a follow-up to the decorated list

another version

Singularly linked lists as before, with 1 type of node and a constant acting as a null link. This time everything is held in a single class. There is also a builder to make it easier to construct longer lists.

```
public class LList
{
	private Object data;
	private LList  link;

	public static final LList NULL_LINK = new LList()
	{	public int    length()   { return 0;  }
		public String toString() { return ""; }
	};

	public LList()
	{	this(null,null);
	}
	public LList(Object data)
	{	this(data,NULL_LINK);
	}
	public LList(Object data, LList link)
	{	this.data = data;
		this.link = link;
	}

	public static LList makeList(Object... dataset)
	{	LList node = NULL_LINK;
		System.out.println(dataset);
		return node;
	}

	public String toString()
	{	return data.toString() + ", " + link.toString();
	}
	public int length()
	{	return 1+ link.length();
	}
}
```

in operation...

```
// test run...
System.out.println( "LList example #1" );
LList x = new LList("kipper",
                  new LList("cod",
                        new LList("carp")));
System.out.println( "list = " + x.toString());
System.out.println( "size = " + x.length());

System.out.println( "\nLList example #2" );
x = LList.makeList( "herring", "cod", "kipper", "carp" );
System.out.println( "list = " + x.toString());
System.out.println( "size = " + x.length());

// output...
run:
LList example #1
list = kipper, cod, carp,
size = 3

LList example #2
list = herring, cod, kipper, carp,
size = 4
BUILD SUCCESSFUL (total time: 2 seconds)
```