Closed Lab #4:  
Linked Lists

1 Objectives

There are three learning objectives for this closed lab. First, you will reinforce your understanding of Java interfaces. Second, you will reinforce your understanding of object references. Third, you will gain experience implementing data structures that manage memory resources throughout their lifetimes.

2 Requirements

This lab is divided into three parts. You must satisfy the requirements for each part to complete the lab successfully.

2.1 Linked Nodes

For the first component of the lab, you will design an interface and a corresponding implementation to represent a linked node. A linked node is an object that holds data and maintains a reference to another linked node. In this case, each node will store a String object. You’ve seen this type of data structure before when implementing linked lists and binary trees.

Requirements

- You must define the interface LinkedNode. The interface must include getter and setter methods for the String data stored by the node, as well as for the reference maintained to the “next” linked node.
- You must develop the class LinkedNodeImpl, an implementation of the LinkedNode interface. The class should be as general as possible. (Hint: The implementation of LinkedNodeImpl should use LinkedNode rather than LinkedNodeImpl wherever possible.)

2.2 A Dynamic Stack

For the second component of the lab, you will design an interface and a corresponding implementation to represent a stack. The stack will hold String objects. The implementation will support stacks of arbitrary size and will use a per-element dynamic allocation strategy for efficiency. In particular, the implementation will be built using linked nodes.

Requirements

- You must define the interface Stack. The interface must include the following methods (with the appropriate signatures): push(), pop(), getTop(), getLength(), clear(), and toString().
- You must develop the class StackImpl, an implementation of the Stack interface. Your class must be implemented using a linked list, represented using the linked node components developed in the first part of the lab. Your class should be as efficient and as general as possible.
• You must test your implementation thoroughly. Your test driver should be as general as possible.

2.3 A Dynamic Queue

For the third component of the lab, you will design an interface and a corresponding implementation to represent a queue. The queue will hold String objects. The implementation will support queues of arbitrary size and will use a per-element dynamic allocation strategy for efficiency. In particular, the implementation will be built using linked nodes.

Requirements

• You must define the interface Queue. The interface must include the following methods (with the appropriate signatures): enqueue(), dequeue(), getFront(), getLength(), clear(), and toString().

• You must develop the class QueueImpl, an implementation of the Queue interface. Your class must be implemented using a linked list, represented using the linked node components developed in the first part of the lab. Your class should be as efficient and as general as possible.

• You must test your implementation thoroughly. Your test driver should be as general as possible.

3 Grading

To receive full credit for this lab, you must demonstrate (within Eclipse) that you have satisfied the above requirements. When you think you are ready, ask one of the TAs to check your solution. When they have checked your name off of their list, you are free to leave. If you are unable to complete the lab before the end of the period, but have made a reasonable attempt, you will receive 1/2 credit for the day. If the TA feels that you have not made a reasonable attempt, you will not receive any credit for your participation.

4 Collaboration

You may work independently or with one partner. One good idea would be to work with someone you haven’t worked with before. You could use this as an opportunity to test potential partners for your first course project. Remember, bad partners are bad news. Don’t choose a stinker.