Arrays, Collections, Hash Tables, Strings

Last Time

- We began looking at GUI Programming
- Completed talking about exception handling

Arrays

- Arrays are contiguous bytes of memory that store data elements that are accessed using an index into the array

Arrays

- Single dimension

```
int[] myInts = new int[20];
. . .
Console.WriteLine(myInts[i]);
```

- Multidimension

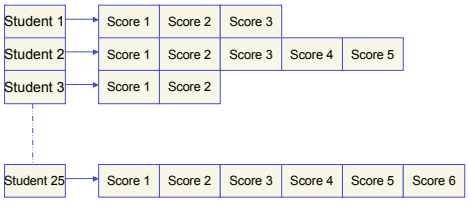
```
string[,] myStrings = new string[5,6];
double[,,] myDoubles = new double[3,8,5];
. . .
Console.WriteLine(myDoubles[i,j,k]);
```

- Jagged

```
Point[][] myPolygons = new Point[3][];
myPolygons[0] = new Point[10];
myPolygons[1] = new Point[20];
myPolygons[2] = new Point[30];
. . .
for (int x = 0; x < myPolygons[1].Length; x++)
  Console.WriteLine(myPolygons[1][x]);
```


Jagged Arrays

- Each element in the array is itself an array



foreach

```
Point[][] myPolygons = new Point[3][];
myPolygons[0] = new Point[10];
myPolygons[1] = new Point[20];
myPolygons[2] = new Point[30];

for (int x = 0;
     x < myPolygons[1].Length; x++)
 Console.WriteLine(myPolygons[1][x]);
foreach (Point p in myPolygons[1])
 Console.WriteLine(p);
```

Note: inside a **foreach** loop we have read only access to the array elements.

foreach

- **foreach** statements can be used to iterate over the elements in a collection
- Arrays in C# support the **foreach** statement
- The syntax for **foreach** is the **foreach** keyword followed by parenthesis and
  - Type of element in the collection
  - Identifier name for element in collection
  - The keyword **in**
  - The identifier of the collection

Another Example

```
int [] MyArray;

MyArray = new int [5];
MyArray[0] = 0;
MyArray[1] = 1;
MyArray[2] = 2;
MyArray[3] = 3;
MyArray[4] = 4;

foreach(int ArrayElement in MyArray)
  System.Console.WriteLine(ArrayElement);
```


Declaring Arrays

- **int[] myInts;**
  - Creates a variable that can point to an array
- **myInts = new int[100];**
  - Creates an array of 100 ints.
  - These ints are initialized to 0, and stored, unboxed, in a memory block on the managed heap.
- **Control myControls;**
  - Creates a variable that can point to an array
- **myControls = new Control[100];**
  - Creates an array of Control references, initialized to null. Since Control is a reference type, creating the array creates references—the actual objects are not created.

Array Initialization

- Single dimension
  - **int[] myInts = new int[3] {1, 2, 3};**
  - **int[] myInts = new int[] {1, 2, 3};**
  - **int[] myInts = {1, 2, 3};**
- Multidimensional
  - **int[,] MyArray = {{0, 1, 2}, {3, 4, 5}};**

Arrays are Derived from System.Array

- **int[] myInts = new int[3] {1, 2, 3};**
- **double[,,] myDoubles = new double[3,8,5];**
- Properties
  - **myInts.Length is 3**
  - **myInts.Rank is 1**
  - **myDoubles.Length is 120**
  - **myDoubles.Rank is 3**
  - **myDoubles.GetLength(1) is 8**
- Static Methods
  - **Array.Sort(myInts);**
  - **Array.Sort(keys, items);**
  - **Array.Reverse(myInts);**
  - **Array.Clear(myInts);**
  - **int i = Array.IndexOf(myInts, 17);**

Collections

- A useful namespace to know in the FCL is System.Collections
- It contains classes that serve as containers for groups of data
- Examples of the classes include:
  - ArrayList
  - BitArray
  - Hashtable
  - Queue
  - Stack

Collections

- Collections are Weakly Typed
  - They can store any kind of data
    - Primitive types
    - Any reference types
  - Could create an array of your own objects
- Need to do a lot of type casting to access and use the data in the collections
- Future versions of .NET will support generics
  - Equivalent to C++ templates

Hash Tables

- Hash tables are a form of storing data for easy and efficient retrieval
- Items are stored in a table along with with their keys
  - Key/value pairs
- Key's are hashed to determine location in table

Hash Tables

- Hash table items are stored in System.Collections.DictionaryEntry objects
- Methods in Hashtable include:
  - Add
  - Remove
  - Clear
  - ContainsKey
  - ContainsValue

Example

```
using System.Threading;
using System.Windows.Forms;
using System.Collections;
class DaysOfTheWeek
{
  public static void Main()
  {
    Hashtable days = new Hashtable();
    days.Add( "Sunday", "الأحد" );
    days.Add( "Monday", "الاثنين" );
    days.Add( "Tuesday", "الثلاثاء" );
    days.Add( "Wednesday", "الأربعاء" );
    days.Add( "Thursday", "الخميس" );
    days.Add( "Friday", "الجمعة" );

    days["Saturday"] = "السبت";

    string word = "";
    foreach( DictionaryEntry entry in days )
      word += entry.Key + " : " + entry.Value + "\n\n";

    MessageBox.Show( word );
  }
}
```


Resizable Arrays

- The arrays that we have seen so far are not resizable
- System.Collections.ArrayList encapsulates dynamic arrays
- Methods supported by ArrayList include:
  - Add
  - Clear
  - Clone (creates a shallow copy)
  - Remove
  - Sort

Examples

- Which is more efficient? Why?

```
ArrayList list = new ArrayList();
for( int i = 0; i < 100000; i++ )
  list.Add( i );
```

```
ArrayList list = new ArrayList(100000);
for( int i = 0; i < 100000; i++ )
  list.Add( i );
```

Examples

- To retrieve an item from an ArrayList
  - int i = (int) list[0];
  - Why do we use (int)
- To assign a value to an element in the array
  - list[0] = 333;

Example

- We have two options for looping through all the elements in the array
  - Count property
  - foreach

```
for( int i = 0; i < list.Count; i++ )
  Console.WriteLine( list[i] );

foreach( int i in list )
  Console.WriteLine( i );
```


More ArrayList Methods

- Removing items from an array automatically shifts the items in the array up
- Memory is not released as soon as an item is removed
- To downsize an ArrayList to automatically fit the contents use TrimToSize method

Example

- Let's look at the example in Figure 3.3 on pages 72-74 of the book

Strings

- Another way of finding the classes available through the .NET FCL is the Object Browser
  - View menu, then Object Browser
- You will find the classes under mscorlib
- String is under mscorlib->System
  - It's in the System namespace

Strings

- The methods by the String class include:
  - ToLower
  - ToUpper
  - Replace
  - Length
  - IndexOf

Example

```
string theString = "Programming Microsoft .NET";
Console.WriteLine("default: \t {0}", theString);
Console.WriteLine("lowercase: \t {0}", theString.ToLower());
Console.WriteLine("uppercase: \t {0}", theString.ToUpper());
Console.WriteLine("replace: \t {0}",
                  theString.Replace(".", "dot "));
Console.WriteLine("length: \t {0}", theString.Length);
Console.WriteLine("\"Microsoft\" occurs at character {0}",
                  theString.IndexOf("Microsoft"));
Console.WriteLine();
Console.WriteLine();
Console.WriteLine("Please press enter key to quit");
Console.ReadLine();
```


Summary

- Arrays
- Collections
  - Hash Tables
  - Resizable Arrays
  - Strings
- Completed pages 66 - 74