Joshua Nelson
CSCE 146

Lab 04 Report

Problem

Modify a linked list to be circularly linked, so that tail.next points to head and head.prev points to tail. Ensure that it includes appropriate methods such as hasNext, contains, getCurrent, and goToNext.

Solution

Modify the constructors of the listNode class so that unspecified links result in the node being appended. Modify the hasNext method so that it stops when current == head and it is not the first pass through the list. Modify (and greatly simplify) deleteCurrent method, as various special cases will no longer be needed.

Tests and Results

See attached CircularDoubleLinkedListTest.java; all jUnit tests pass successfully.

Problems Encountered

Java does not allow a subclass to be overridden if the superclass is also overridden; more precisely, it will compile successfully, but the ListNode of the superclass will refer to the subclass overridden, not the subclass in the same file. Example:

```
class DoubleLinkedList {

        ListNode head; // works fine

        class ListNode {

        }

}
```

```
class CircularDoubleLinkedList extends
DoubleLinkedList {

        ListNode head; // this is
DoubleLinkedList.ListNode

        // not CircularDoubleLinkedList.ListNode

        class ListNode {

        }

}
```

I eventually gave up on inheritance and polymorphism, choosing instead to copy much of the code from one file to the other. Similarly, the extended Iterator subclass could not be generic, even though the original iterator was generic:

Joshua Nelson
CSCE 146

```
class DoubleLinkedList<T> {

        class I implements Iterator<T> {

                T next() { // works fine

                }

        }

}
```

```
class CircularDoubleLinkedList extends
DoubleLinkedList {

        // compile error: Two type arguments: null
and 'T'

        class J extends I implements Iterator<T> {

        }

}
```

Conclusions and Discussion

Overall, I found this lab very frustrating. It demonstrated the limits of the Java type system, leading to a great deal of repeated code. In the future, I could instead use an abstract or interface LinkedList class, which would be implemented by SinglyLinkedList, DoublyLinkedList, and CircularlyLinkedList.

Additional Questions

1. If the reference to an object is lost and thus the object is now unreachable, what does the Java Virtual Machine (JVM) do with said object?

   Garbage collect it: mark it as free space on the heap. Note that this does not immediately overwrite the data, but merely marks the space as available; see http://www.cplusplus.com/reference/cstdlib/free/ for more details.

2. Is it possible to make a circular doubly linked list? If so how can this be achieved, and if not why?

   Yes, just have head.previous point to tail. This is much simpler (in my personal opinion) than having a tail element. See my code as an example.