Lists

Rob Miles

Department of Computer Science

Storing Collections

- You often want to store a number of things in your program
- We have done this by using an array

```
// Storing sales for 12 months
int [] sales = new int [12];
sales[0] = 30;
```

- Arrays are useful but they have limitations
  - Fixed size means that if we need to store 24 sales values we have to rebuild the program

The List as a better way to store, er, lists

- To solve this problem a program can use a different storage device, called a `List`
- This is one of the C# collection classes
- These are classes which are provided as part of the C# libraries
- This holds a list of references to objects (we will explore what this means a bit later in the course)

Using a List

- The great advantage of the `List` is that it grows and contracts automatically
- The `List` provides methods you can use to remove elements as well as add them
  - You can add and remove items from the list as you need
  - There is no limit to the upper size of the list
- You use it exactly as you would an array
  - You can use subscripts as you would array elements
  - You will get an exception if you go off the end of the list

Using a List

```
List<int> sales = new List<int>();
sales.Add(5);
sales.Add(10);
sales.Add(0);
sales.Add(30);
```

- This creates a `List` called `sales` and then adds four sales values to it
- The `Add` method is given the thing to be stored in the `List`
- Each time `Add` is called the `List` gets one larger
- You don't need to know how this works

Creating a List

```
List<int> sales = new List<int>();
sales.Add(5);
sales.Add(10);
sales.Add(0);
sales.Add(30);
```

- Says that we want to create a `List`
- The `List` class is a special class that was created to manage lists of items
- You can create a `List` to manage a list of any type you like

Creating a List

```
List<int> sales = new List<int>();
sales.Add(5);
sales.Add(10);
sales.Add(0);
sales.Add(30);
```

- This gives the type of the elements to be stored in the `List`
  - This uses a feature of C# called *generics*
- It is possible to create components that will work on any C# type that they are given
- If you think about it, this makes good sense
  - The behaviours of a list of strings or floats or ints will be just the same, the only difference is the thing that is being worked with

Creating a List

```
List<int> sales = new List<int>();
sales.Add(5);
sales.Add(10);
sales.Add(0);
sales.Add(30);
```

- This is the name of the list that is being created
- It serves as a reference (tag) that refers to the list object
- In this respect it works exactly as an array

Creating a List

```
List<int> sales = new List<int>();
sales.Add(5);
sales.Add(10);
sales.Add(0);
sales.Add(30);
```

- This is where the list is actually created
- This performs the same function as the new that is used to create a new array
- Note that we don't have to specify the size for a list, as it will grow as required
- Initially it is empty

Creating a List

```
List<int> sales = new List<int>();
sales.Add(5);
sales.Add(10);
sales.Add(0);
sales.Add(30);
```

- This is how we add elements to the list
- Each time we add one, it is placed in the next location in the list
- The sales value of 5 will be placed in element 0, the value of 10 will be placed in element 1 and so on

Using List Elements

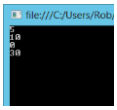
```
for (int i = 0; i < sales.Count; i = i + 1)
{
    Console.WriteLine(sales[i]);
}
```

- The program can use subscripts to get the values out of a `List`
- The `List` also has a property called `Count` which gives the number of items in the list
- A program can access particular elements in the list by using their subscript, just as for an array

Using List Elements

```
for (int i = 0; i < sales.Count; i = i + 1)
{
    Console.WriteLine(sales[i]);
}
```

- The items in the list are printed out in the order that they were entered
- The loop above would work for any size of list



Summary

- Lists can be used to store collections of items
- You don't need to specify how many elements that you want to store when you create the list
- It will expand as you add more values
- You easily find out how many elements there are in the list
- There are also methods that the List provides that can be used to delete elements in the list
- I will leave it to you find out more about the list
  - Intellisense in Visual Studio will tell you all about it

Lists, Dictionaries and Generics

- Generics introduces the idea that you can write code that manipulates objects without worrying about the precise object type
- This is particularly useful when managing collections
  - It is also used by the XNA content manager
- `Lists` can hold collections, and `Dictionary` can hold references managed by a key