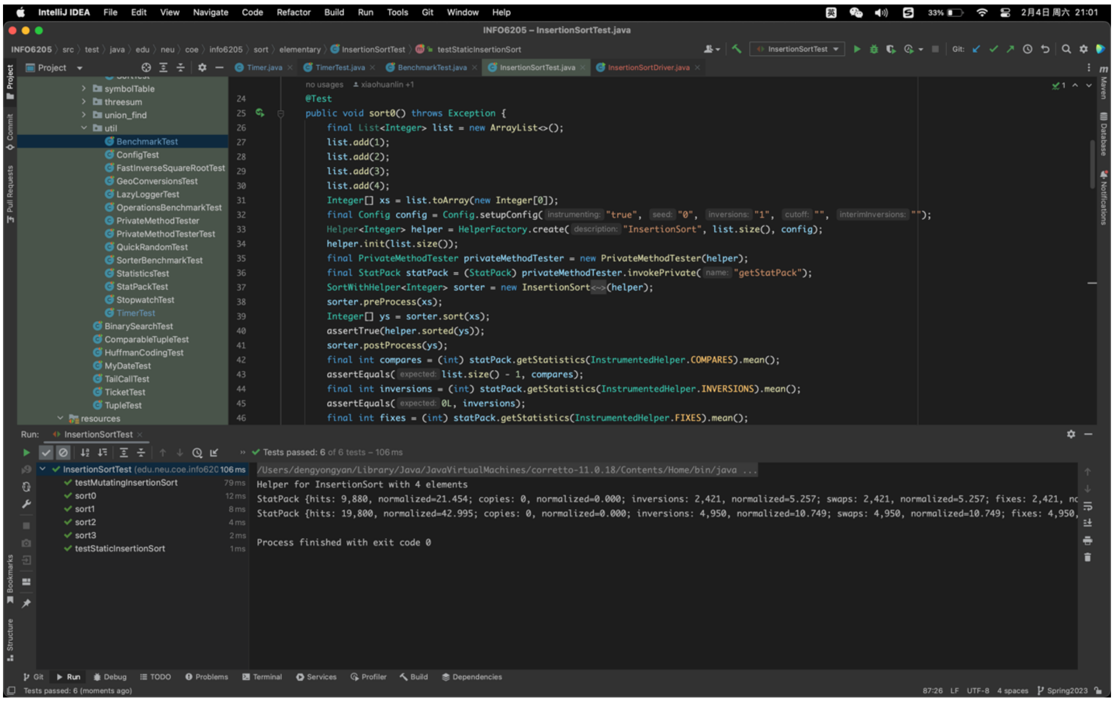
```
BenchmarkTest (edu.neu.coe.info( 1 sec 821 ms
  testWaitPeriods   1 sec 821 ms
  getWarmupRuns     0 ms
```

```
/Users/dengyongyan/Library/Java/JavaVirtualMachines/corretto-11.0.18/Contents/Home/bin/java ...
2023-02-04 21:00:12 INFO  Benchmark_Timer - Begin run: testWaitPeriods with 2 runs

Process finished with exit code 0
```

Tests passed: 2 (moments ago)

For BenchmarkTest, variable changed in **line35**: *assertEquals(nRuns + warmups, run - 2);*

Here is the unit test for Insertsort.class:



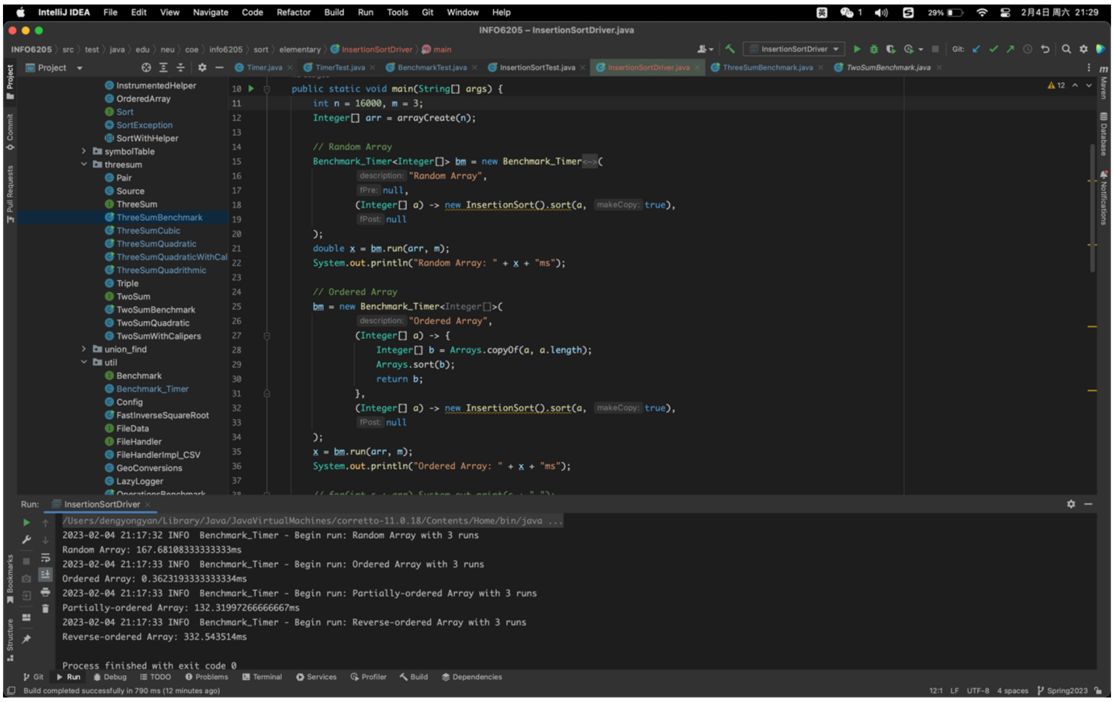
```java
@Test
public void sort0() throws Exception {
    final List<Integer> list = new ArrayList<>();
    list.add(1);
    list.add(2);
    list.add(3);
    list.add(4);
    Integer[] xs = list.toArray(new Integer[0]);
    final Config config = Config.setupConfig("true", "0", "1", "", "");
    Helper<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<~>(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();
    assertEquals(list.size() - 1, compares);
    final int inversions = (int) statPack.getStatistics(InstrumentedHelper.INVERSIONS).mean();
    assertEquals(0L, inversions);
    final int fixes = (int) statPack.getStatistics(InstrumentedHelper.FIXES).mean();
```

Tests passed: 6 of 6 tests – 106 ms

```
✓ InsertionSortTest (edu.neu.coe.info620 106 ms
    ✓ testMutatingInsertionSort   79 ms
    ✓ sort0                       12 ms
    ✓ sort1                        8 ms
    ✓ sort2                        4 ms
    ✓ sort3                        2 ms
    ✓ testStaticInsertionSort      1 ms
```

```
/Users/dengyongyan/Library/Java/JavaVirtualMachines/corretto-11.0.18/Contents/Home/bin/java ...
Helper for InsertionSort with 4 elements
StatPack {hits: 9,880, normalized=21.454; copies: 0, normalized=0.000; inversions: 2,421, normalized=5.257; swaps: 2,421, normalized=5.257; fixes: 2,421, nc
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,

Process finished with exit code 0
```

Screenshot for test/edu/neu/coe/info6205/sort/elementary/InsertionSortDriver.class



```java
public static void main(String[] args) {
    int n = 16000, m = 3;
    Integer[] arr = arrayCreate(n);

    // Random Array
    Benchmark_Timer<Integer[]> bm = new Benchmark_Timer<~>(
            "Random Array",
            null,
            (Integer[] a) -> new InsertionSort().sort(a, true),
            null
    );
    double x = bm.run(arr, m);
    System.out.println("Random Array: " + x + "ms");

    // Ordered Array
    bm = new Benchmark_Timer<Integer[]>(
            "Ordered Array",
            (Integer[] a) -> {
                Integer[] b = Arrays.copyOf(a, a.length);
                Arrays.sort(b);
                return b;
            },
            (Integer[] a) -> new InsertionSort().sort(a, true),
            null
    );
    x = bm.run(arr, m);
    System.out.println("Ordered Array: " + x + "ms");
```

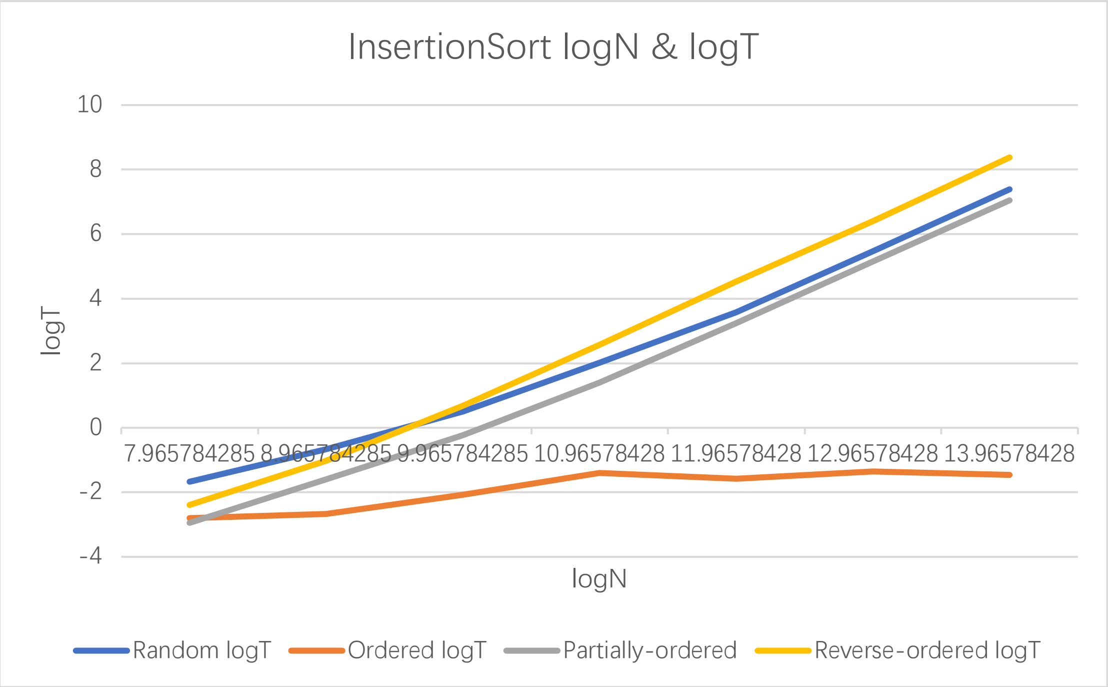
```
/Users/dengyongyan/Library/Java/JavaVirtualMachines/corretto-11.0.18/Contents/Home/bin/java ...
2023-02-04 21:17:32 INFO  Benchmark_Timer - Begin run: Random Array with 3 runs
Random Array: 167.68108333333333ms
2023-02-04 21:17:33 INFO  Benchmark_Timer - Begin run: Ordered Array with 3 runs
Ordered Array: 0.36231933333333334ms
2023-02-04 21:17:33 INFO  Benchmark_Timer - Begin run: Partially-ordered Array with 3 runs
Partially-ordered Array: 132.31997266666667ms
2023-02-04 21:17:33 INFO  Benchmark_Timer - Begin run: Reverse-ordered Array with 3 runs
Reverse-ordered Array: 332.543514ms

Process finished with exit code 0
```

Build completed successfully in 790 ms (12 minutes ago)

With the Benchmark_Timer, here is the relationship between InsertionSort's time T and the array size N:

N	Random	Ordered	Partially-ordered	Reverse-ordered
250	0.31365047	0.14346747	0.12958252	0.19029619
500	0.62686924	0.15721996	0.32898498	0.48964844
1000	1.4139209	0.23859375	0.85563545	1.6055876
2000	4.0603083	0.3801876	2.6194583	5.947829
4000	11.9058164	0.3330168	9.4438414	23.0846914
8000	44.436167	0.390847333	35.65701333	84.00815233
16000	167.6810833	0.362319333	132.3199727	332.543514

and the graph between InsertionSort's logN and logT would be like:



Conclusion:

According to the sheet and graph above, it is clear to see that InsertionSort has O(logN) time complexity. In the best condition(ordered array), InsertionSort's time complexity is nearly O(1).