CSCI 230 – Homework 1

List ADT Implementation

Part 1: Implement a List ADT

Objectives

- Become familiar with an abstract implementation at its simplest form.
- Apply object oriented design principles.
- Test thoroughly a program and understand the principles of unit testing.

Background information

This assignment involves implementing an Abstract Data Type (ADT) for an **ordered list**. A list is ordered if it stores data based on their *natural ordering*. Natural ordering may mean increasing order(ex., 3, 5, 5, 7, 10) or decreasing order (ex, 10, 7, 5, 5, 3) for numeric values, alphabetical order (“cat”, “dog”, “fish”) or lexicographic order (“2dog”, “ca”, “mr ape”) for strings, and so on. When new data are added or deleted, it is done in a way that the list remains in order.

A traversal of the list would visit the records in the natural order that makes sense for that list, even if the values are not stored physically in that order.

The type is abstract because the user doesn’t need to know how the list is actually implemented: an array, an ArrayList, a dynamically linked list (to be studied later this semester), in order to use it.

Assignment

You will need to implement the basic functionality of the SinglyLinkedList of the ordered list as described in the method signatures below:

- **isEmpty** – a parameterless method that returns true or false
- **insert (E value)** – inserts value into the list in the natural order, if it is not already in the list. (This list does not contain duplicate elements. A call to insert a duplicate is ignored.) If additional storage is required to fit this value in the list, the list is automatically enlarged to double its size and the values are inserted into the new list in their natural order.
- **remove(E value)** – removes value from the list, leaving the remaining elements in the natural order. (You may not assume that the value is in the list, use **contains** to be sure before attempting to remove. If value is not in the list, remove does nothing – the call is ignored.)
- **contains(E value)** – boolean method which does the obvious

- **printList** – a parameterless method that displays the data in the list in the natural order (regardless of the order in which the data may be physically stored)

Additional functionality, not necessarily useful to the user, but available for developer to test and demonstrate that above functionality provided:

printDebugList – parameterless method that displays contents of list storage (in the physical order it is stored, displaying both the actual data and the contents of the pointers/links).

Optional functionality:
- **getSize** – a parameterless method that returns the count (int) of elements in the list
- **get** (int index) – returns the value stored in some position in the list, where index could be 1 to size. If a call specifies an index out of range, the request is ignored.
- **location(E value)** – returns the position (1 through size) where this value appears in the list. If the value is not in the list, the request is ignored.

You may include any private methods necessary to implement this system in a way that is efficient.

Implementation requirements: Your list is to be a "linked list". A list is linked if each record in the list stores both data and a link/pointer to the next (in order) element in the list. Note, a linked list may be stored using dynamically allocated records which physically link to each through references (we'll study this later this term) or using a static structure such as an array or ArrayList. For this assignment, **an array is the required physical data structure**. Since we are not yet ready to build generic lists, we will make our ordered list store String objects!

Advice: Start today and ask questions of instructor and TA in person or on slack.

What to submit:
- A zipped folder containing all of your java files **and no subdirectories/folders**.
- One of the files must contain the **OrderedList** class in a file naturally named **OrderedList.java** The filename and class name must match exactly what is in bold.
- One class named **HW1** stored in a file named **HW1.java** that contains a main method that demos your ordered list and what works and specifies in comments what doesn't.
- Any other java files required of your solution. // There might not be any.
- Name your folder: **YourLastName_HW1.zip**. For me, this would be: Mountrouidou_HW1.zip.

Source
Modified by Dr. McCauley's Homework assignments