PLAYBOOK NOTES: BUBBLE SORT

This sort will compare an element of an array to the element next to it. If the first element is larger, the two elements will be swapped.

This process will continue until a loop go through the array with no changes made.

The algorithm is patterned after Program 13.46 in the textbook:

("Data Structures, Algorithms & Software Principles in C", Thomas A. Standish, © 1995)

```
ALGORITHM:

Integer notDone

Loop
{
   Set the variable notDone to false

   For loop: index → 0…length-1
   {
      Check to see if array[index] is greater than array[index + 1)
      If (array[index] > array[index + 1])
         then swap elements and set notDone to true

} while notDone
```

EXAMPLE:

Given the array of 4 integers below:

Index 0	Index 1	Index 2	Index 3
80	16	8	13

There are two loops for this sort. The first loop will keep cycling while the value of notDone is true.

The second loop is a for-loop that will go from 0 to 2
(Even though the array has for elements, the index goes from 0 to 3, so length will be 3 and 3-1 equals 2)

NOTE: Observe in the algorithm above, that before hitting the for-loop, notDone is set to true.

FIRST TIME HITTING THE FOR-LOOP:

First comparison:

Index 0	Index 1	Index 2	Index 3
80	16	8	13

80 is greater than 16, so we will swap them and set nonDone to true, which means we will hit the for-loop at least one more time.

Index 0	Index 1	Index 2	Index 3
16	80	8	13

Second comparison:

Index 0	Index 1	Index 2	Index 3
16	80	8	13

80 is greater than 8, so we will swap them.

Index 0	Index 1	Index 2	Index 3
16	8	80	13

Third Comparison:

Index 0	Index 1	Index 2	Index 3
16	8	80	13

80 is greater than 13, so swap them.

Index 0	Index 1	Index 2	Index 3
16	8	13	80

Because we swapped at least one value, we will go through the loop a second time.

SECOND TIME HITTING THE FOR-LOOP:

First Comparison:

Index 0	Index 1	Index 2	Index 3
16	8	13	80

Since 16 is greater than 8, we will swap them. This also means that we are guaranteed at least one more time through the loop.

Index 0	Index 1	Index 2	Index 3
8	16	13	80

Second Comparison:

Index 0	Index 1	Index 2	Index 3
8	16	13	80

16 is greater than 13, so we will swap them

Index 0	Index 1	Index 2	Index 3
8	13	16	80

Third comparison:

Index 0	Index 1	Index 2	Index 3
8	13	16	80

16 is NOT greater than 80, the values are not swapped.

Index 0	Index 1	Index 2	Index 3
8	13	16	80

Because we swapped at least one value, we will go through the loop a third time.

THIRD TIME HITTING THE FOR-LOOP:

First Comparison:

Index 0	Index 1	Index 2	Index 3
8	13	16	80

Since 8 is NOT greater than 13, the numbers are not swapped.

Index 0	Index 1	Index 2	Index 3
8	13	16	80

Second Comparison:

Index 0	Index 1	Index 2	Index 3
8	13	16	80

Since 13 is NOT greater than 16, the numbers are not swapped.

Index 0	Index 1	Index 2	Index 3
8	13	16	80

Third Comparison:

Index 0	Index 1	Index 2	Index 3
8	13	16	80

Since 16 is NOT greater than 80, the numbers are not swapped.

Index 0	Index 1	Index 2	Index 3
8	13	16	80

Since no numbers were swapped this time through the for-loop, the notDone is NOT set to true and outside loop is now done looping.

Our array is now in order:

Analysis:

The average run time for the bubble sort is $O(n^2)$