Department of Electrical and Computer Engineering
University of Puerto Rico
Mayagüez Campus

ICOM 4035 – Data Structures
Fall 2012

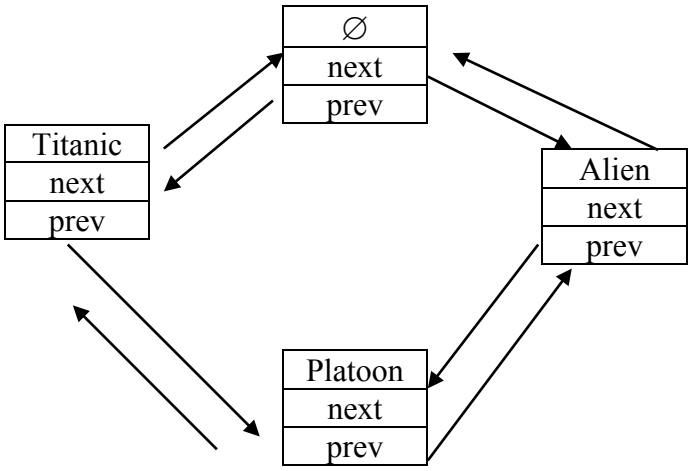
Project #2: Sorted Circular Doubly Linked Lists
Due Date: 11:59 PM-November 16, 2012

Objectives

1. Understand the design, implementation and use of a linked lists class container.
2. Gain experience implementing applications using layers of increasing complexity and fairly complex data structures.
3. Gain further experience with object-oriented programming concepts, specially templates and operator overloading.

Overview

You will design, implement and test a container class the sorted circular doubly-linked list, to be named here on as SortedCircularDoublyLinkedList. This class will implement the methods in an interface called SortedList. The sorted order will be in non-decreasing order (i.e., from small to large, equals get inserted at the first available spot). The following figure depicts an instance of this class storing names of movies (String). You class will store any value that implements the Java Comparable interface.



SortedList Inteface

The SortedList interface extends the Java Iterable interface and provides the following methods:

- public boolean add(E obj) – Adds a new element to the list in the right order The method traverses the list, looking for the right position for obj.
- public int size() - returns the number of elements in the list.
- public boolean remove(E obj) – removes the first occurrence of obj from the list. Returns true if erased, or false otherwise.
- public boolean remove(int index) – removes the elements at position index. Returns true if the element is erased, or an IndexOutBoundsException if index is illegal.
- public int removeAll(E obj) – removes all copies of element obj, and returns the number of copies erased.
- public E first() – returns the first (smallest) element in the list, or null if the list is empty.
- public E last() – returns the last (largest) element in the list, or null if the list is empty.
- public E get(int index) - returns the elements at position index, or an IndexOutBoundsException if index is illegal.
- public void clear() – removes all elements in the list.
- public boolean contains(E e) – returns true if the element e is in the list or false otherwise.
- public boolean isEmpty() – returns true if the list is empty, or false otherwise.
- public Iterator<E> iterator(int index) – Returns a forward iterator from position index, or an IndexOutBoundsException if index is illegal.
- public int firstIndex(E e) – returns the index (position) of the first position of element e in the list or -1 if the element is not present.
- public int lastIndex(E e) - returns the index (position) of the last position of element e in the list or -1 if the element is not present.
- public ReverseIterator<E> reverseIterator() – returns a reverse iterator, starting from the last element in the list.
- public ReverseIterator<E> reverseIterator(int index) – returns a reverse iterator, starting from position index in the list, or an IndexOutBoundsException if index is illegal.

Reverse Iterator

The reverse iterator provides the mechanism to traverse the list backwards, from either the last element, or an element at a given position. The methods in the interface are:

- public boolean hasPrevious() – returns true if the iterator has a previous element to give, or false otherwise (i.e., we reached the header noder).
- public E previous() – the return the next previous element, and moves the iterator backward one position.

Sorted Circular Doubly Linked List Class Container

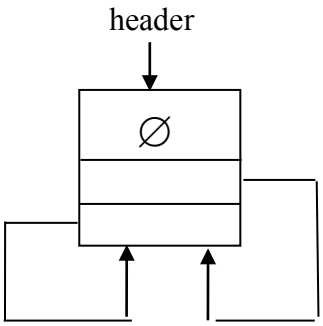
You **must** implement a sorted circular doubly linked-list class container to keep the information in a sorted list. To implement this class you will first develop a toolkit for implementing a node for circular doubly linked lists. Then, you will use this toolkit to implement a List container class that works as sorted circular doubly linked list. The circular sorted doubly linked list is a template that receives one parameter types:

1. E **extends** Comparable<E>– the object to be stored in the list. It is assumed by the interface that E implements the Java Comparable interface.

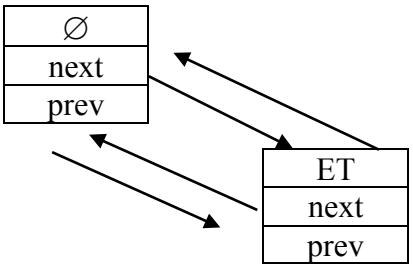
The sorted circular doubly linked list has one special node delimiting the actual nodes with the data in the list:

1. header node– has no data and points to the first and last elements in the list. If the list is empty, it points to the tail node.
2. currentSize – integer with current number of elements in list.

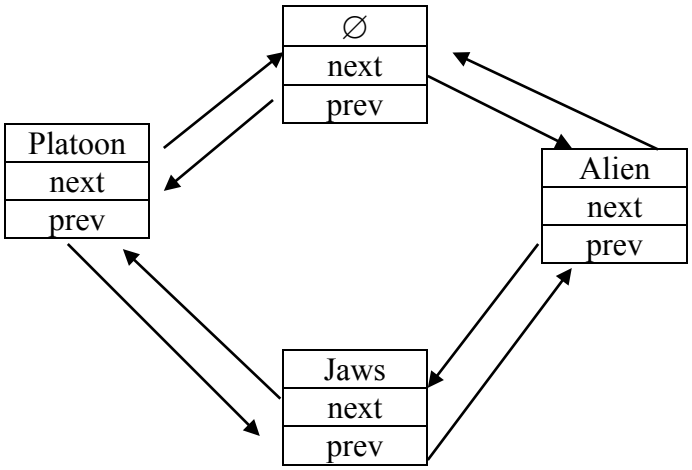
Each node in the list has a **next** field that points to the next node in the list. Similarly, each node has a **prev** field which points to the previous node in the list. Finally, the data in each node is stored in a field called **data**. The following diagrams show several cases of sorted circular doubly linked lists:



Empty list



One element list



Three element list

NOTE: For the circular sorted linked list class container, you MUST follow these guidelines.

1. The erase operator must do an in-place removal of the target node. It cannot copy nodes to a new list, skipping the node that you want to delete. Programs that deviate from this direction will not be considered a running program.
2. The clear method cannot create memory leaks.
3. The insert operation must put a new node in the appropriate sorted order. If your lists are unordered, your program will not be considered a running program.

Distribution Files

You can go to the class web page and download a zip file containing a project name p2. Just access the link named Projects, and download the sources files associated with the link: *Project # 2- Sorted Circular Doubly-Linked List*. The zip has a project and it has some of the code already implemented, and other code is blank for you to fill. In summary, you program will consist of the following files:

1. SortedList.java – interface for the sorted list ADT.
2. ReversetIterator.java – interface for the reverse iterator class.
3. SortedCircularDoublyLinkedList.java – implementation for the sorted circular doubly linked list container class.
4. ListTested.java – simple test program for the sorted circular doubly linked list container class.
5. Test1.java – JUnit test file 1.
   NOTE: YOU PROGRAM MUST PASS THIS FILE WITHOUT ERRORS IN ORDER TO BE CONSIDERED A RUNNING PROGRAM.
6. Test2.java – JUnit test file 2.
7. Test3.java – JUnit test file 3.

Due date: 11:59 PM, November 16, 2012.