CSCI 145 Week 9
Lab 6

Goals for this Lab

1. Write a generic linked list class.

Task Description

Your goal for this lab is to write a generic `LinkedList` class similar to the one in the given in the lecture notes on linked lists. You will test this class using JUnit, similar to the `LString` class for Assignment 2.

The LinkedList Class

The public constructors and methods required for the `LinkedList` class are listed here. The type `E` is the generic type of an element of the list.

LinkedList()	Construct an empty LinkedList object.
int size()	Return the size (number of items) in this LinkedList.
boolean isEmpty()	Return true if this LinkedList has no items. (This is the same as the size equal to zero.) Return false if the size is greater than zero.
void add(E value)	Add the given element, value, to the end of the list.
void add(int index, E value)	Add the given element, value, to the list at the given index. After this operation is complete, get(index) will return value. This operation is only valid for 0 <= index <= size().
E get(int index)	Return the element of the list at the given index. his operation is only valid for 0 <= index < size(). This operation does not modify the list.
E remove()	Remove and return the first element (element number zero) from the list. This operation is only valid for non-empty (size() > 0) lists.
E remove(int index)	Remove and return the element with the given index from the list. This operation is only valid for 0 <= index < size(). After this operation, all elements that had an index greater than index (as determined by get()) will have their index reduced by one.

Requirements

1. Your class must be named LinkedList.

2. Your class must provide the methods listed above for construction, accessing, and manipulating LinkedList objects.
3. Other than for testing purposes, your LinkedList class should do no input or output.
4. Your package must enable the provided test program, LinkedListTest.java, to be compiled and run correctly. Use of this program is described in more detail, below.

Testing

With this assignment, I have provided a class LinkedListTest.java that tests your implementation of LinkedList. I have not included the junit.jar file with this lab. You can use the one that was included with Assignment 2.

Compiling and running this lab is handled the same way as Assignment 2.

Here a sample of a test run for this lab:

```
Running Basic tests: constructor, isEmpty, size tests (2 tests)
Starting tests: ..
Time: 0.009
OK! (2 tests passed.)

Running Test add(value) tests (6 tests)
Starting tests: ......
Time: 0.004
OK! (6 tests passed.)

Running Test get tests (3 tests)
Starting tests: ...
Time: 0.002
OK! (3 tests passed.)

Running Test add(index, value) tests (5 tests)
Starting tests: .....
Time: 0.004
OK! (5 tests passed.)

Running Test remove tests (6 tests)
Starting tests: ......
Time: 0.005
OK! (6 tests passed.)

Congratulations! All tests passed.
```

Notes

1. You should work on this lab in a manner similar to Assignment 2:

1. Get the program to compile.
2. Do the constructor, size and isEmpty methods.
3. Do the one parameter add method.
4. Do the get method.
5. Do the two parameter add method.
6. Do the remove method.

The tests in LinkedListTest are organized in five phases corresponding to the five sets of methods, above. Like the LStringTest program, LinkedListTest will not proceed to a phase unless all tests in the prior phases have passed.

2. Most of the logic for LinkedList can be copied with minor edits from the lecture notes on Linked Lists. The only difference between this class and what is shown in the lecture notes is that this class must work for any reference type, not just for integers. To help you get started, here's the beginning of a linked list class, following the lecture notes:

```
public class LinkedList<E> {

   private class ListNode {
      E data;
      ListNode next;

      ListNode(E data) {
         this.data = data;
         next = null;
      }

      ListNode(E data, ListNode next) {
         this.data = data;
         this.next = next;
      }
   }

   private ListNode front;

   public LinkedList() {
      front = null;
   }

   public void add(E value) {
      if (front == null) {
         front = new ListNode(value);
      } else {
         ListNode current = front;
         while (current.next != null) {
```

```
                current = current.next;
            }
            current.next = new ListNode(value);
        }
    }
    . . .
}
```

3. The discussion of methods, above, says for some methods, "This operation is only valid for ...". There are no tests in `LinkedListTest` that test the behavior for invalid conditions. You might want (but are not required) to add checks for invalid conditions and raise an Exception when those conditions are not valid. The obvious exception to raise is `IndexOutOfBoundsException`. (This might be useful when you are trying to do Assignment 3.)

Turn in your program on Canvas

Turn in your `LinkedList.java` file. Look for the Assignment link on the Canvas page for Lab 6 and click that to turn in the file you just created.

Once you have completed this lab you should move on to Assignment 3.