Program Structures and Algorithms
Spring 2023(SEC 03)

NAME: Vipul Rajderkar
NUID: 002700991

Task:

Part-1: Implement (*repeat*, *getClock*, and *toMillisecs*) methods in the Timer class

Part-2: Implement InsertionSort in InsertionSort class

Part-3: Implement the main method that can calculate execution time for random, ordered, partial ordered, and reverse-ordered arrays.

Relationship Conclusion:

After benchmarking insertion sort on 4 different types of arrays following relationship was concluded basis execution time:

<u>Ordered < Partially Ordered < Randomly Ordered < Reverse Ordered</u>

(*For a smaller number of inputs, the algorithm performs better with a partially sorted array than with a randomly sorted array. Please refer to the graphical representation in a later section)

Evidence to support that conclusion:

<u>Part 1: Timer class methods</u>

Code Snippet:

```
Vipul Rajderkar *
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");
    // FIXME: note that the timer is running when this method is called and should still be running when it returns. by replacing the follow
    //Added for ass3
    if(running){
        pause();
    }
    for(int i=0; i<n; i++){
        T x= supplier.get();
        if(preFunction != null){
            x= preFunction.apply(x);
        }
        resume();
        U y= function.apply(x);
        pauseAndLap();
        if(postFunction!=null){
            postFunction.accept(y);
        }
    }
    return meanLapTime();
}
```

```
2 usages  Vipul Rajderkar *
private static long getClock() {
    // Added for ass3
    return System.nanoTime();
    // END
}
```

```
2 usages  Vipul Rajderkar *
private static double toMillisecs(long ticks) {
    // Added for ass3
    return (ticks / 1000000);
    // END
}
```

Part 2: Implement Insertion Sort

Code Snippet:

```java
/**
 * Sort the sub-array xs:from:to using insertion sort.
 *
 * @param xs   sort the array xs from "from" to "to".
 * @param from the index of the first element to sort
 * @param to   the index of the first element not to sort
 */
1 override  Vipul Rajderkar *
public void sort(X[] xs, int from, int to) {
    final Helper<X> helper = getHelper();

    for (int i =  1; i < to; i++) {
        int j = i;
        while (j > 0 && helper.less(xs[j], xs[j - 1])) {
            helper.swap(xs, j - 1, j--);
        }
    }

}
```

Part 3: Main method to get execution time with different inputs

```java
    new *
    public static void main(String args[]) {

        Random random = new Random();
        for (int k = 1; k <= 4; k++) {
            int n = 1000;
            for (int i = 0; i < 5; i++) {
                Integer[] arr = new Integer[n];
                for (int j = 0; j < n; j++) {
                    arr[j] = random.nextInt();
                }

                if(k==2){
                    Arrays.sort(arr);
                    System.out.print("Mean time for sorted array of size " + n + " is ");
                }
                else if(k==3){
                    Arrays.sort(arr, fromIndex: n/2, n);
                    System.out.print("Mean time for partially sorted array of size " + n + " is " );
                }
                else if(k==4){
                    Arrays.sort(arr, Collections.reverseOrder());
                    System.out.print("Mean time for a reverse ordered array of size " + n + " is " );
                }
                else{
                    System.out.print("Mean time for a randomly sorted array of size " + n + " is " );
                }
                int temp = n;
                Timer timer = new Timer();
                final double time = timer.repeat( n: 50, () -> 5, t -> {
                    new InsertionSort().sort(arr, from: 0, temp);
                    return null;
                });

                System.out.println(time);

                n = n * 2;
            }

        }
    }
```

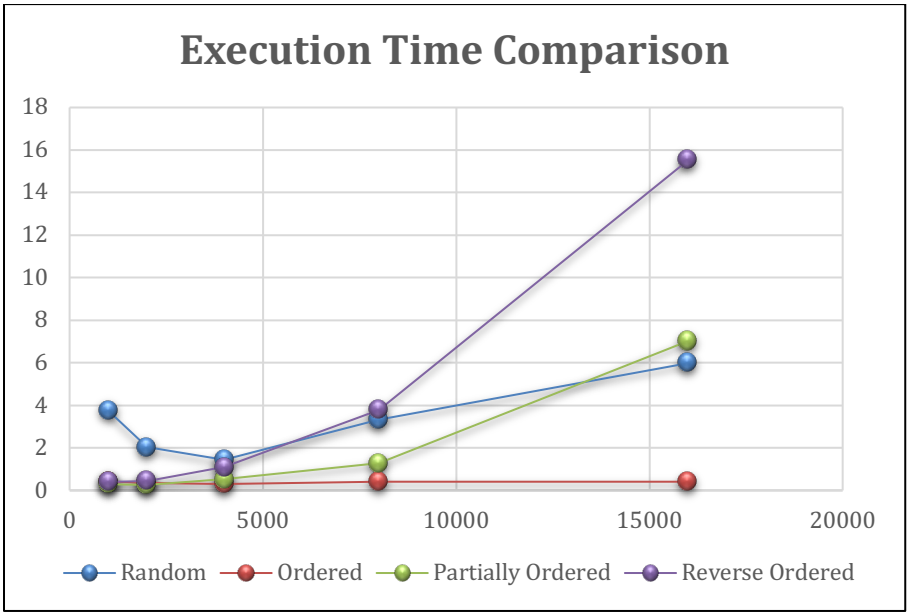
```
/Users/vipulrajderkar/Library/Java/JavaVirtualMachines/openjdk-19
Mean time for a randomly sorted array of size 1000 is 5.46
Mean time for a randomly sorted array of size 2000 is 2.96
Mean time for a randomly sorted array of size 4000 is 1.68
Mean time for a randomly sorted array of size 8000 is 4.72
Mean time for a randomly sorted array of size 16000 is 7.56
Mean time for sorted array of size 1000 is 0.4
Mean time for sorted array of size 2000 is 0.34
Mean time for sorted array of size 4000 is 0.34
Mean time for sorted array of size 8000 is 0.56
Mean time for sorted array of size 16000 is 0.5
Mean time for partially sorted array of size 1000 is 0.6
Mean time for partially sorted array of size 2000 is 0.34
Mean time for partially sorted array of size 4000 is 0.8
Mean time for partially sorted array of size 8000 is 2.92
Mean time for partially sorted array of size 16000 is 6.08
Mean time for a reverse ordered array of size 1000 is 0.32
Mean time for a reverse ordered array of size 2000 is 0.42
Mean time for a reverse ordered array of size 4000 is 0.94
Mean time for a reverse ordered array of size 8000 is 2.9
Mean time for a reverse ordered array of size 16000 is 21.74

Process finished with exit code 0
```

Graphical Representation:

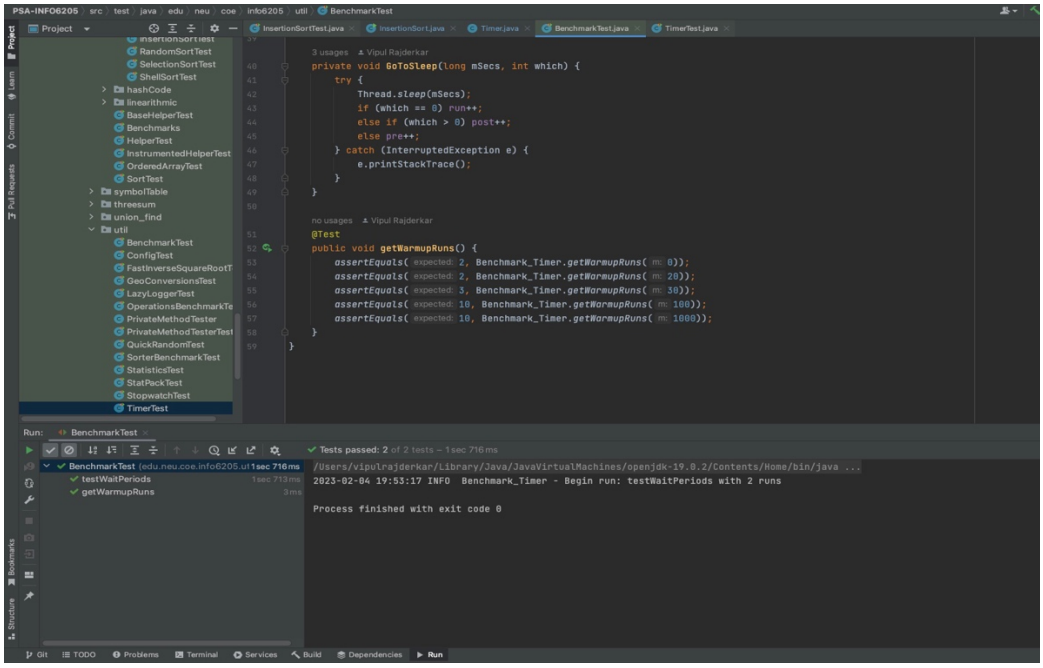
Timing Table:

N	Random	Ordered	Partially Ordered	Reverse Ordered
1000	3.76	0.42	0.3	0.4
2000	2.02	0.36	0.26	0.44
4000	1.44	0.3	0.52	1.1
8000	3.32	0.4	1.28	3.78
16000	5.98	0.4	7.02	15.52



Unit Test Screenshots:

BechmarkTest:



TimerTest:

```
PSA-INFO6205 > src > test > java > edu > neu > coe > info6205 > util > TimerTest > testRepeat2

        final double time = timer.millisecs();
        assertEquals(TENTH_DOUBLE, time, delta: 10.0);
        assertEquals( expected: 1, run);
    }

    no usages  Vipul Rajderkar
    @Test
    public void testRepeat1() {
        final Timer timer = new Timer();
        final double mean = timer.repeat( n: 10, () -> {
            GoToSleep(HUNDREDTH, which: 0);
            return null;
        });
        assertEquals( expected: 10, new PrivateMethodTester(timer).invokePrivate( name: "getLaps"));
        assertEquals( expected: TENTH_DOUBLE / 10, mean, delta: 6);
        assertEquals( expected: 10, run);
        assertEquals( expected: 0, pre);
        assertEquals( expected: 0, post);
    }

    no usages  Vipul Rajderkar
    @Test
    public void testRepeat2() {
        final Timer timer = new Timer();
        final int zzz = 20;
        final double mean = timer.repeat( n: 10, () -> zzz, t -> {
            GoToSleep(t, which: 0);

Run: TimerTest
Tests passed: 11 of 11 tests – 2 sec 836 ms
TimerTest (edu.neu.coe.info6205.util) 2 sec 836 ms
  testPauseAndLapResume0 422 ms
  testPauseAndLapResume1 318 ms
  testLap 210 ms
  testPause 212 ms
  testStop 107 ms
  testMillisecs 105 ms
  testRepeat1 126 ms
  testRepeat2 253 ms
  testRepeat3 606 ms
  testRepeat4 366 ms
  testPauseAndLap 111 ms
/Users/vipulrajderkar/Library/Java/JavaVirtualMachines/openjdk-19.0.2/Contents/Home/bin/java ...

Process finished with exit code 0
```

InsertionSortTest:

```
PSA-INFO6205 > src > test > java > edu > neu > coe > info6205 > sort > elementary > InsertionSortTest > testStaticInsertionSort

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

import ...

/ALL/
public class InsertionSortTest {

    no usages  Vipul Rajderkar
    @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( instrumenting: "true", seed: "0", inversions: "1", cutoff: "", interimInversions: "");
        Helper<Integer> helper = HelperFactory.create( description: "InsertionSort", list.size(), config);
        helper.init(list.size());
        final PrivateMethodTester privateMethodTester = new PrivateMethodTester(helper);
        final StatPack statPack = (StatPack) privateMethodTester.invokePrivate( name: "getStatPack");
        SortWithHelper<Integer> sorter = new InsertionSort<>(helper);
        sorter.preProcess(xs);
        Integer[] ys = sorter.sort(xs);

Run: InsertionSortTest
Tests passed: 6 of 6 tests – 492 ms
InsertionSortTest (edu.neu.coe.info6205.sort 492 ms
  testMutatingInsertionSort 391 ms
  sort0 37 ms
  sort1 22 ms
  sort2 27 ms
  sort3 10 ms
  testStaticInsertionSort 5 ms
/Users/vipulrajderkar/Library/Java/JavaVirtualMachines/openjdk-19.0.2/Contents/Home/bin/java ...
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:
StatPack {hits: 19,800, normalized=42.995; copies: 0, normalized=0.000; inversions: 4,950, normalized=10.749; swaps: 4,950, normalized=10.749; fixe

Process finished with exit code 0
```