Assignment: Bubble Sort

Review the bubble sort algorithm lecture, and/or find online resources to study the algorithm until you understand the algorithm well.

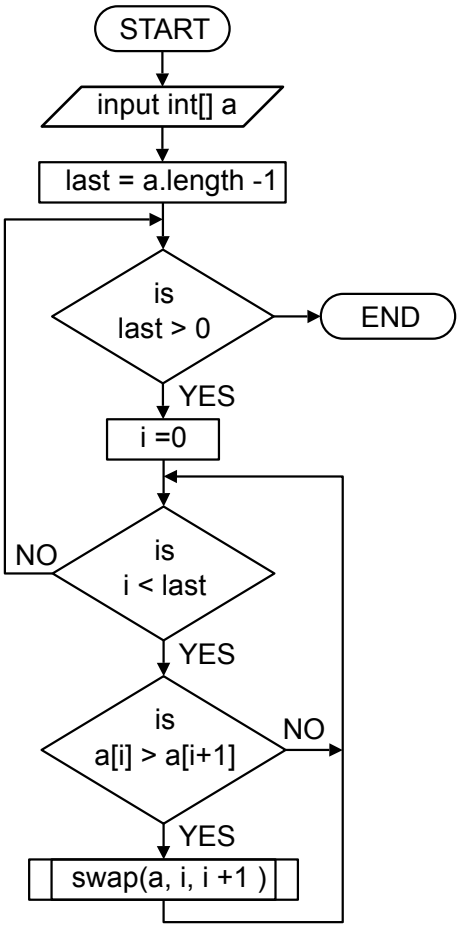
Your assignment is to create a Java class named `Sort` that contains a method `bubbleSort` that implements the bubble sort algorithm on an array of integers as described in more detail below.

In the algorithm, the large values “bubble” up to the end of the list. (Other versions the algorithm can be implemented so that small values bubble down to the front of the list.) This “bubbling” is, reality, the swapping of adjacent array elements.

Before writing the full bubble sort algorithm, first write a method that can be called called `swap` that will swap the position of two elements of the array. This method is to have no return value, and take three parameters: the array containing the elements to swap, and two indexes of the elements to swap. Write some simple test code to test your method.

Once that is complete, implement the bubble sort algorithm using the algorithm represented by the flowchart to the right. There is no need for your `bubbleSort` method to return any value or preserve the original array.

Test code for `bubbleSort` and the expected results are given below. Note that the actual elements of the arrays are generated randomly, so will vary with each run, but it should be easy to tell if the code is sorting the arrays properly.



Results of test code for `bubbleSort` **:**

```
Array 1, unsorted:
4 10 8 1 5 7 3 2 9 6
Array 1, sorted:
1 2 3 4 5 6 7 8 9 10

Array 2, unsorted:
66 0 16 37 5 68 62 72 99 45 68 84 45 79 22 2 93 35 23 94
100 6 54 12 15 32 8 19 83 75 80 91 83 44 93 96 79 10 78 63
58 35 26 91 0 81 48 39 88 47 12 79 42 71 30 40 31 41 42 52
20 38 85 76 47 16 16 23 73 82 99 26 62 66 26 99 40 30 76 1
31 69 54 18 72 45 14 17 52 32 34 88 24 68 3 100 14 35 84 56
Array 2, sorted:
0 0 1 2 3 5 6 8 10 12 12 14 14 15 16 16 16 17 18 19
20 22 23 23 24 26 26 26 30 30 31 31 32 32 34 35 35 35 37 38
39 40 40 41 42 42 44 45 45 45 47 47 48 52 52 54 54 56 58 62
62 63 66 66 68 68 68 69 71 72 72 73 75 76 76 78 79 79 79 80
81 82 83 83 84 84 85 88 88 91 91 93 93 94 96 99 99 99 100 100

Array 3, unsorted:
1 4 3 5 4
Array 3, sorted:
1 3 4 4 5
```

Java test code for `bubbleSort`:

```
public class UseBubbleSort {

     public static void main(String[] args) {

          int[] a1 = { 4, 10, 8, 1, 5, 7, 3, 2, 9, 6 };
          int[] a2 = newIntArray(100,0,+100);
          int[] a3 = newIntArray(5,0,5);

          System.out.println("Array 1, unsorted:");
          printIntArray(a1);
          Sort.bubbleSort(a1);
          System.out.println("Array 1, sorted:");
          printIntArray(a1);

          System.out.println();

          System.out.println("Array 2, unsorted:");
          printIntArray(a2);
          Sort.bubbleSort(a2);
          System.out.println("Array 2, sorted:");
          printIntArray(a2);

          System.out.println();

          System.out.println("Array 3, unsorted:");
          printIntArray(a3);
          Sort.bubbleSort(a3);
          System.out.println("Array 3, sorted:");
          printIntArray(a3);

     }


     /*** Continued on the next page ... ***/
```

```
/** Print an array to the console, with a new line after 20
 *  elements are printed in the current row
 *
 * @param a the array to be printed
 */
public static void printIntArray(int[] a) {
     printIntArray(a, 20);
}


/** Print an array to the console, with a new line after each
 *  "rowLength" number of elements are printed
 *
 * @param a the array to be printed
 * @param rowLength the maximum number of elements printed per row
 */
public static void printIntArray(int[] a, int rowLength) {

     for (int i = 0; i < a.length; i++) {

          // Print the array element followed by a space character
          System.out.print(a[i] + " ");

          // if the row is complete, print a newline
          if( (i+1)%rowLength == 0 ) {
               System.out.println();
          }

     }

     // If the final row is incomplete, print a final newline
     if ( (a.length)%rowLength != 0) {
          System.out.println();
     }

}
```

/* Continued on the next page ... ***/**

```
	/** Generate an array with random integers in the range [min,max]
	 *
	 * @param length the number of elements in the array
	 * @param min the smallest value of any element
	 * @param max the largest value of any element
	 * @return the array of random integers in the range [min,max]
	 */
	public static int[] newIntArray(int length, int min, int max) {

		int[] a = new int[length];

		for(int i = 0; i<a.length; i++) {
			a[i] = randomInt(min, max);
		}

		return a;

	}


	/** Generate a random integer in the range [min,max]
	 *
	 * @param min the smallest possible value generated
	 * @param max the largest possible value generated
	 */
	public static int randomInt(int min, int max) {
		return (int)(Math.random() * (max-min+1) + min);
	}

}
```