C# Debugging Objects

To access the debugging controls `using System.Diagnostics;` needs to be imported.

This gives access to the Debug object. At its simplest `Debug.WriteLine();` will write to the output Window. The output Window is not automatically visible. It needs to be added from the Debug_Windows menu. Here is some typical output data.

```
The thread 'vshost.LoadReference' (0x127c) has exited with code 0
(0x0).
'testDebug.vshost.exe' (Managed (v4.0.30319)): Loaded
'C:\Users\graydeb.ADMIN0\AppData\Local\Temporary
Projects\testDebug\bin\Debug\testDebug.exe', Symbols loaded.
'testDebug.vshost.exe' (Managed (v4.0.30319)): Loaded
'C:\Windows\Microsoft.Net\assembly\GAC_MSIL\System.Configuration\v4.
0_4.0.0.0__b03f5f7f11d50a3a\System.Configuration.dll', Skipped
loading symbols. Module is optimized and the debugger option 'Just
My Code' is enabled.
test
test
```

Apart from having a Window that code can be copied from this is not a lot more useful than Console.WriteLine().

There are other more useful methods of Debug.

```
int num = 1;
Debug.WriteIf(num == 1, "num is one");
Debug.WriteIf(num != 1 , "num is not one");
```

With this sort of access we can monitor the state of the program without needing to break and step through the code.

A useful trick is to send the debug messages to a file.

```
TextWriterTraceListener tr1 = new
TextWriterTraceListener(System.IO.File.CreateText("Output.txt"));
Debug.Listeners.Add(tr1);
int num = 1;
Debug.WriteIf(num == 1, "num is one");
Debug.WriteIf(num != 1 , "num is not one");
tr1.Close();
```

Note that the `TextWriterTraceListener` is created before any Debug commands and closed when it is done. By default the file will be in bin\debug for the project.