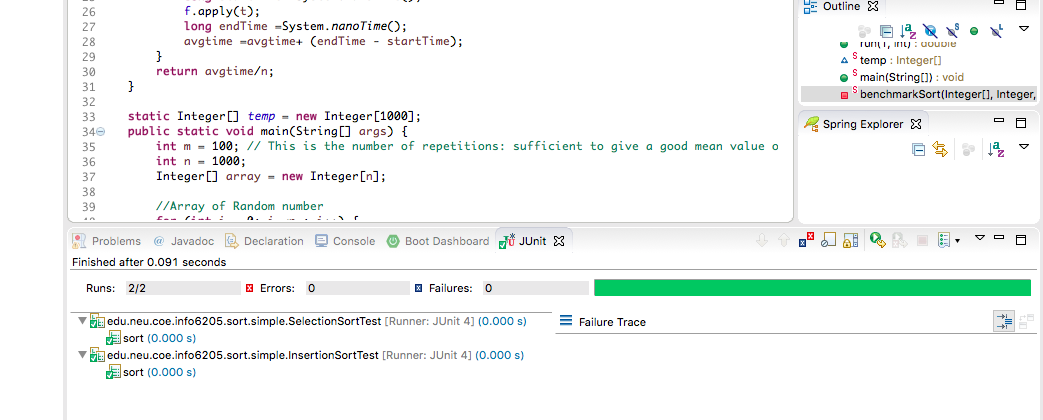
**Assignment3**

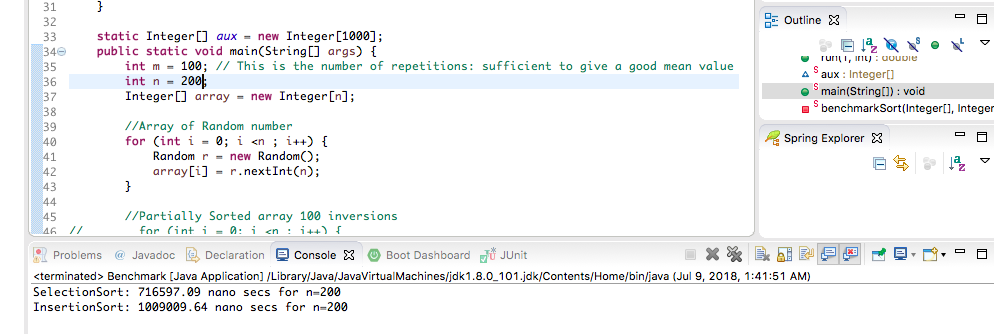
**Problem statement:**

Implement benchmark for sorting algorithm – selection sort and insertion sort. Check time complexity of the given sorting algorithm if array is random, partially sorted, sorted and reverse sorted.

Unite test cases pass for – selection sort and insertion sort:



**Random Array**



Data:

For m =100

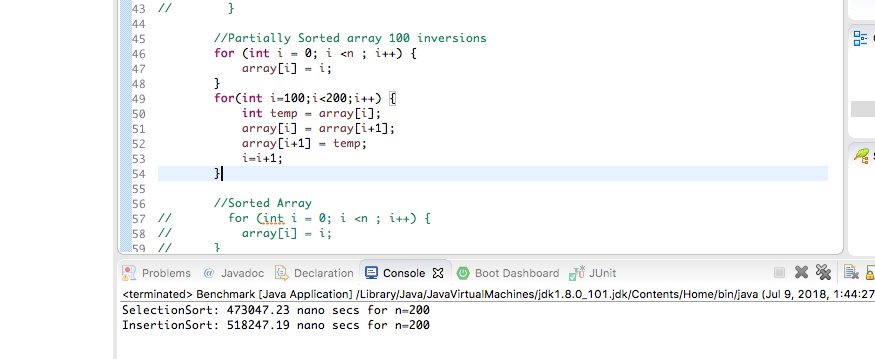
|  |  |  |
| --- | --- | --- |
| N | SelectionSort (nanosec) | InsertionSort (nanosec) |
| 200 | 715145.63 | 604212.36 |
| 400 | 875209.99 | 1055784.8 |
| 600 | 1382346.75 | 1599351.53 |
| 800 | 1413411.4 | 2101401.94 |
| 1000 | 2517972.54 | 3280938.77 |

Graph:

conclusion:

1. For less value of N random array, insertion sort algorithm is faster than selection sort algorithm.
2. For large value of N random array, selection sort algorithm become faster than insertion sort algorithm.

**Partially sorted Array**



Data:

For m =100

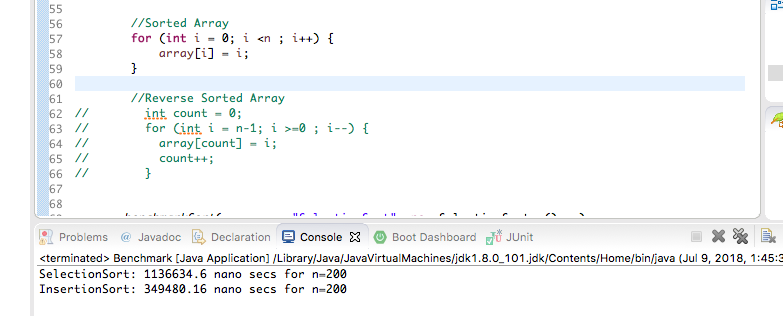
|  |  |  |
| --- | --- | --- |
| N | SelectionSort (nanosec) | InsertionSort (nanosec) |
| 200 | 480324.5 | 522838.03 |
| 400 | 1182696.85 | 949378.94 |
| 600 | 1397604.17 | 1313881.29 |
| 800 | 1695804.64 | 1868205.72 |
| 1000 | 2042147.52 | 2542002.51 |

Graph:

Conclusion:

1. For partially sorted array, insertion sort is preferable than selection sort as the number of element N increases.
2. For partially-sorted arrays, insertion sort runs in linear time. (Number of exchanges equals the number of inversions)

**Sorted array**



Data:

For m =100

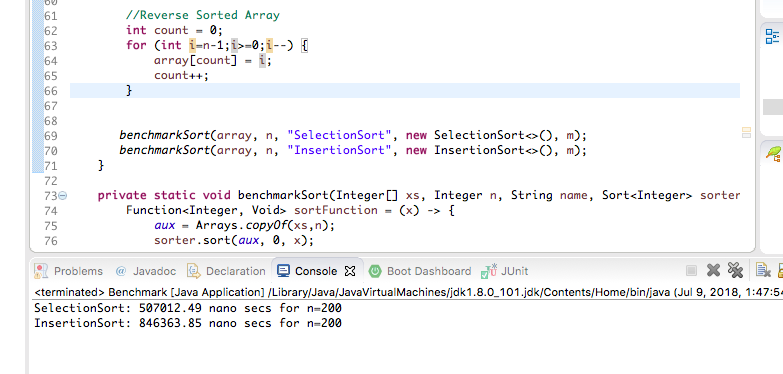
|  |  |  |
| --- | --- | --- |
| N | SelectionSort (nanosec) | InsertionSort (nanosec) |
| 200 | 723859.68 | 250297.35 |
| 400 | 805417.43 | 284289.07 |
| 600 | 989641.4 | 485412.93 |
| 800 | 1464055.85 | 813892.13 |
| 1000 | 1630867.42 | 1104330.09 |

Graph:

Conclusion:

1. For sorted array, Insertion sort is faster than selection sort since insertion sort makes N–1 compares and 0 exchanges
2. For selection sort, running time is insensitive to input. It takes quadratic time, even if the input is sorted.

**Reverse sorted array**



Data:

For m =100

|  |  |  |
| --- | --- | --- |
|  | SelectionSort (nanosec) | InsertionSort (nanosec) |
| 200 | 1028908.36 | 719215.1 |
| 400 | 1320279.43 | 999823.72 |
| 600 | 1238423.85 | 1663080.8 |
| 800 | 1615473.02 | 2899557.39 |
| 1000 | 2195628.55 | 4059888.41 |

Graph:

Conclusion:

1. For reverse sorted array, as the number of elements increases selection sort takes less time than insertion sort since number of swap for insertion sort increases with number of elements. It makes ~ ½ N 2 compares and ~ ½ N 2 exchanges.