Lê Thanh Phương Nam

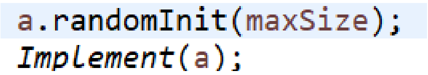
ITITWE19025

Lab 02
Simple Sorting Methods (Buble Sort, Selection Sort, Insertion Sort)
Revised by Tran Thanh Tung

a. Write a Java program to measure time (in seconds) needed for each simple sorting algorithms applying on ***the same random array*** of integer values. Sizes of arrays are accordingly 10000, 15000, 20000, 25000, 30000, 35000, 40000, 45000 and 50000. Each time, you write down the measured time in following table.

Table 1 - Experiment 1: Simple sorting on random data

	Bubble Sort (ms)	**Selection Sort (ms)**	**Insertion Sort (ms)**
10000	191	54	87
15000	435	113	203
20000	828	202	368
25000	1386	321	665
30000	2105	458	961
35000	2945	624	1334
40000	3944	946	1970
45000	5005	1081	2546
50000	6338	1360	2815



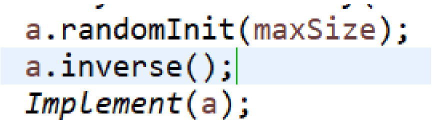
```
a.randomInit(maxSize);
Implement(a);
```

b. Write some code to measure time (in seconds) needed for each simple sorting algorithms applying on ***Inversely sorted*** and ***Already-sorted order*** integer arrays of **10000** elements.

Table 2 - Experiment 2: Simple sorting in special cases

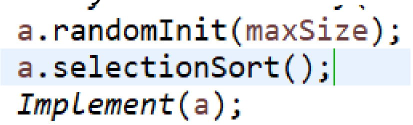
	Bubble Sort (ms)	**Selection Sort (ms)**	**Insertion Sort (ms)**
Inverse order(10000)	189	56	84
Already order(10000)	48	41	1

Inverse order:



```
a.randomInit(maxSize);
a.inverse();
Implement(a);
```

Already order:



```
a.randomInit(maxSize);
a.selectionSort();
Implement(a);
```

c. Based on above table, give your comments on real complexity of the three simple sorting algorithms. (Remember, all of them are O(n^2) in theory).

Average time complexity for this algorithm is O(n log(n)). For many collections, that is an excellent average sorting time. It also has the benefit of being completely in place, so no additional storage is required. In the worst-case scenario, however, the time complexity is O. (n2).