ASSIGNMENT 3 - PROGRAM STRUCTURE AND ALGORITHM
SATYAM JAGTAP

PART 1 - IMPLEMENTING BENCHMARK

QUESTION

Implement three (3) methods (*repeat*, *getClock*, and *toMillisecs*) of a class called *Timer*. Please see the skeleton class that I created in the repository. *Timer* is invoked from a class called *Benchmark_Timer* which implements the *Benchmark* interface.

OBSERVATION

The repeat method iterates over the number of repetitions and calls the function with the supplier, then laps in the iteration and then pauses the repetition if the preFunction or postFunction is null. PreFunction and postFunction are paused and resumed if the preFunction or postFunction is not null.
The get clock method gives the system clock's tick count as a result.
The toMillisecs method converts a nanosecond value for the ticks parameter into a millisecond value.

OUTPUT

```
Output
LeetCodePractice (run) ×   Run (InsertionSortBenchmark) ×   Test (TimerTest) ×

----------------------< edu.neu.coe.mgen:INFO6205 >-----------------------
Building INFO6205 1
--------------------------------[ jar ]---------------------------------

--- maven-surefire-plugin:2.12.4:test (default-cli) @ INFO6205 ---
Surefire report directory: /Users/satyamjagtap/Documents/PSA/INFO6205/target/surefire-reports

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running edu.neu.coe.info6205.util.TimerTest
Tests run: 11, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 2.517 sec

Results :

Tests run: 11, Failures: 0, Errors: 0, Skipped: 0

------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  2.987 s
Finished at: 2023-02-04T23:00:30-05:00
------------------------------------------------------------------------
```

PART 2 - IMPLEMENTING INSERTION SORT

QUESTION

Implement *InsertionSort* (in the *InsertionSort* class) by simply looking up the insertion code used by *Arrays.sort.* If you have the *instrument* = *true* setting in *test/resources/config.ini*, then you will need to use the *helper* methods for comparing and swapping (so that they correctly count the number of swaps/compares).

OBSERVATION

The insertionSort function compares each item in turn to create the final sorted array (or list). The array xs is sorted from "from" to "to". It offers the hit count, the normalized value, the element inversions, and element swaps.

OUTPUT

```
Output
LeetCodePractice (run)   Run (InsertionSortBenchmark)   Test (InsertionSortTest)

-----------------------< edu.neu.coe.mgen:INFO6205 >------------------------
Building INFO6205 1
--------------------------------[ jar ]---------------------------------

--- maven-surefire-plugin:2.12.4:test (default-cli) @ INFO6205 ---
Surefire report directory: /Users/satyamjagtap/Documents/PSA/INFO6205/target/surefire-reports

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running edu.neu.coe.info6205.sort.elementary.InsertionSortTest
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: 2,421, normalized=5.257;
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, normalized=10.7
Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.099 sec

Results :

Tests run: 6, Failures: 0, Errors: 0, Skipped: 0

------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
Total time:  0.559 s
Finished at: 2023-02-04T23:03:49-05:00
------------------------------------------------------------------------
```

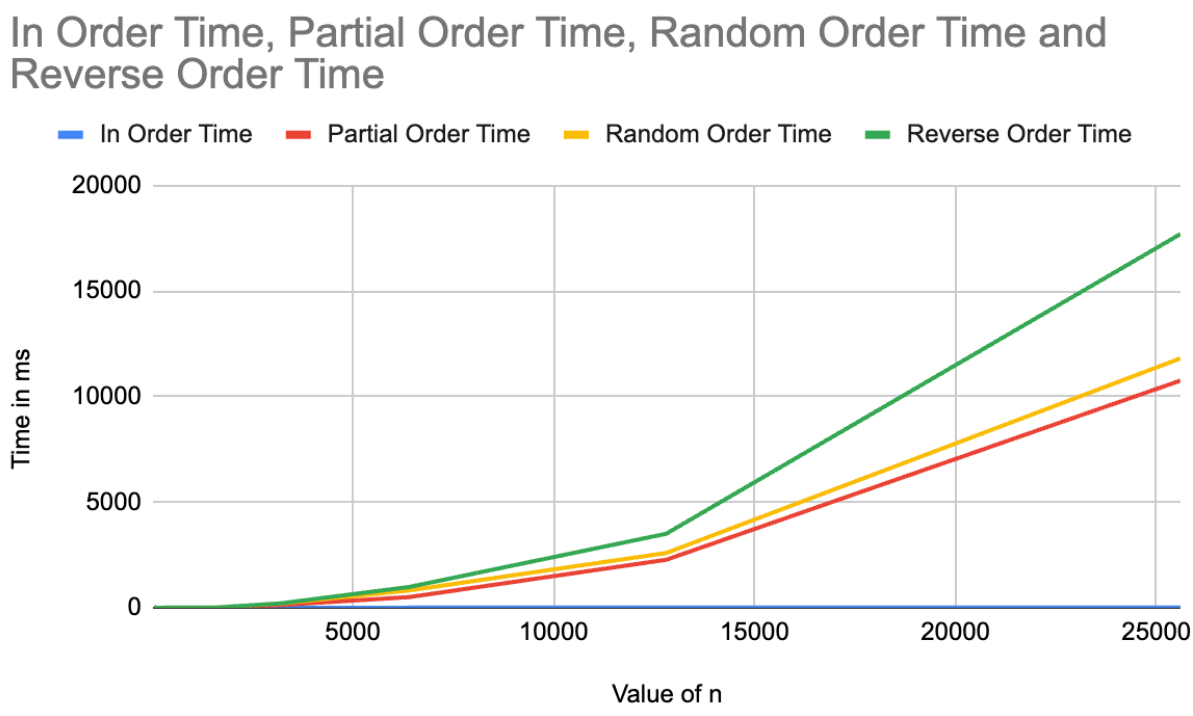
PART 3 - IMPLEMENTING THE MAIN PROGRAM TO RUN BENCHMARK

QUESTION

Implement a main program (or you could do it via our own unit tests) to actually run the following benchmarks: measure the running times of this sort, using four different initial array ordering situations: random, ordered, partially-ordered and reverse-ordered

OUTPUT

N	In Order Time	Partial Order Time	Random Order Time	Reverse Order Time
50	1.67	0.96	2.57	4.2
100	1.59	0.64	1.12	0.85
200	0.48	0.63	0.7	0.94
400	1.4	1.29	1.98	1.53
800	1.33	1.98	2.97	4.24
1600	1.03	6.94	8.72	8.74
3200	1.23	125.95	194.25	204.23
6400	1.29	501.17	817.22	980.1
12800	1.55	2268.73	2589.72	3500.78
25600	1.88	10756.21	11809.34	17699.32



```
Run: BenchmarkInsertion
/Library/Java/JavaVirtualMachines/jdk-18.0.2.1.jdk/Contents/Home/bin/java ...
-------START--------
No of element, N: 150
2023-02-04 22:13:18 INFO  Benchmark_Timer - Begin run: Reverse Order Array with 100 runs
Reverse Order Array : mean sort time (in ms): 2.12
2023-02-04 22:13:19 INFO  Benchmark_Timer - Begin run: Random Order Array with 100 runs
Random Order Array : mean sort time (in ms): 1.38
2023-02-04 22:13:19 INFO  Benchmark_Timer - Begin run: Partial Order Array with 100 runs
Partial Order Array : mean sort time (in ms): 0.65
2023-02-04 22:13:19 INFO  Benchmark_Timer - Begin run: In-Order Array with 100 runs
In-Order Array : mean sort time (in ms): 0.34
----END-----
-------START--------
No of element, N: 250
2023-02-04 22:13:19 INFO  Benchmark_Timer - Begin run: Reverse Order Array with 100 runs
Reverse Order Array : mean sort time (in ms): 1.17
2023-02-04 22:13:19 INFO  Benchmark_Timer - Begin run: Random Order Array with 100 runs
Random Order Array : mean sort time (in ms): 1.44
2023-02-04 22:13:19 INFO  Benchmark_Timer - Begin run: Partial Order Array with 100 runs
Partial Order Array : mean sort time (in ms): 0.92
2023-02-04 22:13:19 INFO  Benchmark_Timer - Begin run: In-Order Array with 100 runs
In-Order Array : mean sort time (in ms): 0.45
----END-----
-------START--------
No of element, N: 450
2023-02-04 22:13:19 INFO  Benchmark_Timer - Begin run: Reverse Order Array with 100 runs
Reverse Order Array : mean sort time (in ms): 1.7
2023-02-04 22:13:20 INFO  Benchmark_Timer - Begin run: Random Order Array with 100 runs
Random Order Array : mean sort time (in ms): 1.51
```

```
Run: BenchmarkInsertion
-------START--------
No of element, N: 450
2023-02-04 22:13:19 INFO  Benchmark_Timer - Begin run: Reverse Order Array with 100 runs
Reverse Order Array : mean sort time (in ms): 1.7
2023-02-04 22:13:20 INFO  Benchmark_Timer - Begin run: Random Order Array with 100 runs
Random Order Array : mean sort time (in ms): 1.51
2023-02-04 22:13:20 INFO  Benchmark_Timer - Begin run: Partial Order Array with 100 runs
Partial Order Array : mean sort time (in ms): 1.61
2023-02-04 22:13:20 INFO  Benchmark_Timer - Begin run: In-Order Array with 100 runs
In-Order Array : mean sort time (in ms): 0.4
----END-----
-------START--------
No of element, N: 850
2023-02-04 22:13:20 INFO  Benchmark_Timer - Begin run: Reverse Order Array with 100 runs
Reverse Order Array : mean sort time (in ms): 4.48
2023-02-04 22:13:20 INFO  Benchmark_Timer - Begin run: Random Order Array with 100 runs
Random Order Array : mean sort time (in ms): 2.76
2023-02-04 22:13:21 INFO  Benchmark_Timer - Begin run: Partial Order Array with 100 runs
Partial Order Array : mean sort time (in ms): 1.88
2023-02-04 22:13:21 INFO  Benchmark_Timer - Begin run: In-Order Array with 100 runs
In-Order Array : mean sort time (in ms): 0.28
----END-----
-------START--------
No of element, N: 1200
2023-02-04 22:13:21 INFO  Benchmark_Timer - Begin run: Reverse Order Array with 100 runs
Reverse Order Array : mean sort time (in ms): 4.93
2023-02-04 22:13:21 INFO  Benchmark_Timer - Begin run: Random Order Array with 100 runs
Random Order Array : mean sort time (in ms): 4.95
```

```
Run: BenchmarkInsertion
2023-02-04 22:13:22 INFO  Benchmark_Timer - Begin run: In-Order Array with 100 runs
In-Order Array : mean sort time (in ms): 0.22
----END-----
-------START--------
No of element, N: 2200
2023-02-04 22:13:22 INFO  Benchmark_Timer - Begin run: Reverse Order Array with 100 runs
Reverse Order Array : mean sort time (in ms): 16.33
2023-02-04 22:13:24 INFO  Benchmark_Timer - Begin run: Random Order Array with 100 runs
Random Order Array : mean sort time (in ms): 16.06
2023-02-04 22:13:26 INFO  Benchmark_Timer - Begin run: Partial Order Array with 100 runs
Partial Order Array : mean sort time (in ms): 12.43
2023-02-04 22:13:27 INFO  Benchmark_Timer - Begin run: In-Order Array with 100 runs
In-Order Array : mean sort time (in ms): 0.22
----END-----
-------START--------
No of element, N: 3400
2023-02-04 22:13:27 INFO  Benchmark_Timer - Begin run: Reverse Order Array with 100 runs
Reverse Order Array : mean sort time (in ms): 42.17
2023-02-04 22:13:32 INFO  Benchmark_Timer - Begin run: Random Order Array with 100 runs
Random Order Array : mean sort time (in ms): 40.05
2023-02-04 22:13:36 INFO  Benchmark_Timer - Begin run: Partial Order Array with 100 runs
Partial Order Array : mean sort time (in ms): 49.14
2023-02-04 22:13:41 INFO  Benchmark_Timer - Begin run: In-Order Array with 100 runs
In-Order Array : mean sort time (in ms): 0.24
----END-----
-------START--------
No of element, N: 8800
2023-02-04 22:13:41 INFO  Benchmark_Timer - Begin run: Reverse Order Array with 100 runs
Reverse Order Array : mean sort time (in ms): 353.55
Version Control  Run  TODO  Problems  Terminal  Services  Build  Dependencies
Build completed successfully in 4 sec, 220 ms (a minute ago)
23:13  LF  UTF-8  4 spaces
```