[0941] Lists

Lists are by far the most popular and probably the most useful of the C# Generic Collections. Most old-school arrays can be easily coverted into Lists.

A List is a dynamically resizable, sequential collection of data items. Lists are Enumeratable, meaning that you can use foreach/in loops to get at all of their contents. Lists are also (umm...) "Lists" - meaning that they implement the (umm...) "IList" interface - meaning (finally!) that you can access their members using a numeric index just like an array. You can also use Add() and Remove() to (umm...) add and remove items from the list.

Here's an example of how to use a List:

```
using System;
using System.Collections.Generic;

namespace LWTech.Testing
{
    public class ListDemo
    {
        public static void Main()
        {
            List<string> bookTitles = new List<string>();

            bookTitles.Add("Anna Karenina");
            bookTitles.Add("Great Expectations");
            bookTitles.Add("Ulysses");
            bookTitles.Add("Lake Washington Tech Course Catalog");

            foreach (string s in bookTitles)
                Console.WriteLine(s);

            Console.WriteLine("Book #2: " + bookTitles[1]);
            bookTitles.Remove("Great Expectations");
            bookTitles.Remove("Ulysses");
            Console.WriteLine("Book #2: " + bookTitles[1]);

        }
    }
}
```

Here's the output from that program:

```
Anna Karenina
Great Expectations
Ulysses
Lake Washington Tech Course Catalog
Book #2: Great Expectations
Book #2: Lake Washington Tech Course Catalog
```

No doubt you have noticed the duplication of keystrokes in the List's declaration. In Java, you could use the so-called "Diamond operator" to eliminate that duplication. i.e., List<string> bookTitles = new List<>(); Unfortunately, that syntax is not available in C#. Instead, C# programmers recommend that you use "var" for the first part of the declaration. i.e. var bookTitles = new List<string>();

Also keep in mind that, while Lists are super-useful, there is a small amount of overhead that comes with using a List instead of an array. If you are running in an environment where memory is tight and/or performance is super-critical, you'll want to see if you can get by with using simple arrays instead.