Name: KEERTHANA SATHEEESH

NUID: 002747795

Task:

1. You are to implement three (3) methods (repeat, getClock, and toMillisecs) of a class called Timer.
2. Implement InsertionSort(in the InsertionSort class) by simply looking up the insertion code used byArrays.sort.
3. Implement a main program (or you could do it via your 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.

Code:

Insertion Code:

Text

Description automatically generated

Text

Description automatically generated

Timer Class Code:

Text

Description automatically generated with medium confidence

Text

Description automatically generated

Text

Description automatically generated

Unit Test Case:

1. Timer Test Case

Graphical user interface, text

Description automatically generated

1. Insertion Test Case

Graphical user interface, text, application

Description automatically generated

Evidence:

Partially ordered array time increase based on the size of n - 0(n^2).

Reverse ordered array time increase based on the size of n - 0(n^2).

Ordered array time increase based on the size of n - 0(n).

Random ordered array time increase based on the size of n - 0(n^2).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| n | Ordered Array | Random Array | Reverse Array | Partially Ordered Array |
| 400 | 0.53 | 0.26 | 0.28 | 0.15 |
| 800 | 0.92 | 0.48 | 1.11 | 0.38 |
| 1600 | 2.06 | 1.85 | 3.49 | 1.36 |
| 3200 | 4.08 | 7.03 | 14.3 | 5.39 |
| 6400 | 9.29 | 28.57 | 56 | 21.45 |
| 12800 | 19.49 | 113.1 | 228.96 | 88.23 |

Conclusion:

According to the graph given above we can see that the Reverse array takes the largest sorting time. Next is random array followed by partially ordered array and finally ordered array.

Ordered array takes the least time.