**CSC 20: Lab 13 - SmartStringList**

This lab is designed to give you another opportunity to practice writing and testing a class.

Write a SmartStringList class with the following methods.

public SmartStringList(int capacity) constructs a list of given max capacity (use an array internally)

public void add(String s) adds given String to end of list

public String get(int i) returns the String at index i.

public int indexOf(String s) returns the position of the given value or -1 if not found.

public void sort() puts the list in sorted order, if not already.

Your class should create an array of size "capacity" and can then accommodate that many "add" method calls.

What makes the list "smart" is that it always knows whether its elements are in sorted order and avoids extra work when they are. Do the following to ensure this.

* Maintain a field sorted, which is true if the elements of the list are in sorted order. (Lists of length 0 *are* sorted so begin with this field true, set it to false if a newly added element is smaller than the old last element, and set it to true each time the sort method is called.)
* If sorted is true, the method indexOf should do a binary search and the method sort should do nothing.
* If sorted is false, the method indexOf should do a linear search and the method sort should do a selection sort.

You may assume that the list will not have any duplicates, so don't test for this case. Because these are strings, you will have to use equalsIgnoreCase and/or compareToIgnoreCase to make your comparisons (search "java 8 string" to see the javadoc on these methods if you need to).

Write a main method that tests your class thoroughly.

Use selection sort and binary search in your class, but do not copy them from anywhere, instead recreate them from the following pseudocode.

sort(arr)

n = length of arr

for i = 0 .. n-1

let min be the index of smallest element in arr[i..n-1]

swap elements at index i and min

indexOf(arr,val) // return index of where val is in arr, or -1 if not found

min = 0

max = (length of arr)-1

while max >= min

mid = middle of min..max range

if (val and arr[mid] are equal)

return mid

else

update min or max

Write a simple javadoc for the class and each of its public members.

When you are done, call me over to have a look.