~\OneDrive - VietNam National University - HCM INTERNATIONAL UNIVERSITY\Desktop\DSA\DSA LAB NEW\Lab 2 Simple sorting\ITITSB22029_DoMinhDuy_Lab2\Problem 2\SelectSortApp.java

```
1  class ArraySel {
2     private long[] a; // reference to array a
3     private int nElems; // number of data items
4     private int nComparisons; // number of comparisons
5     private int nSwaps; // number of swaps
6
7     // --------------------------------------------------------------
8     public ArraySel(int max) {
9        a = new long[max]; // create the array
10       nElems = 0; // no items yet
11       nComparisons = 0; // initialize comparisons to 0
12       nSwaps = 0; // initialize swaps to 0
13    }
14
15    // --------------------------------------------------------------
16    public void insert(long value) { // put element into array
17       a[nElems] = value; // insert it
18       nElems++; // increment size
19    }
20
21    // --------------------------------------------------------------
22    public void display() { // displays array contents
23       for (int j = 0; j < nElems; j++) // for each element,
24          System.out.print(a[j] + " "); // display it
25       System.out.println("");
26    }
27
28    // --------------------------------------------------------------
29    public void selectionSort() {
30       int out, in, min;
31
32       for (out = 0; out < nElems - 1; out++) { // outer loop
33          min = out; // minimum element starts as out
34          for (in = out + 1; in < nElems; in++) { // inner loop
35             nComparisons++; // increment comparisons counter
36             if (a[in] < a[min]) { // if min is greater,
37                min = in; // we have a new min
38             }
39          }
40          // Display the array after each inner loop
41          System.out.println("After inner loop " + out + ": ");
42          display();
43
44          if (min != out) { // swap only if necessary
45             swap(out, min); // swap them
46             System.out.println("Swapping " + a[min] + " and " + a[out]);
47          } else {
```

```
48              System.out.println("No swap needed for index " + out);
49           }
50           System.out.println("Number of comparisons so far: " + nComparisons);
51           System.out.println("Number of swaps so far: " + nSwaps);
52        }
53        System.out.println("Total comparisons made: " + nComparisons);
54     }
55
56     // ------------------------------------------------------------
57     private void swap(int one, int two) {
58        long temp = a[one];
59        a[one] = a[two];
60        a[two] = temp;
61        nSwaps++; // increment swap counter
62     }
63
64     public int getComparisonNumber() {
65        return nComparisons;
66     }
67
68     public int getSwapNumber() {
69        return nSwaps;
70     }
71
72     // ------------------------------------------------------------
73  } // end class ArraySel
74
75  ////////////////////////////////////////////////////////////////
76
77  class SelectSortApp {
78     public static void main(String[] args) {
79        int maxSize = 100; // array size
80        ArraySel arr; // reference to array
81        arr = new ArraySel(maxSize); // create the array
82
83        arr.insert(77); // insert 10 items
84        arr.insert(99);
85        arr.insert(44);
86        arr.insert(55);
87        arr.insert(22);
88        arr.insert(88);
89        arr.insert(11);
90        arr.insert(00);
91        arr.insert(66);
92        arr.insert(33);
93
94        arr.display(); // display items
95
96        arr.selectionSort(); // selection-sort them
97
```

```
98        arr.display(); // display them again
99
100       // display the number of swaps and comparisons
101       System.out.println("The number of swaps = " + arr.getSwapNumber());
102       System.out.println("The total number of comparisons = " + arr.getComparisonNumber());
103
104       // The algorithm complexity estimation
105       int n = arr.getComparisonNumber();
106       System.out.println("Estimated complexity (n*(n-1)/2): " + n * (n - 1) / 2);
107       System.out.println("The algorithm has O(n^2) complexity.");
108    }
109 } // end class SelectSortApp
110
```