Linked List Example Code

ListInterface interface:

```
public interface ListInterface
{
    public int getSize();

    public boolean isEmpty();

    public void add(Comparable newItem);

    public void addOrdered(Comparable newItem);

    public void add(int index, Comparable newItem);

    public void remove(int index);

    public void removeAll();

    public String toString();
}
```

```
@SuppressWarnings({ "unchecked" })
public class LinkedList implements ListInterface
{
    private class Node
    {
        public Comparable item;
        public Node next;

        public Node(Comparable newItem)
        {
            this.item = newItem;
            this.next = null;
        }

        public String toString()
        {
            return this.item.toString();
        }
    }

    private Node head;
    private int size;

    // Default constructor...
    public LinkedList()
    {
        head = null;
        size = 0;
    }
```

```
@Override
public boolean isEmpty()
{
    return size == 0;
}

@Override
public int getSize()
{
    return size;
}

//  This will add a new node to the end of a list
//   (Special case for an empty list...)
@Override
public void add(Comparable newItem)
{
    Node newNode = new Node(newItem);
    Node curr;

    if(isEmpty())
    {
        this.head = newNode;
    }
    else
    {
        for(curr = head; curr.next != null; curr = curr.next);
        curr.next = newNode;
    }

    size++;  //  Don't forget to bump the size...
}
```

```
//  Add a node at a specific index in the list...
@Override
public void add(int index, Comparable newItem)
{
    if (index >= 0 && index <= size)
    {
        Node newNode = new Node(newItem);
        Node prev;

        //  Adding to the front of the list is a special case...
        if(index == 0)
        {
            newNode.next = head;
            this.head = newNode;
        }
        else
        {
            prev = find(index - 1);
            newNode.next = prev.next;
            prev.next = newNode;
        }
        size++;
    }
    else
    {
        throw new IndexOutOfBoundsException("Invalid index value: " + index);
    }
}

//  Add a node to a sorted list...
@Override
public void addOrdered(Comparable newItem)
{
    Node newNode = new Node(newItem);
    Node prev;

    prev = find(newItem);

    if (prev == null)
    {
        newNode.next = head;
        head = newNode;
    }
    else
    {
        newNode.next = prev.next;
        prev.next = newNode;
    }

    size++;
}
```

```
private Node find(int index)
{
    Node curr = head;

    for (int skip = 0; curr.next != null && skip < index; skip++)
    {
        curr = curr.next;
    }

    return curr;
}

//  Find the node that is immediately less than a value...
private Node find(Comparable pItem)
{
    Node curr = null;
    Node prev = null;

    for (curr = head;
    curr != null && curr.item.compareTo(pItem) < 0;
    prev = curr, curr = curr.next);

    return prev;
}


@Override
public void remove(int index)
{
    // Special case...
    if (index == 0)
    {
        head = head.next;
    }
    else
    {
        Node prev = find(index - 1);
        Node curr = prev.next;
        prev.next = curr.next;
        curr = null;
    }
    size--;
}

@Override
public void removeAll()
{
    this.head = null;
    size = 0;
}
```

```
@Override
public String toString()
{
    String result = "";

    //  Here's where there's often a "off-by-one" error...
    for(Node curr = this.head; curr != null; curr = curr.next)
    {
        result = result + curr.item.toString() + "\n";
    }
    return result;
}
```