

## 1 Objectives

- Practice using lists
- Learn and use Java generics
- Become familiar with iterators.

This lab will give you some practice using lists. You will start with the code for linked lists that is in your text. Then, you will write some code that adds a new feature to the linked list. You will also get practice with Java generics.

## 2 Preliminaries

Create a lab08 project for today's lab and import the files in

```
~csci204/2009-fall/student/labs/lab08
```

Commit the files to your Subversion repository. You should have gotten `LinkedList`, `ListIterator`, `ListTester`, and `AddTester`. This list uses an iterator as seen in the Big Java text book.

You can run the `ListTester` class to see the list in action before you begin the lab.

## 3 Exercise 1: Add to End Method

The linked list class of the standard library has an `addLast()` method that allows efficient insertion at the end of the list. Your job is to implement this method and add it to the `LinkedList` class that you just copied. Add an instance field to the linked list class that points to the last node in the list. Make sure that the other mutator methods update that field.

After you implement the `addLast()` method, you will need to make changes to the following `LinkedList` methods.

```
LinkedList  
removeFirst  
addFirst
```

to support the new member data.

You will also need to make changes to the following `LinkedListIterator` methods (found in the `LinkedList` class).

add  
remove

to support the new member data.

Note that your `ListIterator` class is inside the `LinkedList` class. This means you can call `LinkedList` methods from inside the `ListIterator` methods. You can see an example of this in the `ListIterator` `add()` and `remove()` methods. You can also create an instance of a `ListIterator` inside a `LinkedList` method.

Although none of these changes are difficult, you need to be careful. I suggest drawing a before and after picture before changing a method.

Test your additions using the `AddTester` class. Leave the comments code lines alone until the next exercise.

## 4 Exercise 2: Count the Number of Nodes in a List

Add a field to `LinkedList` called `size`. Use this variable to keep track of the current size of the linked list. Every time something is added to the list increase its value, and when something is removed, decrease its value. Add a public method `size()` that returns the current value of `size`. **Note:** Technically, your method should be called `getSize()` since it's a getter. We will call it `size()` to be consistent with the linked list class in the Java library.

Your second task is to write a public method `count()` for the class `LinkedList` to count the number of existing nodes in the list. This method will calculate the current size of the list by traversing through the list from beginning to end using the iterator and counting the number of nodes, rather than rely on the instance variable `size`.

Once you have written both methods, uncomment the size testing lines in `AddTester` to test your methods. Both methods should return the same value for the number of nodes in a list.

## 5 Upon Completion

Make sure that the `AddTester` tests pass and that you have added Javadoc comments to your `addLast()`, `size()`, and `count()` methods. Then, share your project with SVN and then commit all of your files to the repository. (Reminder: sharing your project with SVN does not do a commit for you).