Credit Name: Chapter13
Assignment Name: Reverse List

Name: Grayson Ardron

Reflection log

Firstly I copy and pasted the code from stack 3 and changed the methods to implement linked list.

```
package StackList;

public class StackList
{
    private LinkedList data;
    private int top;

    public StackList()
    {
        data = new LinkedList();
        top = -1;

    }

    public Object top()
    {
        return(data.getHead());
    }

    public Object pop()
    {

        return(data.remove());
    }

    public void push(Object item)
    {
        data.addAtFront(item);
    }

    public boolean isEmpty()
    {
        if(data.size() == 0){
            return(true);

        } else {

            return(false);  }

    }

    public int size()
    {
        return data.size();

    }

    public void makeEmpty()
    {
        data.makeEmpty();
    }

}
```

Using the class demo I copied and pasted node and linked list

```java
package StackList;

public class Node
{
    private Object data;
    private Node next;


    public Node(Object newData) {
        data = newData;
        next = null;
    }


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



    public void setNext(Node newNode) {
        next = newNode;
    }


    public Object getData() {
        return(data);
    }


}
```

For LinkedList however I edited and snipped the layout of methods for the task at hand

```
public class LinkedList
{
    private Node head;

    public LinkedList()
    {
        head = null;
    }

    public void addAtFront(Object str)
    {
        Node newNode = new Node(str);
        newNode.setNext(head);
        head = newNode;
    }



    public Object remove()
    {
        Node current = head;
        head = current.getNext();


        return(current.getData());
            }



    public String toString()
    {
        Node current = head;
        String listString;

        if (current != null) {
            listString = current.getData() + "\n";
            while (current.getNext() != null) {
                current = current.getNext();
                listString += current.getData() + "\n";
            }
            return(listString);
        } else {
            return("There are no items in list.");
        }
```

```
public int size()
{
    Node current = head;
    int count = 0;

    if (current != null) {
        count += 1;
        while (current.getNext() != null) {
            current = current.getNext();
            count += 1;
        }
        return(count);
    } else {
        return(0);
    }
}

public void makeEmpty()
{
    head = null;
}


public Object getHead()
{
    return head.getData();
}
```

Lastly I copied and pasted the test case from class demo

```java
package StackList;

public class StackListTest
{

    public static void main(String[] args)
    {
        StackList s2 = new StackList();

        System.out.println("Adding \"red\" and \"yellow \" to stack. ");
        s2.push("red");
        s2.push("yellow");
        System.out.print("Top of stack: "+ s2.top() + "\n");
        System.out.print("Items in stack: "+ s2.size() + "\n");
        System.out.println("Removing top item.");
        s2.pop();
        System.out.print("Top of stack: "+ s2.top());
        System.out.print("Items in stack: "+ s2.size());



    }

}
```