Aaron Jones
Lab 11
CSCD 211
5/18/2014

```
public interface ListInterface
{
   public int getSize();
   public boolean isEmpty();
   public void addNode(Object newItem);
   public void addNode(int index, Object newItem);
   public void removeNode(int index);
   public void removeAll();
   public String toString();


}

public class ListReferenceBased implements ListInterface
{
   private Node head;
   private int numItems;

   public ListReferenceBased()
   {
      head = null;
      numItems = 0;
   }

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

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

  @Override
   public void addNode(Object newItem)
   {
      Node newNode = new Node(newItem);
      Node curr;

      if(isEmpty())
      {
         this.head = newNode;
      }
      else
      {
         for(curr = head;
             curr.getNext() != null;
             curr = curr.getNext());
         curr.setNext(newNode);
      }
      numItems++;
   }

   @Override
   public void addNode(int index, Object newItem)
   {
      Node newNode = new Node(newItem);
      Node prev;

      if(index == 1)
      {
         newNode.setNext(head);
         this.head = newNode;
      }
      else
      {
         prev = find(index -1);
         newNode.setNext(prev.getNext());
         prev.setNext(newNode);
      }
      numItems++;
   }

   @Override
   public void removeNode(int index)
   {
     if(index == 1)
      {
         head = head.getNext();
      }
      else
      {
         Node prev = find(index -1);
      Node curr = prev.getNext();
      prev.setNext(curr.getNext());
    }
    numItems--;

 }

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

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

    for(int skip = 1; skip < index; skip++)
    {
      curr = curr.getNext();
    }
    return curr;
  }

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

    for(Node curr = this.head; curr != null; curr = curr.getNext())
    {
      result = result + curr.getItem().toString() + "\n";
    }

    return result;
  }
}

public class Node
{
  private Object item;
  private Node next;
```

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

   public Node(Object newItem, Node nextNode)
   {
      this(newItem);
      this.next = nextNode;
   }

   public Object getItem()
   {
      return this.item;
   }

   public void setItem(Object newItem)
   {
      this.item = newItem;
   }

   public Node getNext()
   {
      return this.next;
   }

   public void setNext(Node nextNode)
   {
      this.next = nextNode;
   }

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

}

public class ListTester
{
   public static void main(String [] args)
```

Aaron Jones
Lab 11
CSCD 211
5/18/2014

```
   {
      ListReferenceBased myList = new ListReferenceBased();

      for(int i = 1;i < 7; i++)
      {
         double temp;
         temp = 1.23456 * i;
         Double d = new Double(temp);
         myList.addNode(d);
      }

      System.out.println(myList);

      myList.removeNode(1);
      System.out.println(myList);

      Double d = new Double(99.0);
      myList.addNode(1, d);
      System.out.println(myList);

      myList.addNode(2, new Double(68));
      System.out.println(myList);
   }

}
```