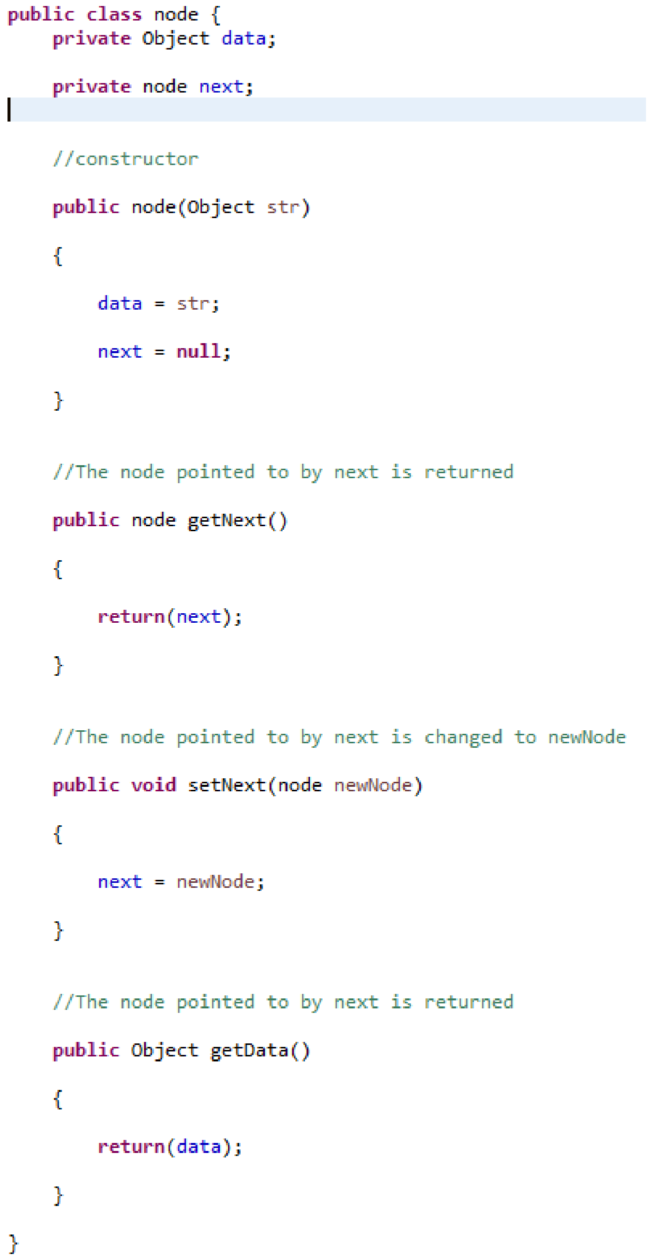
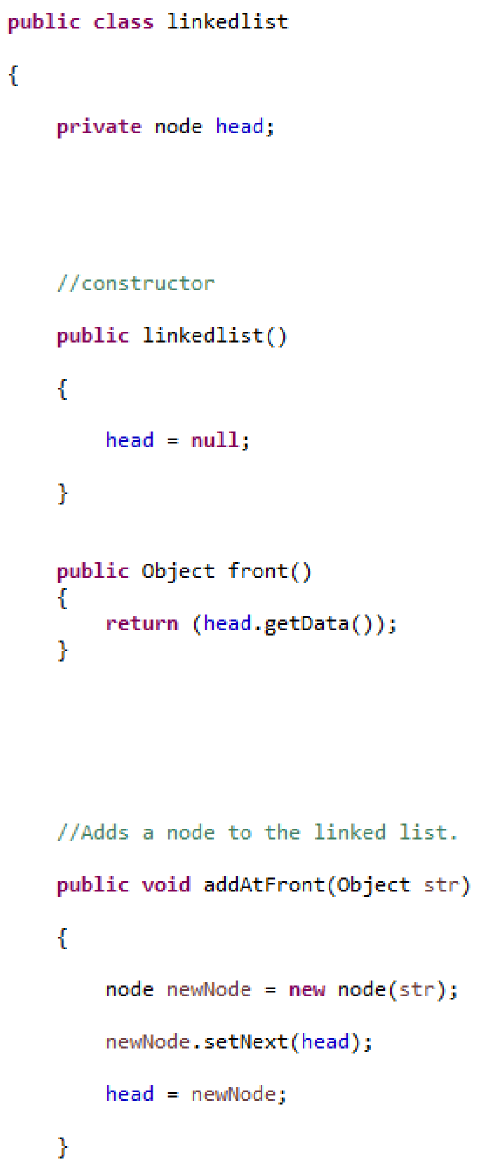


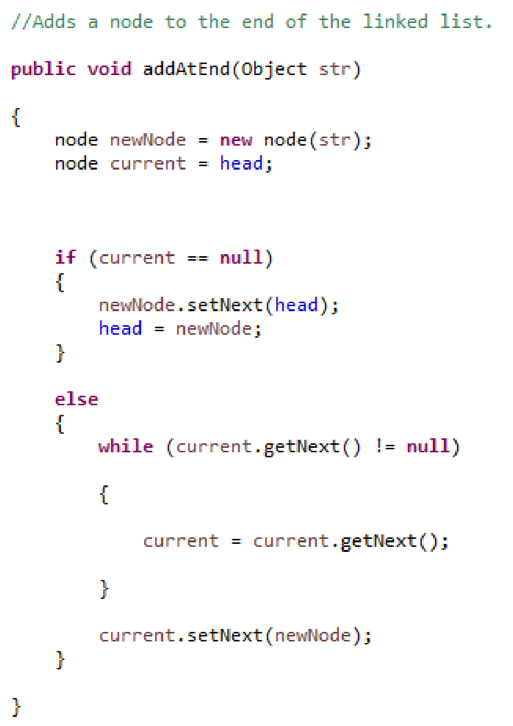
```
public class node {
    private Object data;

    private node next;


    //constructor

    public node(Object str)

    {

        data = str;

        next = null;

    }


    //The node pointed to by next is returned

    public node getNext()

    {

        return(next);

    }


    //The node pointed to by next is changed to newNode

    public void setNext(node newNode)

    {

        next = newNode;

    }


    //The node pointed to by next is returned

    public Object getData()

    {

        return(data);

    }

}
```

I started by copying node from my linked list skillbuilder and modifying it to use objects

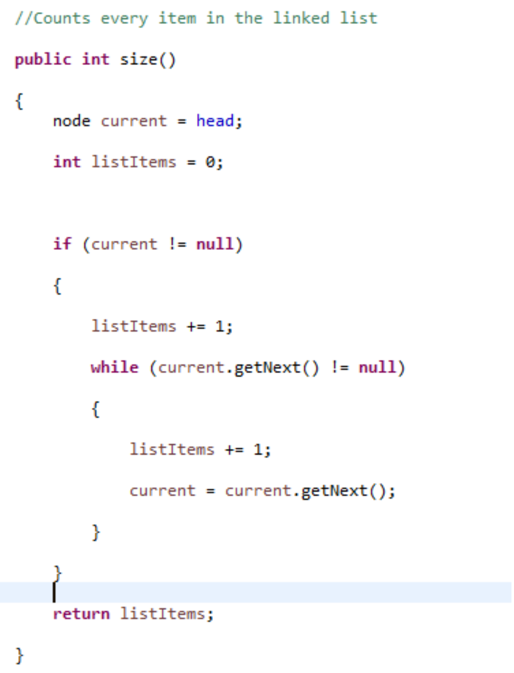


```
public class linkedlist

{

    private node head;




    //constructor

    public linkedlist()

    {

        head = null;

    }

    public Object front()
    {
        return (head.getData());
    }




    //Adds a node to the linked list.

    public void addAtFront(Object str)

    {

        node newNode = new node(str);

        newNode.setNext(head);

        head = newNode;

    }
```

Then I copied linked list from the same skill builder, then changed the called variable to Object so it would work better with the queue2 code. Additionally I added an if statement to the add at end method, to check if there was any objects already within the list.



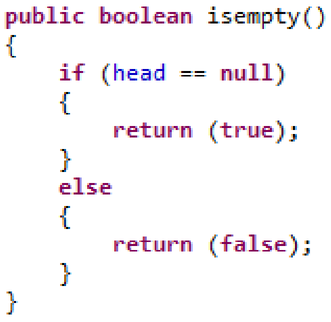
```
//Adds a node to the end of the linked list.

public void addAtEnd(Object str)

{
    node newNode = new node(str);
    node current = head;



    if (current == null)
    {
        newNode.setNext(head);
        head = newNode;
    }

    else
    {
        while (current.getNext() != null)

        {

            current = current.getNext();

        }

        current.setNext(newNode);
    }

}
```



```
//Counts every item in the linked list

public int size()

{
    node current = head;

    int listItems = 0;


    if (current != null)

    {

        listItems += 1;

        while (current.getNext() != null)

        {

            listItems += 1;

            current = current.getNext();

        }

    }

    return listItems;

}
```

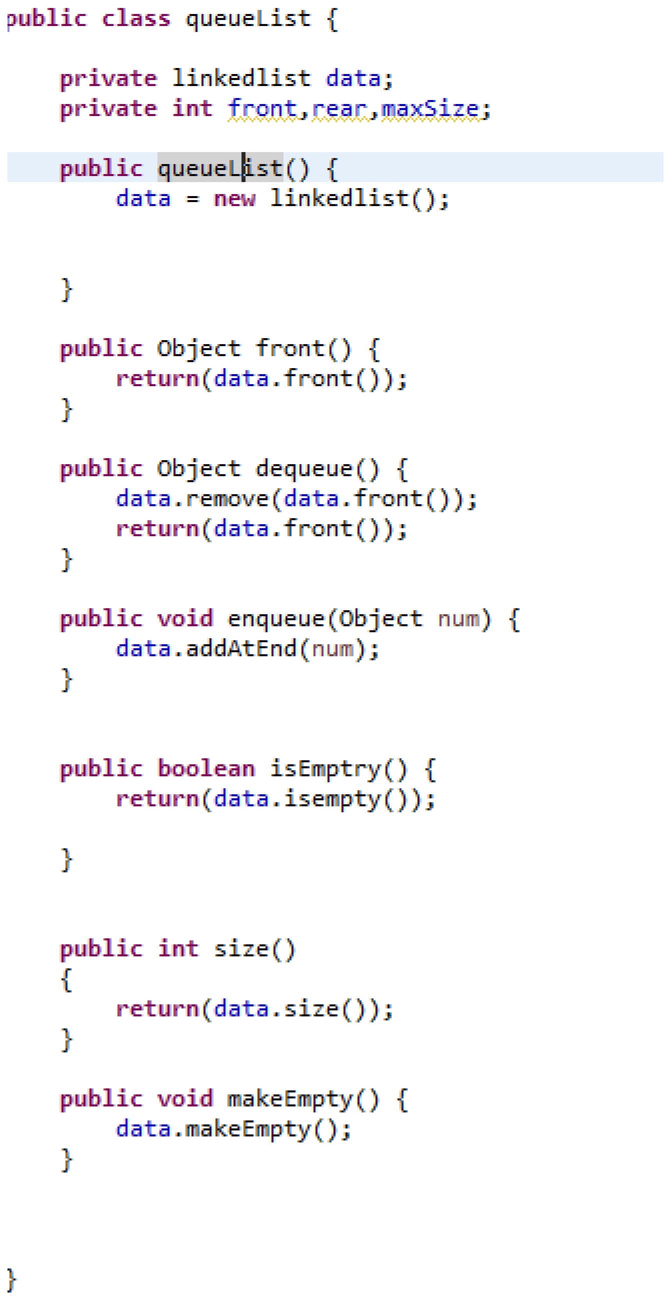
```
//Deletes a node in the linked list.
public void remove(Object object)
{
    node current = head;
    node previous = head;
    if (current.getData().equals(object))
    {
        head = current.getNext();
    }
    else
    {
        while (current.getNext() != null)
        {
            previous = current;
            current = current.getNext();

            if (current.getData().equals(object))
            {
                previous.setNext(current.getNext());
            }
        }
    }
}
```

```
//Creates a string that lists the nodes of the linked list.

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.");

    }

}
```



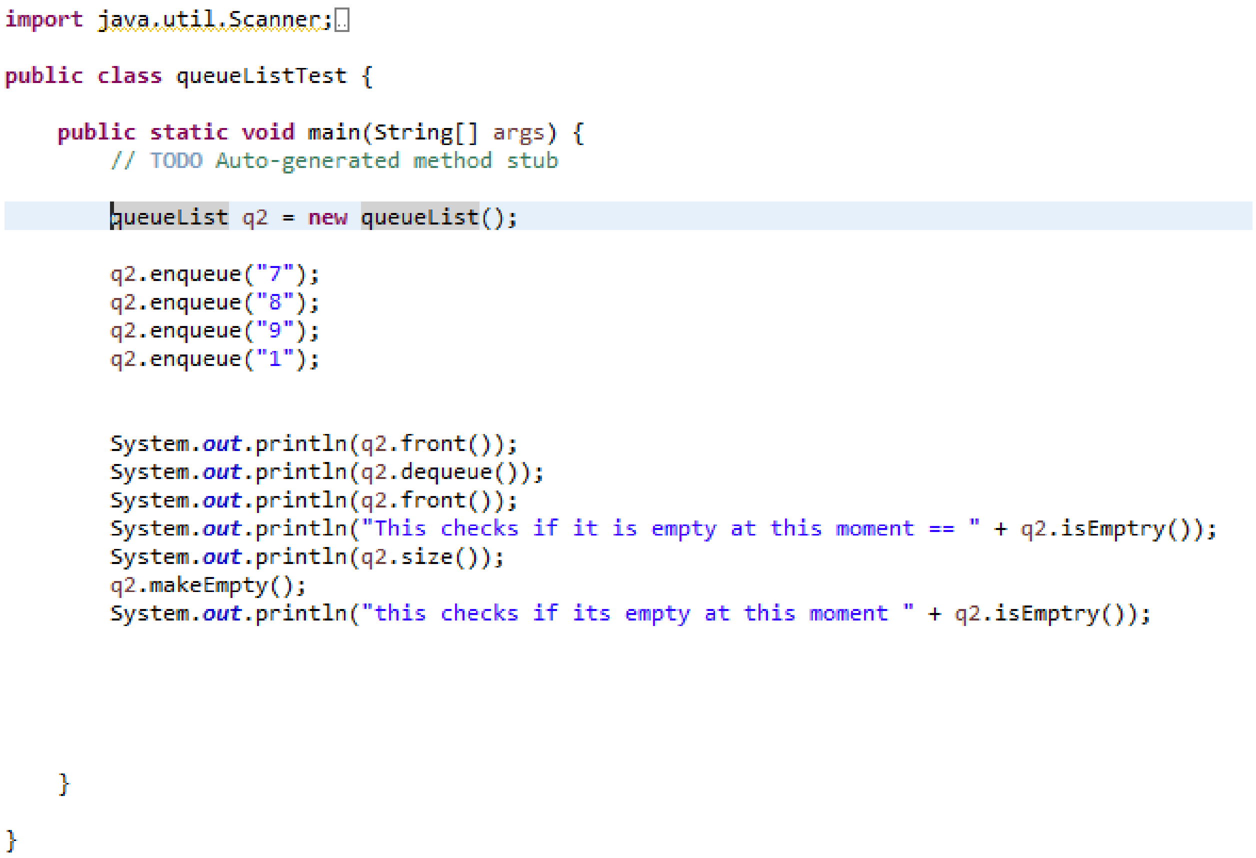
```
public boolean isempty()
{
    if (head == null)
    {
        return (true);
    }
    else
    {
        return (false);
    }
}
```

```
//Empty the linked list

public void makeEmpty()

{

    head = null;

}
```



```
public class queueList {

    private linkedlist data;
    private int front,rear,maxSize;

    public queueList() {
        data = new linkedlist();


    }

    public Object front() {
        return(data.front());
    }

    public Object dequeue() {
        data.remove(data.front());
        return(data.front());
    }

    public void enqueue(Object num) {
        data.addAtEnd(num);
    }


    public boolean isEmptry() {
        return(data.isempty());

    }


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

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



}
```

I started by recycling my code from queue2 then changing it to fit with the linked-list format. I started by changing data to a linked list, then calling all the methods from linked list into their respective categories.



```
import java.util.Scanner;

public class queueListTest {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        queueList q2 = new queueList();

        q2.enqueue("7");
        q2.enqueue("8");
        q2.enqueue("9");
        q2.enqueue("1");


        System.out.println(q2.front());
        System.out.println(q2.dequeue());
        System.out.println(q2.front());
        System.out.println("This checks if it is empty at this moment == " + q2.isEmptry());
        System.out.println(q2.size());
        q2.makeEmpty();
        System.out.println("this checks if its empty at this moment " + q2.isEmptry());




    }

}
```

I copied the test from queue 2.