

## PROGRAM STRUCTURES AND ALGORITHMS

FALL 2021

Assignment - 2

Shashwat Shrey -- 002128122

Tasks Performed :

- 1.Implemented the 3 functions in timer.java
- 2.Implemented Insertion Sort
- 3.Plotted a graphical relation for growth in random ordered , partially ordered , ordered and reverse ordered arrays.

1.Timer.java

First 3 images are the executed functions, last image is the unit test

```
private static double toMillisecs(long ticks) {  
    // TO BE IMPLEMENTED  
    return ticks / Math.pow(10,6);  
}
```

```
*/  
private static long getClock() {  
    // TO BE IMPLEMENTED  
    return System.nanoTime();  
}
```

```

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");
    // TO BE IMPLEMENTED: note that the timer is running when this method is called and should still be running when it returns
    pause();
    T input = supplier.get();
    for(int i = 0; i < n; i++){
        if(preFunction != null){
            preFunction.apply(input);
        }
        resume();
        U output = function.apply(input);
        pauseAndLap();
        if(postFunction != null){
            postFunction.accept(output);
        }
    }

    //meanLapTime();

    return meanLapTime();
}

```

The screenshot shows an IDE with the following components:

- Project Explorer:** A tree view on the left showing the project structure. The 'util' package is expanded, showing files like BenchmarkTest, ConfigTest, FastInverseSquareR, LazyLoggerTest, PrivateMethodTeste, and TimerTest.
- Editor:** The main window displays the `TimerTest` class. It includes a package declaration, imports, and two methods: `@Before` `setup()` and `@Test` `testStop()`. The `testStop()` method contains logic to create a `Timer` object, sleep, stop it, and assert the time.
- Run Console:** At the bottom, it shows the execution of the tests. It reports 'Tests passed: 10 of 10 tests - 2 sec 335 ms'. Below this, a list of individual test results is shown, including `testPauseAndLapResume0`, `testPauseAndLapResume1`, `testLap`, `testPause`, `testStop`, `testMillisecs`, `testRepeat1`, `testRepeat2`, `testRepeat3`, and `testPauseAndLap`, each with its execution time in milliseconds.
- Terminal:** The bottom right pane shows the message 'Process finished with exit code 0'.

## 2.InsertionSort(First image is the code and second image is of the unit tests)

```
public void sort(X[] xs, int from, int to) {  
    final Helper<X> helper = getHelper();  
    int start = from;  
    int items = from + to;  
    for(int i = start; i < items; i++){  
        for(int j = i; j > from && helper.compare(xs, i: j - 1, j) > 0; j-- ){  
            helper.swap(xs, i: j - 1, j);  
        }  
        start++;  
    }  
};  
}
```

The screenshot shows an IDE with the following components:

- Project Explorer:** A tree view on the left showing the project structure. The 'elementary' package is expanded, showing files like BubbleSortTest, InsertionSortMST, InsertionSortOpt, InsertionSortTest, SelectionSortTest, and ShellSortTest.
- Editor:** The main window displays the source code for `InsertionSortTest.java`. The code includes package declarations, imports, and a `@Test` annotated `sort0()` method. The method uses `ArrayList` to create a list of integers and `ConfigTest` to configure the test environment.
- Run Console:** The bottom panel shows the output of the test execution. It indicates that 6 out of 6 tests passed in 287ms. The output includes debug messages for configuration settings and performance statistics for the `InsertionSort` algorithm.

**Test Results:**

Test Name	Duration (ms)	Status
testMutatingInsertionSort	236	Passed
sort0	24	Passed
sort1	2	Passed
sort2	14	Passed
sort3	7	Passed
testStaticInsertionSort	4	Passed

**Performance Statistics:**

```
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}
```

### 3. Analysis of random , ordered, partially ordered and reverse ordered array.

=====Reverse=====

```
2021-09-25 16:54:04 INFO Benchmark_Timer - Begin run: Reverse -> 100 with 100 runs
4.896E-4
2021-09-25 16:54:04 INFO Benchmark_Timer - Begin run: Reverse -> 200 with 100 runs
8.4871E-4
2021-09-25 16:54:04 INFO Benchmark_Timer - Begin run: Reverse -> 400 with 100 runs
0.00312921
2021-09-25 16:54:04 INFO Benchmark_Timer - Begin run: Reverse -> 800 with 100 runs
0.00274078
2021-09-25 16:54:04 INFO Benchmark_Timer - Begin run: Reverse -> 1600 with 100 runs
0.00540955
```

=====Random=====

```
2021-09-25 16:52:23 INFO Benchmark_Timer - Begin run: Random -> 10000 with 100 runs
0.0941350000000001
2021-09-25 16:52:23 INFO Benchmark_Timer - Begin run: Random -> 20000 with 100 runs
0.07234498
2021-09-25 16:52:25 INFO Benchmark_Timer - Begin run: Random -> 40000 with 100 runs
0.14900833
2021-09-25 16:52:34 INFO Benchmark_Timer - Begin run: Random -> 80000 with 100 runs
0.2878596200000004
2021-09-25 16:54:04 INFO Benchmark_Timer - Begin run: Random -> 160000 with 100 runs
0.5908650200000001
```

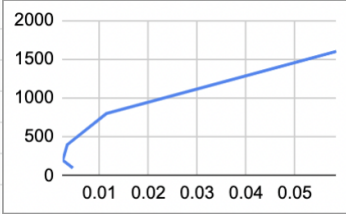
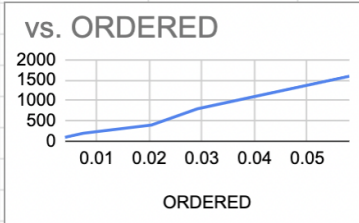
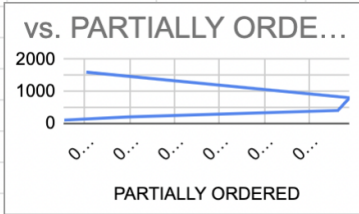
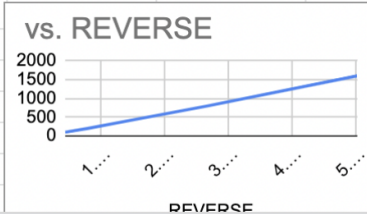
=====Ordered=====

```
2021-09-25 16:54:04 INFO Benchmark_Timer - Begin run: Ordered -> 100 with 100 runs
5.89010000000001E-4
2021-09-25 16:54:04 INFO Benchmark_Timer - Begin run: Ordered -> 200 with 100 runs
9.174799999999999E-4
2021-09-25 16:54:04 INFO Benchmark_Timer - Begin run: Ordered -> 400 with 100 runs
0.001514610000000002
2021-09-25 16:54:04 INFO Benchmark_Timer - Begin run: Ordered -> 800 with 100 runs
0.00276623
2021-09-25 16:54:04 INFO Benchmark_Timer - Begin run: Ordered -> 1600 with 100 runs
0.00529169
```

=====Partially ordered=====

```
2021-09-25 16:54:04 INFO Benchmark_Timer - Begin run: PartiallyOrdered ->100 with 100 runs
4.4336E-4
2021-09-25 16:54:04 INFO Benchmark_Timer - Begin run: PartiallyOrdered ->200 with 100 runs
8.32070000000001E-4
2021-09-25 16:54:04 INFO Benchmark_Timer - Begin run: PartiallyOrdered ->400 with 100 runs
0.002812089999999997
2021-09-25 16:54:04 INFO Benchmark_Timer - Begin run: PartiallyOrdered ->800 with 100 runs
0.00273625
2021-09-25 16:54:04 INFO Benchmark_Timer - Begin run: PartiallyOrdered ->1600 with 100 runs
0.00549628
```

GRAPH(Mathematical Relation) , Growth --  $O(N^2)$

RANDOM	N	T	
	100	0.00454086	
	200	0.00245126	
	400	0.00340501	
	800	0.01140583	
	1600	0.05852463	
ORDERED	N	T	
	100	0.0039479	
	200	0.00747792	
	400	0.02043042	
	800	0.02912919	
	1600	0.05802542	
PARTIALLY ORDERED	N	T	
	100	0.00384331	
	200	0.00743536	
	400	0.01910835	
	800	0.01976669	
	1600	0.00508419	
REVERSE	N	T	
	100	4.34E-04	
	200	7.91E-04	
	400	0.00141625	
	800	0.00267488	
	1600	0.00501586	