Keyur Ashokbhai Barot (NU ID: 001568664)

Program Structures and Algorithms

Fall 2021

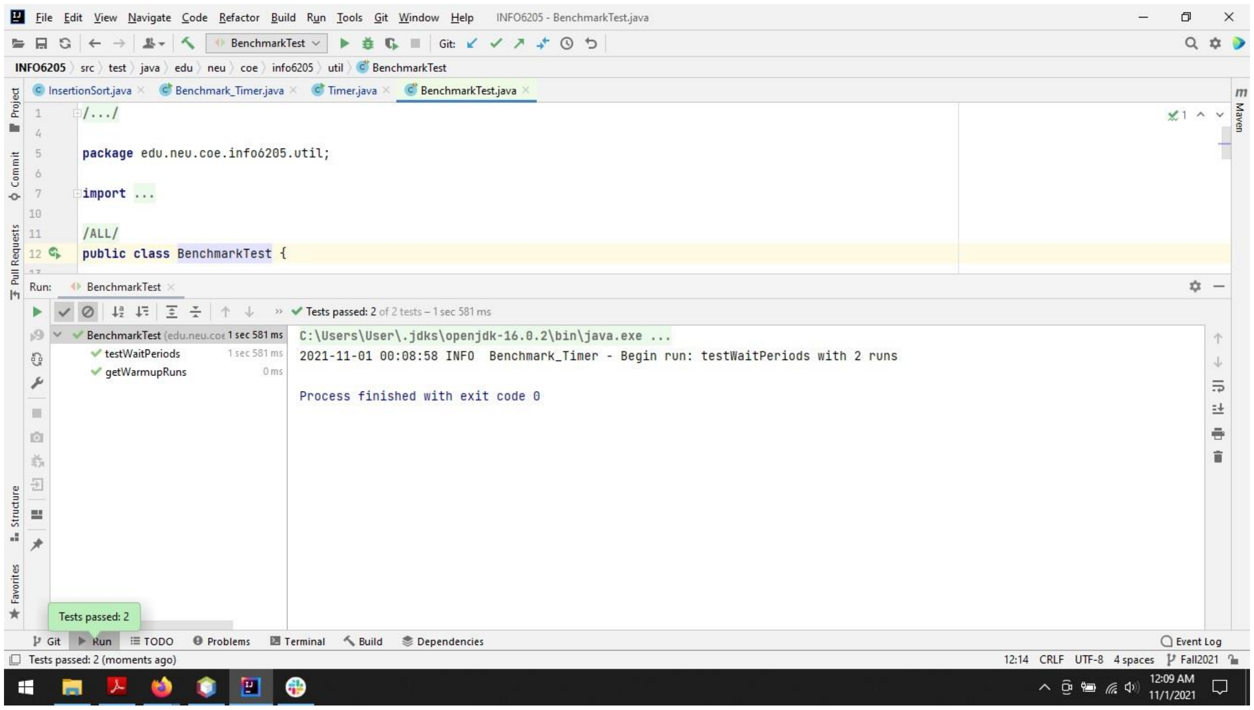
Assignment No.2 (Benchmark)

- **Task:**

To Calculate benchmark timing for Insertion sort for differently sorted arrays (random, sorted, partially sorted and reverse ordered).

1. Part-1: Implement methods in Timer class.
2. Part-2: Implement InsertionSort in InsertionSort class.
3. Part-3: Implement main class in Benchmark_Timer class where mean time for random, sorted, partially sorted and reversed array for various sizes will be calculated.

- **Test Cases for Benchmark Test:**



```java
/.../

package edu.neu.coe.info6205.util;

import ...

/ALL/
public class BenchmarkTest {
```

Tests passed: 2 of 2 tests – 1 sec 581 ms

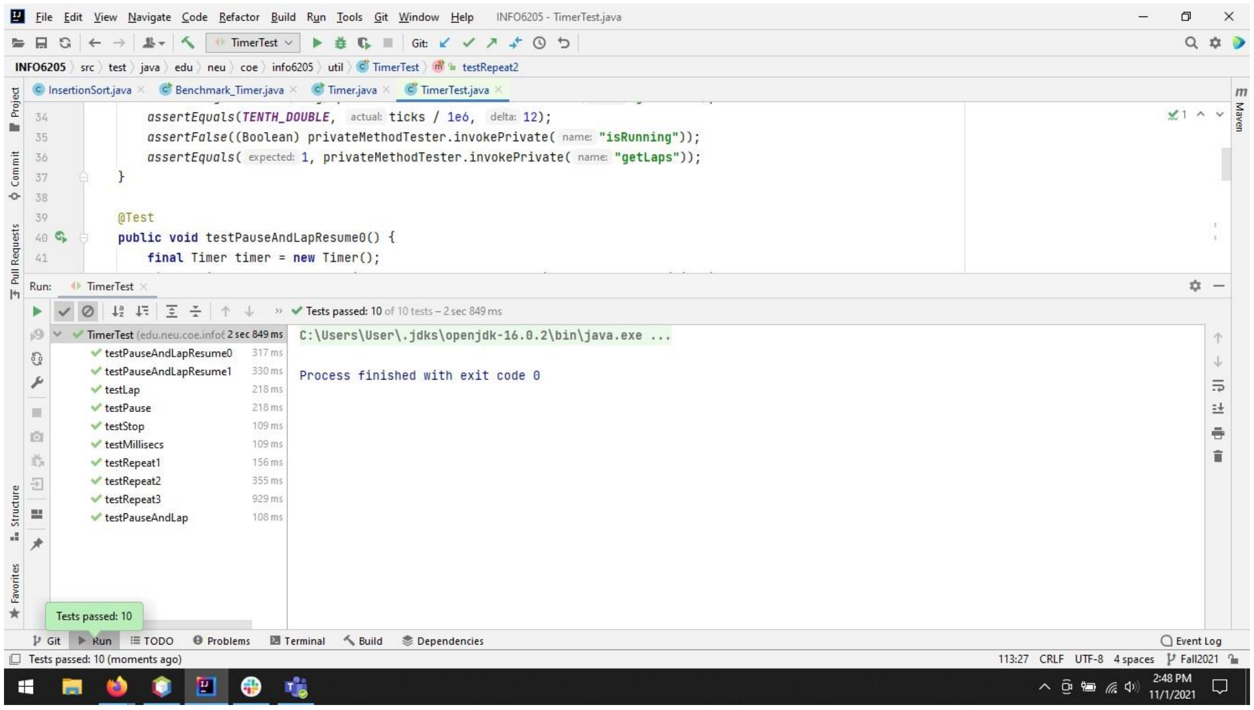
```
BenchmarkTest (edu.neu.coe  1 sec 581 ms
  testWaitPeriods           1 sec 581 ms
  getWarmupRuns                     0 ms
```

```
C:\Users\User\.jdks\openjdk-16.0.2\bin\java.exe ...
2021-11-01 00:08:58 INFO  Benchmark_Timer - Begin run: testWaitPeriods with 2 runs

Process finished with exit code 0
```

Tests passed: 2 (moments ago)

- **Test Cases for TimerTest:**



```
        assertEquals(TENTH_DOUBLE, actual: ticks / 1e6, delta: 12);
        assertFalse((Boolean) privateMethodTester.invokePrivate( name: "isRunning"));
        assertEquals( expected: 1, privateMethodTester.invokePrivate( name: "getLaps"));
    }

    @Test
    public void testPauseAndLapResume0() {
        final Timer timer = new Timer();
```

Tests passed: 10 of 10 tests – 2 sec 849 ms

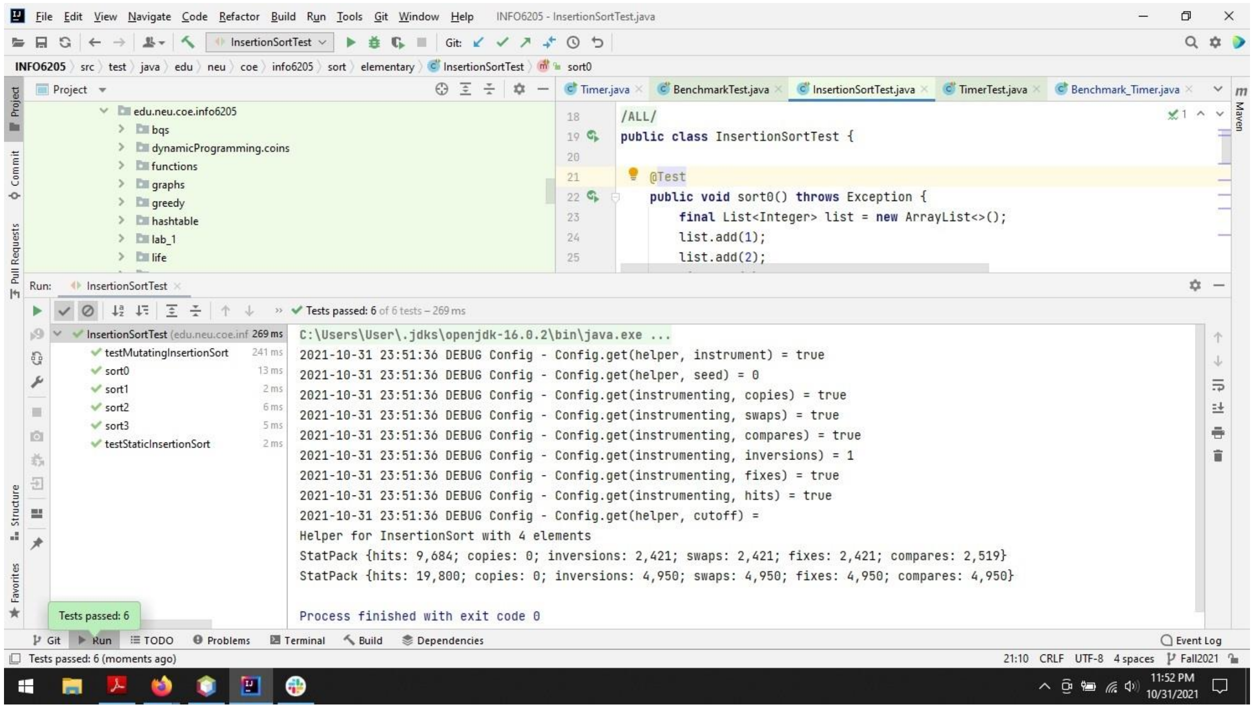
```
TimerTest (edu.neu.coe.infof 2 sec 849 ms
  testPauseAndLapResume0   317 ms
  testPauseAndLapResume1   330 ms
  testLap                  218 ms
  testPause                218 ms
  testStop                 109 ms
  testMillisecs            109 ms
  testRepeat1              156 ms
  testRepeat2              355 ms
  testRepeat3              929 ms
  testPauseAndLap          108 ms
```

```
C:\Users\User\.jdks\openjdk-16.0.2\bin\java.exe ...

Process finished with exit code 0
```

Tests passed: 10 (moments ago)

- **Test Cases for InsertionSortTest:**



```
/ALL/
public class InsertionSortTest {

    @Test
    public void sort0() throws Exception {
        final List<Integer> list = new ArrayList<>();
        list.add(1);
        list.add(2);
```

Tests passed: 6 of 6 tests – 269 ms

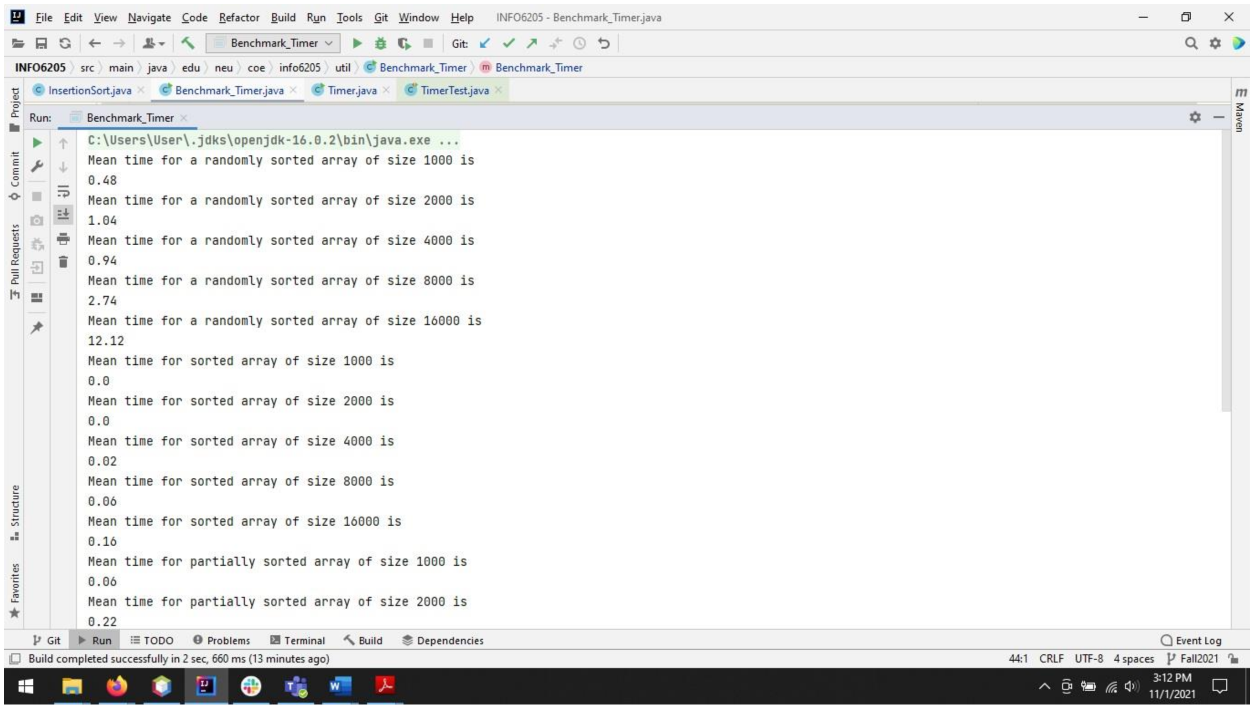
```
InsertionSortTest (edu.neu.coe.inf 269 ms
  testMutatingInsertionSort   241 ms
  sort0                        13 ms
  sort1                         2 ms
  sort2                         6 ms
  sort3                         5 ms
  testStaticInsertionSort       2 ms
```

```
C:\Users\User\.jdks\openjdk-16.0.2\bin\java.exe ...
2021-10-31 23:51:36 DEBUG Config - Config.get(helper, instrument) = true
2021-10-31 23:51:36 DEBUG Config - Config.get(helper, seed) = 0
2021-10-31 23:51:36 DEBUG Config - Config.get(instrumenting, copies) = true
2021-10-31 23:51:36 DEBUG Config - Config.get(instrumenting, swaps) = true
2021-10-31 23:51:36 DEBUG Config - Config.get(instrumenting, compares) = true
2021-10-31 23:51:36 DEBUG Config - Config.get(instrumenting, inversions) = 1
2021-10-31 23:51:36 DEBUG Config - Config.get(instrumenting, fixes) = true
2021-10-31 23:51:36 DEBUG Config - Config.get(instrumenting, hits) = true
2021-10-31 23:51:36 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
```

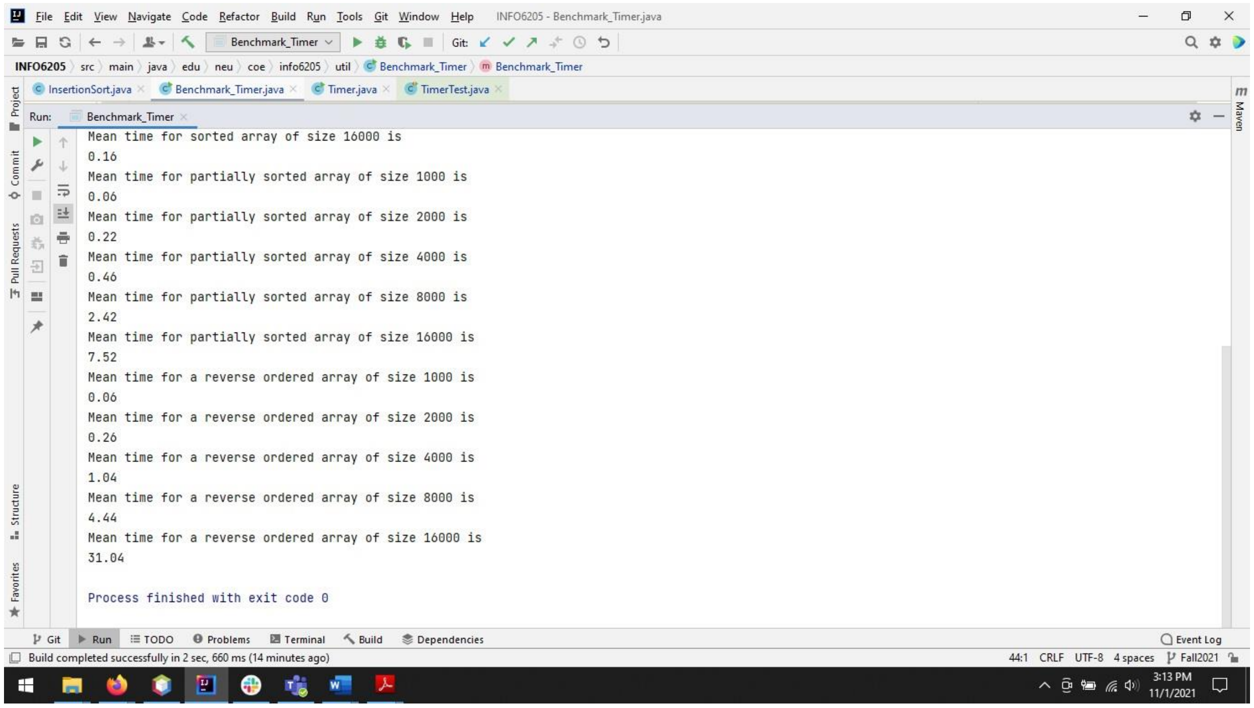
Tests passed: 6 (moments ago)

- **Output:**



```
C:\Users\User\.jdks\openjdk-16.0.2\bin\java.exe ...
Mean time for a randomly sorted array of size 1000 is
0.48
Mean time for a randomly sorted array of size 2000 is
1.04
Mean time for a randomly sorted array of size 4000 is
0.94
Mean time for a randomly sorted array of size 8000 is
2.74
Mean time for a randomly sorted array of size 16000 is
12.12
Mean time for sorted array of size 1000 is
0.0
Mean time for sorted array of size 2000 is
0.0
Mean time for sorted array of size 4000 is
0.02
Mean time for sorted array of size 8000 is
0.06
Mean time for sorted array of size 16000 is
0.16
Mean time for partially sorted array of size 1000 is
0.06
Mean time for partially sorted array of size 2000 is
0.22
```

Build completed successfully in 2 sec, 660 ms (13 minutes ago)



```
Mean time for sorted array of size 16000 is
0.16
Mean time for partially sorted array of size 1000 is
0.06
Mean time for partially sorted array of size 2000 is
0.22
Mean time for partially sorted array of size 4000 is
0.46
Mean time for partially sorted array of size 8000 is
2.42
Mean time for partially sorted array of size 16000 is
7.52
Mean time for a reverse ordered array of size 1000 is
0.06
Mean time for a reverse ordered array of size 2000 is
0.26
Mean time for a reverse ordered array of size 4000 is
1.04
Mean time for a reverse ordered array of size 8000 is
4.44
Mean time for a reverse ordered array of size 16000 is
31.04

Process finished with exit code 0
```

Build completed successfully in 2 sec, 660 ms (14 minutes ago)

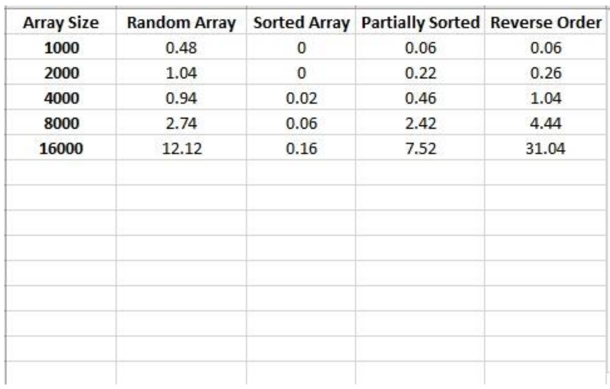
- **Relationship Conclusion:**

After performing tests on different types of arrays like: Sorted Array, Partially Sorted Array, Randomly Sorted Array and Reverse Ordered Array we can say that Insertion Sort takes the least time for sorting an already Sorted Array followed by Partially Sorted Array, which is followed by Randomly Sorted Array and finally, the most time is taken by a Reverse Ordered Array to be sorted.

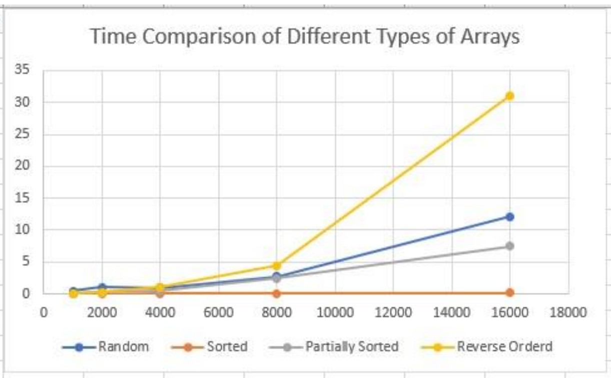
The order of time taken can be summarized in the following way:

Sorted < Partially Sorted < Randomly Sorted < Reverse Ordered

- **Evidence:**



Array Size	Random Array	Sorted Array	Partially Sorted	Reverse Order
1000	0.48	0	0.06	0.06
2000	1.04	0	0.22	0.26
4000	0.94	0.02	0.46	1.04
8000	2.74	0.06	2.42	4.44
16000	12.12	0.16	7.52	31.04


Time Comparison of Different Types of Arrays
Random
Sorted
Partially Sorted
Reverse Orderd