**CIS 181 - Lab 7**

**Array-Based Implementation of ADT SortedList**

## ****Objectives****

* To gain a better understanding of ADT by implementing the SortedList class using **array**.
* To gain a better understanding of Java exceptions.
* To become familiar with the Comparable interface.

## ****Description****

One of the most frequently performed computing tasks is the maintenance, in some specified order, of a collection of data. Such data are called **sorted** data, or a **sorted list**. The problem of maintaining sorted data requires more than simply sorting the data. Often you need to insert some new data item into its proper, sorted place. Similarly, you often need to delete some data item. The operations of the ADT **SortedList** is defined as follows:

**public boolean** isEmpty();

// Determines whether a sorted list is empty.

// Precondition: None.

// Postcondition: Returns true if the sorted list is empty, otherwise returns false.

**public int** size();

// Determines the length of a sorted list.

// Precondition: None.

// Postcondition: Returns the number of items that are currently in the sorted list.

**public void** sortedAdd(Comparable newItem) **throws** ListException;

// Adds an item to the sorted list.

// Precondition: None.

// Postcondition: The item is added to the sorted list in sorted order.

**public void** sortedRemove(Comparable anItem) **throws** ListException;

// Removes an item from the sorted list.

// Precondition: None.

// Postcondition: The item is removed from the sorted list and the sorted order maintained.

**public** Object get(**int** index) **throws** ListException;

// Retrieves a list item by position.

// Precondition: index is the position of the item to be retrieved.

// Postcondition: If 1 <= index <= size(), the item at position index in the sorted

// list is returned.

**public void** removeAll();

// Deletes all the items from the sorted list.

// Precondition: None.

// Postcondition: The sorted list is empty.

**public int** locateIndex(Comparable anItem);

// Finds an item in the sorted list.

// Precondition: None.

// Postcondition: If the item is in the sorted list, its index position in the sorted

// list is returned. If the item is not in the sorted list, the index of where it belongs

// in the sorted list is returned.

The ADT SortedList differs from the ADT List in that a sorted list inserts and deletes items by their values and not by their positions. For example, sortedAdd determines the proper position for item according to its value. Also, locateIndex - which determines the position of any item, given its value - is a sorted list operation but not a list operation.

The element of the sorted list have one more requirement: they must implement the interface [*java.lang.Comparable*](http://java.sun.com/j2se/1.5.0/docs/api/java/lang/Comparable.html) and have an implementation of the method compareTo, so that the sorted list can order the elements. For example, when we implement the method sortedAdd, if newItem.compareTo(item\_X) > 0, where item\_X is an item in the list, then newItem should be added after item\_X; otherwise, newItem should be added before item\_X. Note that, in this lab, the String class we use in the test class has already implemented the compareTo method, so you can compare two strings directly.

**Exercises**

1. Download the following code

* SortedListInterface.java
* SortedList.java
* ListException.java
* SortedListTester.java

1. In Eclipse, create a new Java project called "Lab 7", and import the above 4 files into a default package.
2. Compile and run the program. It should print out the following message:

ListException: get (index out of range): 99

The index of "dd" in the list is: 1

The index of "kk" in the list is: 1

1. Complete the following three methods in class *SortedList*:

* **public void** sortedAdd(Comparable newItem) **throws** ListException;
* **public void** sortedRemove(Comparable anItem) **throws** ListException;
* **public int** locateIndex(Comparable anItem) **throws** ListException;

1. Your program should print out exactly the following messages:

aa, bb, cc, dd, ee.

aa, bb, dd, ee, ff.

ListException: sortedRemove (item not in the list): cc

aa, bb, cat, dd, ee, ff.

ListException: get (index out of range): 99

aa, bb, cat, dd, ee, ff.

The index of "dd" in the list is: 4

The index of "kk" in the list is: 7

1. Submit your completed program (SortedList.java) to the myCourses!