Program Structures and Algorithms
Spring 2023(SEC –01)

NAME: **Harshini Venkata Chalam**
NUID: **002934047**
Assignment 3

Task:
Part 1 - To implement three methods (*repeat*, *getClock*, and *toMillisecs*) of a class called *Timer* and checking the implementation by running the unit tests in *BenchmarkTest* and *TimerTest*.

Part 2-Implementation of InsertionSort *and testing using InsertionSortTest*.

Part 3- Implementation of main program to run the benchmarks using four different initial array ordering situations: random, ordered, partially ordered and reverse-ordered. To also draw conclusions from your observations regarding the order of growth.

Relationship Conclusion:

The time complexity of Insertion sort for random arrays is O(n^2) where n is the number of elements in the array. When the input array is already ordered the time complexity will be O(n) as no swapping of elements will be required which is the best-case scenario but when the input array is reverse ordered the time complexity is O(n^2) which is the worst-case scenario.
In the partially ordered case, the time complexity will be somewhere between O(n) and O(n^2), depending on the order of the elements of the array
The time required to sort a random array of n elements is O(n^2), in the worst case.

Hence, the performance of insertion sort varies based on the initial ordering of the elements in the array. This means that the time required to sort the array grows quadratically with the size of the input.

Evidence to support that conclusion:

Below are the run values for values of N in each case using Insertion sort ,

N	Random	Ordered	Reverse Ordered	Partially Ordered
250	380765	24372	756997	119567
500	532544	13092	414641	186926
1000	901381	13050	1795917	946829
2000	3436569	37604	6595801	738746
4000	13224606	194317	26255279	2772493
8000	53436437	57441	109268153	45383446

```
private static long getClock() {
    // FIXME by replacing the following code
    return System.nanoTime();
     //return 0;
    // END
}
```

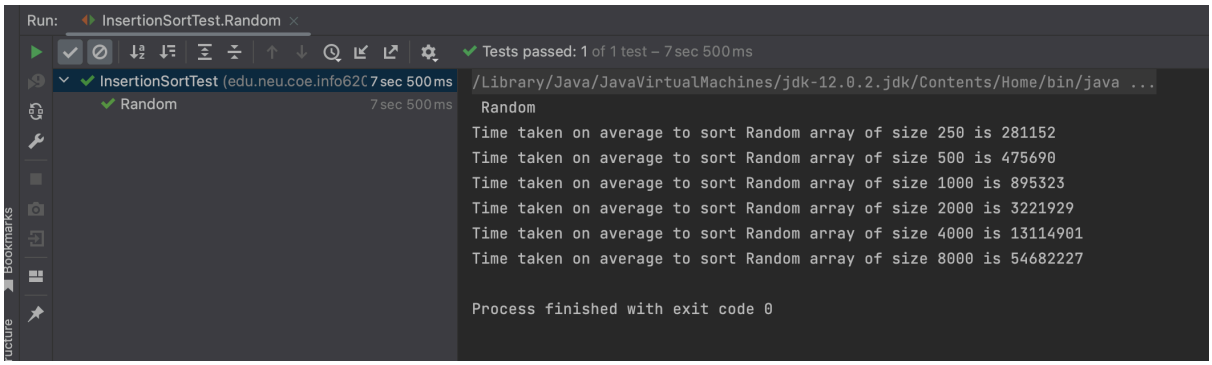
```
public double millisecs() {
    if (running) throw new TimerException();
    return toMillisecs(ticks);
}
```

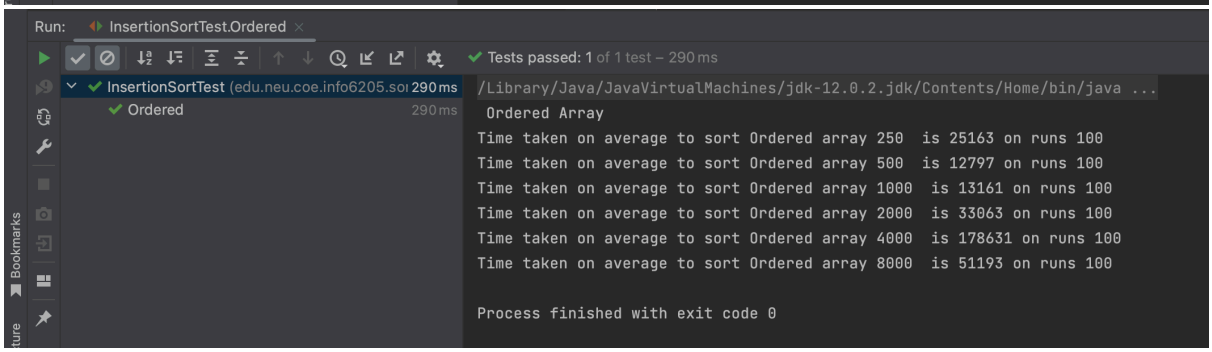
```
xiaohuanlin *
public <T, U> double repeat(int n, Supplier<T> supplier, Function<T, U> function, UnaryOperator<T> preFunction, Consumer<U> postFunction) {
    logger.trace("repeat: with " + n + " runs");
    pause();
    // FIXME: note that the timer is running when this method is called and should still be running when it returns. by replacing the following code
    double tTime = 0;
    for (int i = 1; i <= n; i++) {
        T inputValue = supplier.get();
        if (preFunction != null) {
            inputValue = preFunction.apply(inputValue);
        }
        resume();
        long startTime = System.nanoTime();
        U result = function.apply(inputValue);
        long endTime = System.nanoTime();
        tTime += (endTime - startTime) / 1_000_000.0;
        pauseAndLap();
        if (postFunction != null) {
            postFunction.accept(result);
        }
    }
    resume();
    return tTime / n;
     //return 0;
    // END
}
```

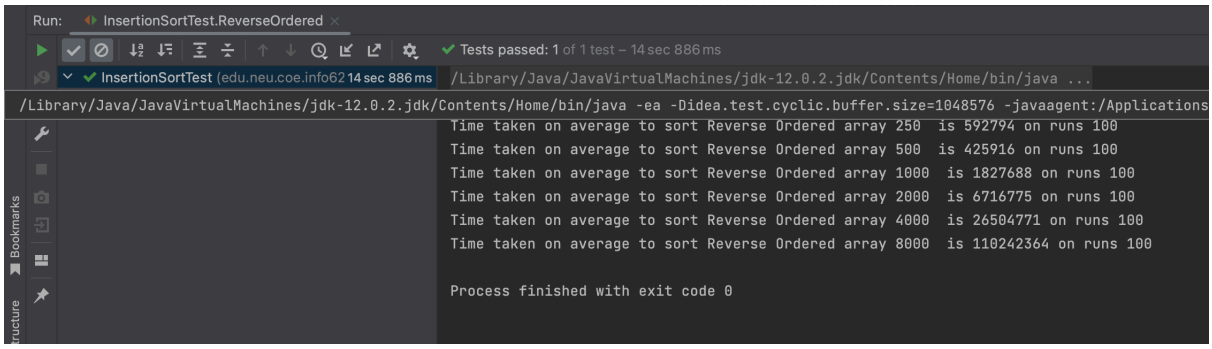
InsertionSort

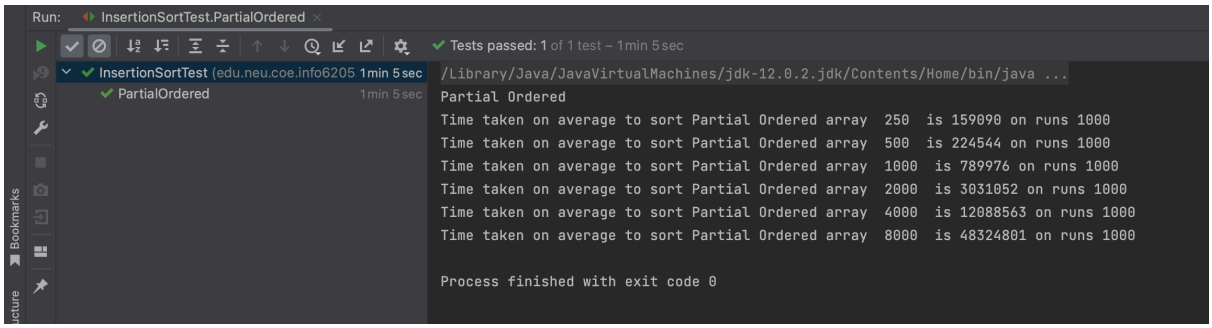
```
     */
    1 override  xiaohuanlin *
    public void sort(X[] xs, int from, int to) {
        final Helper<X> helper = getHelper();

        for (int i = from + 1; i < to; i++) {
            X x = xs[i];
            int j = i;
            while (j > from && helper.compare(xs[j - 1], x) > 0) {
                //xs[j] = xs[j - 1];
                helper.swap(xs, i: j - 1, j);
                j--;
            }
            xs[j] = x;
        }
    }
```

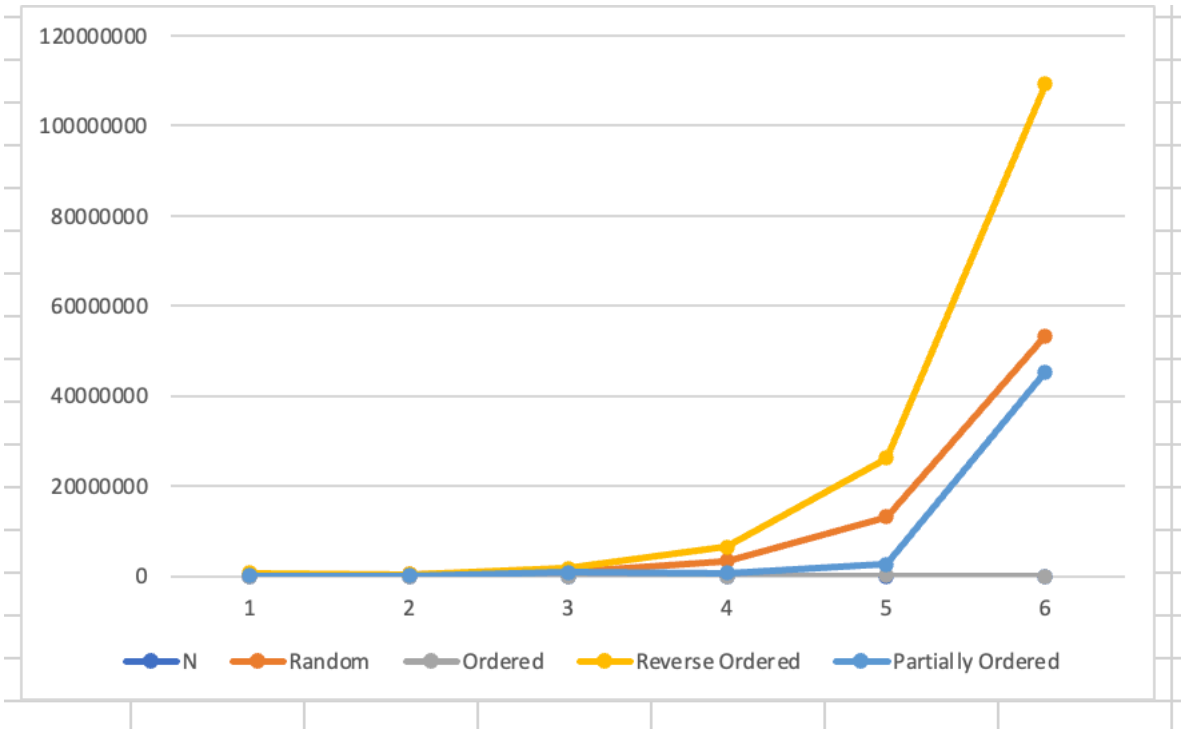

Run: InsertionSortTest.Random
Tests passed: 1 of 1 test – 7 sec 500 ms
InsertionSortTest (edu.neu.coe.info620 7 sec 500 ms
Random 7 sec 500 ms
/Library/Java/JavaVirtualMachines/jdk-12.0.2.jdk/Contents/Home/bin/java ...
Random
Time taken on average to sort Random array of size 250 is 281152
Time taken on average to sort Random array of size 500 is 475690
Time taken on average to sort Random array of size 1000 is 895323
Time taken on average to sort Random array of size 2000 is 3221929
Time taken on average to sort Random array of size 4000 is 13114901
Time taken on average to sort Random array of size 8000 is 54682227
Process finished with exit code 0


Run: InsertionSortTest.Ordered
Tests passed: 1 of 1 test – 290 ms
InsertionSortTest (edu.neu.coe.info6205.sor 290 ms
Ordered 290 ms
/Library/Java/JavaVirtualMachines/jdk-12.0.2.jdk/Contents/Home/bin/java ...
Ordered Array
Time taken on average to sort Ordered array 250 is 25163 on runs 100
Time taken on average to sort Ordered array 500 is 12797 on runs 100
Time taken on average to sort Ordered array 1000 is 13161 on runs 100
Time taken on average to sort Ordered array 2000 is 33063 on runs 100
Time taken on average to sort Ordered array 4000 is 178631 on runs 100
Time taken on average to sort Ordered array 8000 is 51193 on runs 100
Process finished with exit code 0


Run: InsertionSortTest.ReverseOrdered
Tests passed: 1 of 1 test – 14 sec 886 ms
InsertionSortTest (edu.neu.coe.info62 14 sec 886 ms
/Library/Java/JavaVirtualMachines/jdk-12.0.2.jdk/Contents/Home/bin/java ...
/Library/Java/JavaVirtualMachines/jdk-12.0.2.jdk/Contents/Home/bin/java -ea -Didea.test.cyclic.buffer.size=1048576 -javaagent:/Applications
Time taken on average to sort Reverse Ordered array 250 is 592794 on runs 100
Time taken on average to sort Reverse Ordered array 500 is 425916 on runs 100
Time taken on average to sort Reverse Ordered array 1000 is 1827688 on runs 100
Time taken on average to sort Reverse Ordered array 2000 is 6716775 on runs 100
Time taken on average to sort Reverse Ordered array 4000 is 26504771 on runs 100
Time taken on average to sort Reverse Ordered array 8000 is 110242364 on runs 100
Process finished with exit code 0


Run: InsertionSortTest.PartialOrdered
Tests passed: 1 of 1 test – 1 min 5 sec
InsertionSortTest (edu.neu.coe.info6205 1 min 5 sec
PartialOrdered 1 min 5 sec
/Library/Java/JavaVirtualMachines/jdk-12.0.2.jdk/Contents/Home/bin/java ...
Partial Ordered
Time taken on average to sort Partial Ordered array 250 is 159090 on runs 1000
Time taken on average to sort Partial Ordered array 500 is 224544 on runs 1000
Time taken on average to sort Partial Ordered array 1000 is 789976 on runs 1000
Time taken on average to sort Partial Ordered array 2000 is 3031052 on runs 1000
Time taken on average to sort Partial Ordered array 4000 is 12088563 on runs 1000
Time taken on average to sort Partial Ordered array 8000 is 48324801 on runs 1000
Process finished with exit code 0

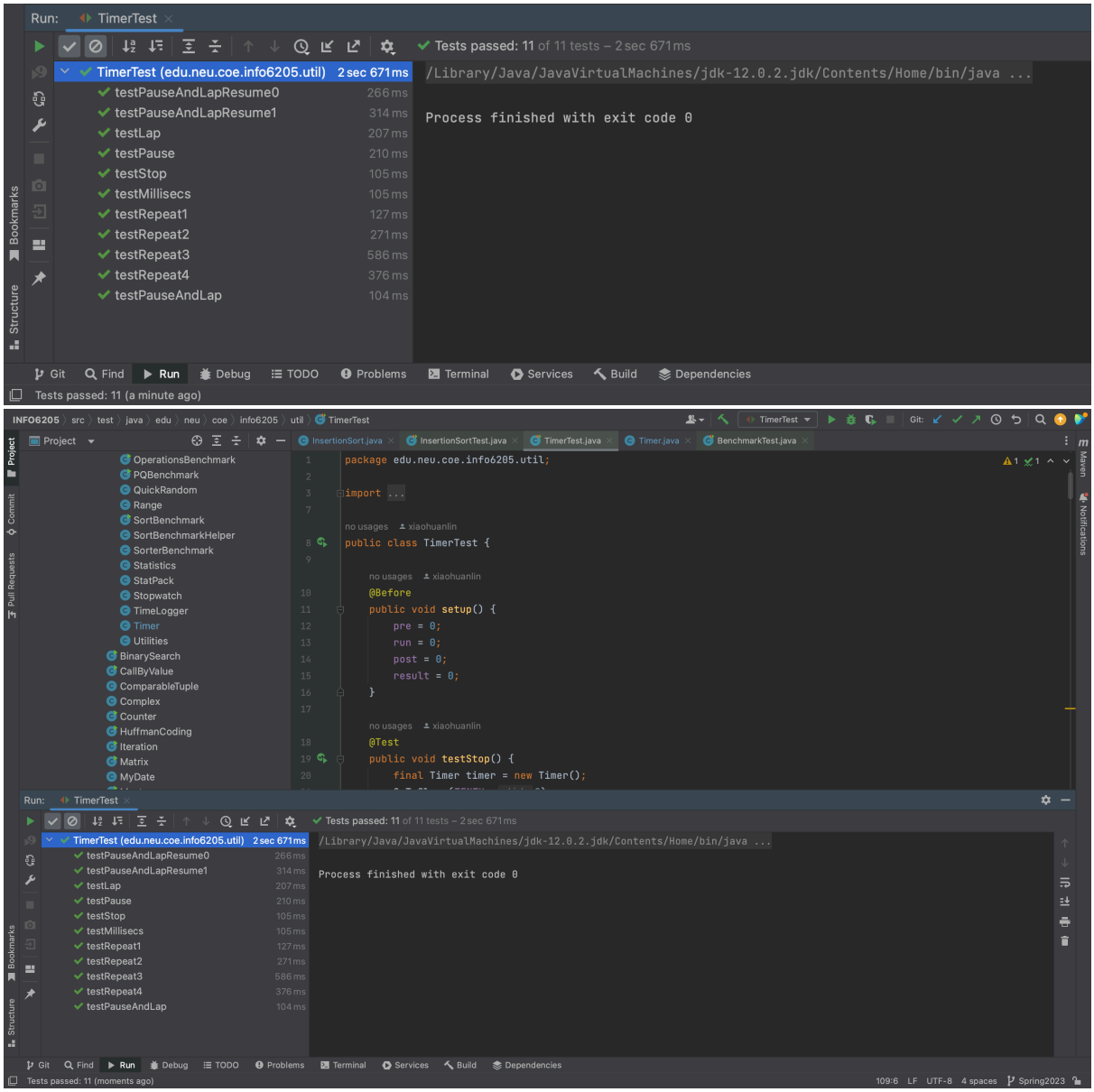
Graphical Representation:



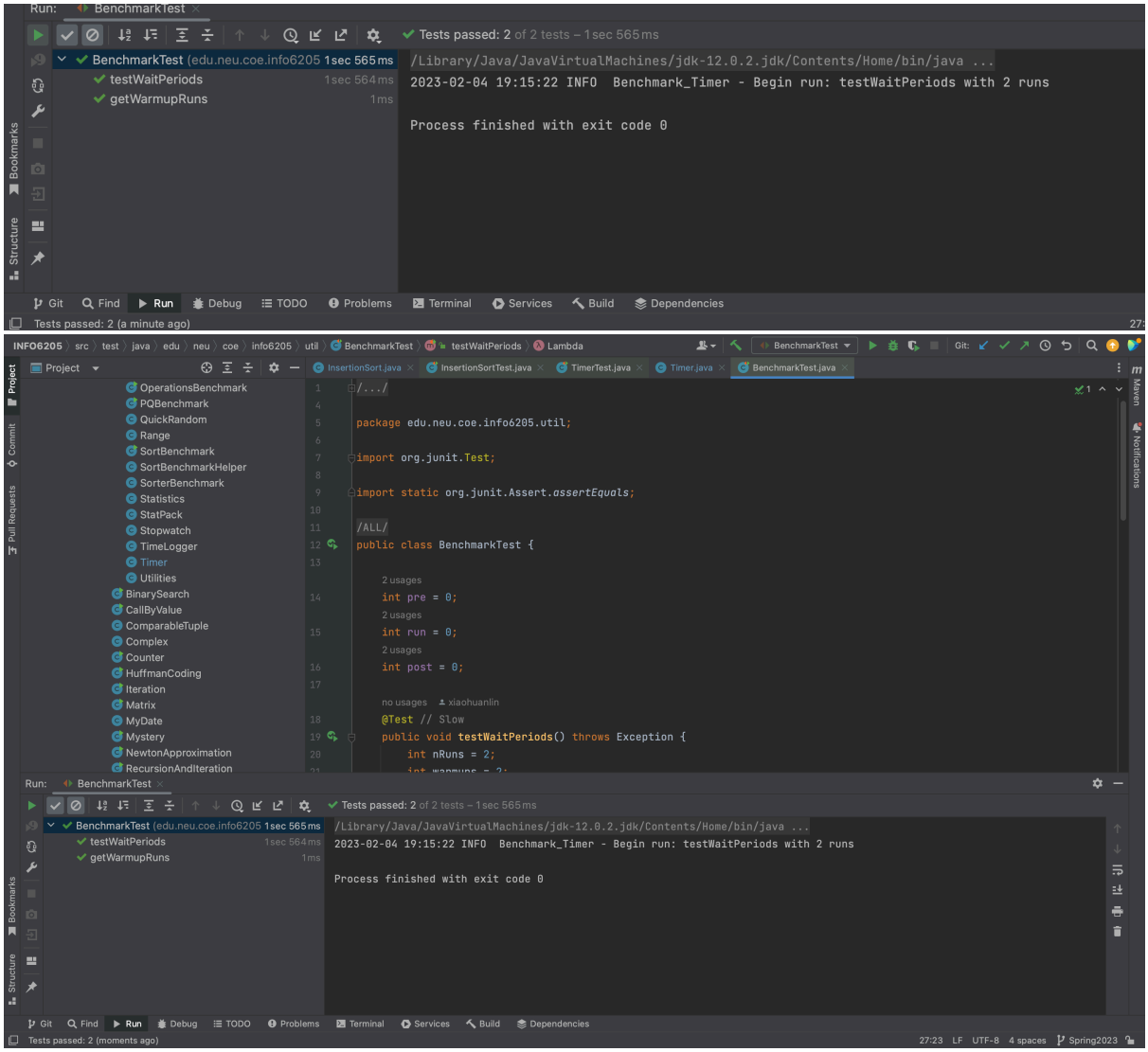
N	Random	Ordered	Reverse Ordered	Partially Ordered
250	380765	24372	756997	119567
500	532544	13092	414641	186926
1000	901381	13050	1795917	946829
2000	3436569	37604	6595801	738746
4000	13224606	194317	26255279	2772493
8000	53436437	57441	109268153	45383446

Unit Test Screenshots:

1.TimerTest



2.BenchmarkTest



3.InsertionSortTest