CS 143

Project 2: "Circular Doubly-Linked List"

Name:

In chapter seven of the textbook is a discussion of the Java LinkedList class. On **Blackboard** you were given a partial implementation of a **SimpleLinkedList** class. Your task is to now modify that code to turn it into a doubly-linked circular list, to "fill out" the implementation and, of course. to test your new creation. A **doubly-linked list** is made up of nodes that contain a reference to the ***previous*** node as well as the ***next*** node--allowing for traversal in ***either*** direction.

A **circular doubly-linked list** has the ***next*** reference of the last node refer to the first node (instead of ***null***), and has the ***previous*** reference of the first node refer to the last node (instead of ***null***)--thus making a circle *(a list with only one node would have both references refer to itself)*.

The new class should contain all of the same public methods as in **SimpleLinkedList** re-written to take advantage of the capabilities of the new construction. It should be re-written to be ***zero-based*** *(**i.e., theList.get(0)** should return the data in the very first node, if it exists; **theList.add(0, entry)** should insert the entry in the very first node, if it exists, **etc.**)*. It should also be able to take in either negative or non-negative numbers for a position *(**i.e., theList.get(-1)** should return the data in the very last node, if it exists, **theList.add(-1, entry)** should insert the entry in the very last node, if it exists, **etc.** Also, an index beyond the size of the list should allowed **e.g.., theList.get(5)** for a list of size 4 would return the same as **theList.get(1)**)*. It should also fully impement the listIterator method (and define the corresponding inner class). In the same spirit, any method that moves linearly through the list should be *optimized* so that it will start at the front or back, depending on which is closer to the index requested (***e.g.**, if **theList*** has 5 elements, ***theList.get(1)** should start looking from the front, while **theList.get(3)** should start looking from the back)*.

You will also need to make this class implement **Iterable**, and it should contain as inner classes both **Iterator** and **ListIterator**.. The **Iterator** should stop at the end of the list *(**i.e., hasnext()*** should return false at the "end" of the list, although ***next*** would still work), so that it will allow a **for each** loop to work; the **ListIterator** should continue on unless the list is empty *(**i.e., hasNext()** and **hasPrevious()**should return true on any non-empty list)*. Be sure to fully implement ALL methods for these two classes--including the "*optional*" operations.

Finish implementing the following methods:

- **addAll(Collection<? extends E> c)**
- **addAll(int index, Collection<? extends E> c)**
- **containsAll(Collection<?> c)**
- **equals(Object o)**
- **listIterator()**
- **listIterator(int index)**
- **remove(Object o)**
- **set(int index, E element)**
- **subList(int fromIndex, int toIndex)**
- **toArray()**
- **toArray(T[] a)**

You may want to change, add, or even eliminate the protected "helper" methods. Many methods will require only minor modification. Others will require more thinking (***e.g. clear*** *will need a major re-write -- you will need to make sure ALL references in the chain are removed*). You will also need to rewrite the test driver for the class to test all the methods of the class and nested classes. This may be a command-line or GUI program--your choice. It should test all of the methods of the class, either directly or indirectly. It is up to you how it goes about doing this. You may use any class that makes sense as the data implementation (**CircularDoublyLinkedList<*String*>** works nicely).

Deliverables:

Physical:

- *The project should be turned in inside a clear plastic Deluxe Locking Project File Folder DOCU Manager or equivalent. This folder should have a simple flap to hold paper in place--NO buttons, strings, velcro, etc.* Pages should be in order, **not** stapled.
- Assignment Sheet *(this page)*, with your **name** written on it, as a cover sheet.
- Printed Source Code with Comments *(including heading blocks. Describe parameters, no line wrapping)*
- Sample Output *(printed)*
- a simple test plan including explanations of any discrepancies and reasons for each test. Show actual input and **ALL** values output as well as **ALL** expected output.

Electronic:

- All ***.class***, ***.jar, .html*** (**javadocs**), and ***.java*** files, ***zipped*** together
- Submit this single ***zip*** file by going to Blackboard, select this class, select the **Course Material** button on the left, select the ***project***, by **attach file, browse** to find the file, and **Submit**.

Due: Monday, May 6, 2013, 11:30 a.m. *(beginning of class)*