**CS1027 Computer Science Fundamentals II (Java)**

**Lab 6: Linked Lists**

**Lab 6: Linked Lists**

This lab will give you experience with the following:

* Understanding how nodes work in a linked list, using the LinearNode class
* Understanding the linked list implementation of the Stack ADT (LinkedStack.java)
* Instantiating and working with generic types

**Pre-Lab Preparation**

* Make sure you have completed lab 5 (and what was done in class) in which you filled in the methods isEmpty, size, and toString in the LinkedStack class
* Review **Topic 7** — make sure you understand how the class LinearNode represents a node of a linked list. You might want to bring a paper copy of LinearNode.java to the lab
* Review **Topic 8** — make sure you understand how a stack works when implemented as a linked list
* Make sure you know which files you will need to download from the CS 1027 website in order to use LinkedStack.java

**Files Required**

* [BuildLinkedList.java](http://www.csd.uwo.ca/%7Ejcamer7/CS1027/Lab6/BuildLinkedList.java)
* [LinearNode.java](http://www.csd.uwo.ca/%7Ejcamer7/CS1027/Lab6/LinearNode.java)
* [LinkedStack.java](http://www.csd.uwo.ca/%7Ejcamer7/CS1027/Lab6/LinkedStack.java)
* All other files needed by LinkedStack.java
* [TestReversibleLinkedStack.java](http://www.csd.uwo.ca/%7Ejcamer7/CS1027/Lab6/TestReversibleLinkedStack.java)

**Exercise 1: Building a Linked List revisited**

In this exercise, you will review how to build a linked list containing integers.

1. Download [BuildLinkedList.java](http://www.csd.uwo.ca/%7Ejcamer7/CS1027/Lab6/BuildLinkedList.java). This file contains a main method that builds a linked list of integers from 1 to 10. In order to run this program, you will also need to download [LinearNode.java](http://www.csd.uwo.ca/%7Ejcamer7/CS1027/Lab6/LinearNode.java)
2. Run the program and see what is printed out as the contents of the linked list
3. **Question**: What would happen if you changed the first for loop as shown, and why?
4. for (int i = 1; i <= 10; i++)

If you are not sure, try making the change and see what happens

1. The code that traverses the linked list and displays each data item works specifically for (and only for) a linked list with exactly ten nodes. In order to make the code more general, we can use the fact that the last node always points to "nothing". So, we can use a while loop that stops looping when the next field of the current node is null. Make this change to your code: change the for loop to a while loop that will work for a linked list of any size
2. Run your BuildLinkedList program to make sure it still displays the contents of the linked list correctly

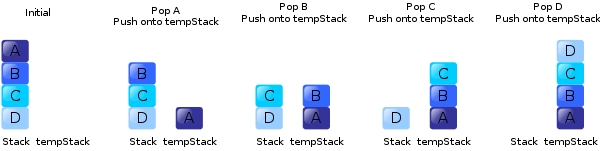
**Exercise 2: Reversing a LinkedStack**

In lab 5, you completed the isEmpty, size, and toString methods in the LinkedStack class. In this exercise, we will extend LinkedStack to create a new ReversibleLinkedStack class that contains a reverse method. This method will allow us to reverse our stacks.

1. Copy the LinkedStack class that you completed in lab 5 into your project, along with all other files required by LinkedStack
2. Create a new class ReversibleLinkedStack that extends the LinkedStack class. The following code will help you get started:
3. public class ReversibleLinkedStack<T> extends LinkedStack<T>
4. {
5. public void reverse() {
6. }
7. }

Notice that the class we are creating is a generic class — just as LinkedStack is generic — allowing us to create a ReversibleLinkedStack that can store any type of object.

1. Fill in the code for the reverse method. Use the following algorithm:
   * Create a new LinkedStack tempStack capable of storing objects of type T (e.g. LinkedStack<T>)
   * While the current stack is not empty, pop the top of the stack and add the popped element to tempStack. Note that after this step, tempStack will contain all the elements that our stack contained, except in reverse order. To understand why, see the example in [Figure 1](http://www.csd.uwo.ca/%7Ejcamer7/CS1027/Lab6/lab6.html#fig1)

  
Figure 1 - Visualizing a Stack Reversal

1. Recall that each stack has a top instance variable that points to the LinearNode representing the top of the stack. Since tempStack is reversed, we now wish to copy its top reference to the top instance variable in the current stack. Add a line to do this.
2. Since the count instance variable in the current stack is currently set to 0 (after all, we popped all of its elements), we need to update this variable as well. Add a line to set the count instance variable to the size of tempStack
3. Compile your code. Note that it does not compile since you are trying to access the top and count instance variables, which are private in LinkedStack
4. Open LinkedStack.java and change the accessibility modifiers on top and count to protected
   * **Question**: What classes can now access these instance variables in LinkedStack?
5. Compile your code. It should now compile successfully.
6. Download [TestReversibleLinkedStack.java](http://www.csd.uwo.ca/%7Ejcamer7/CS1027/Lab6/TestReversibleLinkedStack.java). Compile your code and run TestReversibleLinkedStack. You should see a list of numbers from 10 to 1, followed by a list from 1 to 10 — the result of reversing a stack using the method you just added. You will also see the result of reversing a stack containing 4 characters
7. Open [TestReversibleLinkedStack.java](http://www.csd.uwo.ca/%7Ejcamer7/CS1027/Lab6/TestReversibleLinkedStack.java) and add the following method to it:
8. private static void testStackOfStrings()

You can use the code from testStackOfCharacters() to help you. Have the method push the strings "Hello", "how", "are", "you?" to the stack, and then call the reverseStack method

1. Add a line to the main method to call the testStackOfStrings() method
2. Compile and run your code, verifying that it correctly reverses the stack of strings