Sorting Lexicographically

Lexicographic sorting is effectively how a phone book is sorted. For example, if we have a list of last names we could use this to determine phone book order.

First, let's recall the two sorts from grade 11. Both sorts are based on a comparison of items in an array. The code below refers to an array of integers called A. The problem we have, is to sort an array of strings rather than integer. Review the sorts and then proceed below.

Selection Sort

```
FOR J = 1 TO N-1
 FOR K  = (J+1) TO N
   IF A(K)>A(J) THEN
      SWAP A(J), A(K)
   END IF
 NEXT K
NEXT J
```

Bubble Sort

```
DO
  FLAG = 0
  FOR J =  1 TO N-1
    IF A(J) > A(J+1) THEN
       SWAP A(J), A(J+1)
       FLAG = 1
    END IF
 NEXT J
LOOP UNTIL FLAG = 0
```

To compare strings in Java, we cannot use the <,>, = or other operators as we can imagine. What does it mean to ask if one string is more than another? This is where lexicographic sorting is necessary if we wish to alpha sort a list of names.

The compareTo method is as follows and is found in the string class of objects.

compareTo
`public int compareTo(String anotherString)`

Compares two strings lexicographically. The comparison is based on the Unicode value of each character in the strings.

Parameters: anotherString - the String to be compared.
Returns: the value 0 if the argument string is equal to this string; a value less than 0 if this string is lexicographically less than the string argument; and a value greater than 0 if this string is lexicographically greater than the string argument.

Try it!:

```
String name1 = "Bruce, Scott";
String name2 = "Simpson, Bart";
if (name1.compareTo(name2) > 0)    //if name1 appears before name2
{
        System.out.println(name1 + name2);

}
else
        System.out.println(name2 + name1);
```

Change it!: Change name1 to "Simpson, Bart" and name2 to "Bruce, Scott" and run it again.