# Lab09 - implements List<T>

Due: Wed Jun 4, 2014

Let's implement our own generic doubly-linked list. Implement `List<T>`. Then, you'll use your own list in Program #4.

This lab focuses on:

- ❏ Chapter 20 Linked Lists

**First step** - Copy the files from my `common_area/lab09` folder on the k: drive.
It may be handy to have the Javadoc for List<T> handy to answer specific questions about how the interface works.

docs.oracle.com/javase/7/docs/api/java/util/List.html

## Part 1 - Implementing the interface

Create your list. Mine is:

```
public class ProfBillList<T> implements List<T> { … }
```

When you do this, NetBeans will complain (a lot) about abstract methods in `List<T>` that are not implemented. Let's fix this. NetBeans knows how to create stubs for all your the `List<T>` methods that you're missing:

- Click on the light bulb by your class declaration
- Click on "Implement all abstract methods"

This should make NetBeans very happy and signaling green for your list class.

Copy my `Lab09` class into your project space. I've got a `main()` and lots of test methods there.

## Part 2 - Ctor, Node

Add a private `Node<T>` class to your list with `element`, `next` and `prev` class variables (because we are doubly-linked). This is a rare case where you can make your class variables `public`. Add a ctor for Node.

Add your ctor for your list. Let's maintain head and tail Nodes of your list to null.

### Part 3 - Add, get, toString

Let's do the `add()` and `get()` methods first. We'll work these out on the board. Let's also do toString() so we can start printing out lists. Test them in your `main()`.

### Part 4 - Clear, size, isEmpty

Take a break. Rest your brain. Do the 3 easiest methods of all: `clear()`, `size()`, and `isEmpty()`.
Test these new guys in your `main()`.

### Part5 - Remove, add

These are a bit harder: `remove( Object o), add( int index, T element), remove( int index)`
We'll do some on the board and then code them up.

### Part 6 - ListIterator<T> and foreach

Implement: `iterator(), listIterator()`
With these methods, for-each loops will work for your list. Excellent!

To do this, you'll need to add your own iterator class. This is challenging. So,let's get going!
It should be `private` in your list class. Mine is:

```
private class ProfBillIterator<T> implements ListIterator<T> { … }
```

This is a good description of how your ListIterator should work:

docs.oracle.com/javase/7/docs/api/java/util/ListIterator.html

### Part 7 - Etc

Keep going:

- `toArray()` and `toArray(T[] a)` (Why is an array passed in to the 2nd method?)
- `set(int index, T element)`
- `indexOf(Object o)` and `lastIndexOf(Object o)`

And so on…
Part 8 - use your list in Program #4. Good luck!