Practice Exercise #27: Sorted Linked List

http://www.comp.nus.edu.sg/~cs1020/4_misc/practice.html

Objective:

- Programming on linked list
- Learning about **compareTo()** method in **Comparable** interface

Task Statement

Java provides the **Comparable** and **Comparator** interfaces for comparing object (check out the API documentation). We will explore the former here.

The **Comparable** interface contains the **compareTo()** method that compares this object with the specified object in the parameter, and returns a negative integer, zero, or a positive integer, if this object is less than, equal to, or greater than the specified object, respectively.

(For example, the **String** class and it has a **compareTo()** method too. Assuming that s1 and s2 are String objects, s1.compareTo(s2) compares the two strings lexicographically, and returns a negative integer, zero, or a positive integer if s1 is lexicographically less than s2, equals to s2, or lexicographically greater than s2, respectively.)

In this exercise, you are to add elements into a sorted list containing strings as its elements. Each time you add a new element, you must place it in the correct position to maintain the sorted order of the list. You may assume that all elements added are distinct.

You are given a client program **TestSortedList.java** which you should not modify, and hence need not submit. This client program uses the **addOrdered()** method in **MySortedLinkedList** class which you need to complete.

The **System.out.println(list)** statement (which is commented out) in the **readNames()** method is for you to check whether the method works properly each time you add a new element into the sorted list.

The file **MySortedLinkedList.java** contains the definition of **ListNode** class as well as **MySortedLinkedList** class. You are to complete the **toString()** method and **addOrdered()** method in the **MySortedLinkedList** class. The **addOrdered()** method is to add an element into the list at the right place to keep the list in sorted order.

As **addOrdered()** method requires you to compare list elements, **MySortedLinkedList** class needs to support comparison between instances of the generic type E, via the **Comparable** interface. Hence:

```
class MySortedLinkedList <E extends Comparable <E>>
```

In the client program **TestSortedList**, we also need to state this:

```
public class TestSortedList implements Comparable <String>
```

and provide an implementation of the **compareTo()** method which would be used in **addOrdered()** to compare the elements of two list nodes. As the client program is creating a linked list of strings, the **compareTo()** method (of **Comparable** interface) is implemented here as the **compareTo()** method in the **String** class:

```
public int compareTo(String that) {
   return this.compareTo(that);
}
```

You are not to modify the rest of the given code in **MySortedLinkedList**. You need to submit only **MySortedLinkedList.java**.

Sample run:

Inputs are shown in blue.

```
5
Elaine Diana Avery Candy Bubble
List:
[Avery, Bubble, Candy, Diana, Elaine]
```