■ ECE/CS 4984: Wireless Networking and Mobile Systems ■

Pre-lab and In-class Laboratory Exercise 3 (L3)

Part I – Objectives and Lab Materials

Objectives:

The objectives of this lab are to:

- Introduce the C# programming language
- Introduce the .NET compact framework development environment

After completing the assignment, you should be able to:

- Write *simple* C# applications
- Develop and deploy simple mobile applications using the Visual Studio environment

Hardware to be used in this lab assignment:

- Dell Latitude C640 notebook
- iPAQ 3850 with cradle

Software to be used in this lab assignment:

- Notebook PC: Windows 2000, Microsoft ActiveSync, Microsoft Visual Studio 2003, Microsoft .NET Compact Framework, Microsoft Visual C# .NET 2003

Part II – Pre-lab Assignment

This portion of the assignment should be completed *prior* to the in-class lab session.

There are 2 goals for this pre-lab. The first is to get a glimpse of the C# language. All future mobile applications will be designed using C# and the .NETcf so this is the time to get acquainted with them. The second goal is to learn about the .NETcf and what it does. Together, these two allow fast and easy development of mobile applications.

Reading:

- The .NET overview document

C# and .NET Tutorials

Here are 3 tutorials that give a good introduction to the C# language and how to use it. It is recommended that the sample code in them be tested and tinkered with. To build C# applications create a new project and choose the "Console Application" template from the "Visual C# Projects" section. This template will allow you to use normal console input and output functions, saving you from the graphical programming. The "useful sections" are sections that should be examined, while the "optional sections" are suggested for your personal enrichment. C# is a fully functional language that can be used for normal programming as well as mobile application development.

- Introduction to the C# language
    - Softsteel Solutions C# tutorial – http://www.softsteel.co.uk/tutorials/cSharp/cIndex.html
        - Useful sections: 1-4, 6-13

        - Optional sections: 17
    - C# Station Tutorial – http://www.csharp-station.com/Tutorial.aspx
        - Useful sections: 1-5, 7
        - Optional sections: 6,8
    - FunctionX C# Tutorial – http://www.functionx.com/csharp
        - Useful sections: 1-10

Note: For an in-depth study of C# there are numerous books. A recommendation is Programming C# (3rd Ed.) by Jesse Liberty from O'Reilly publishers.

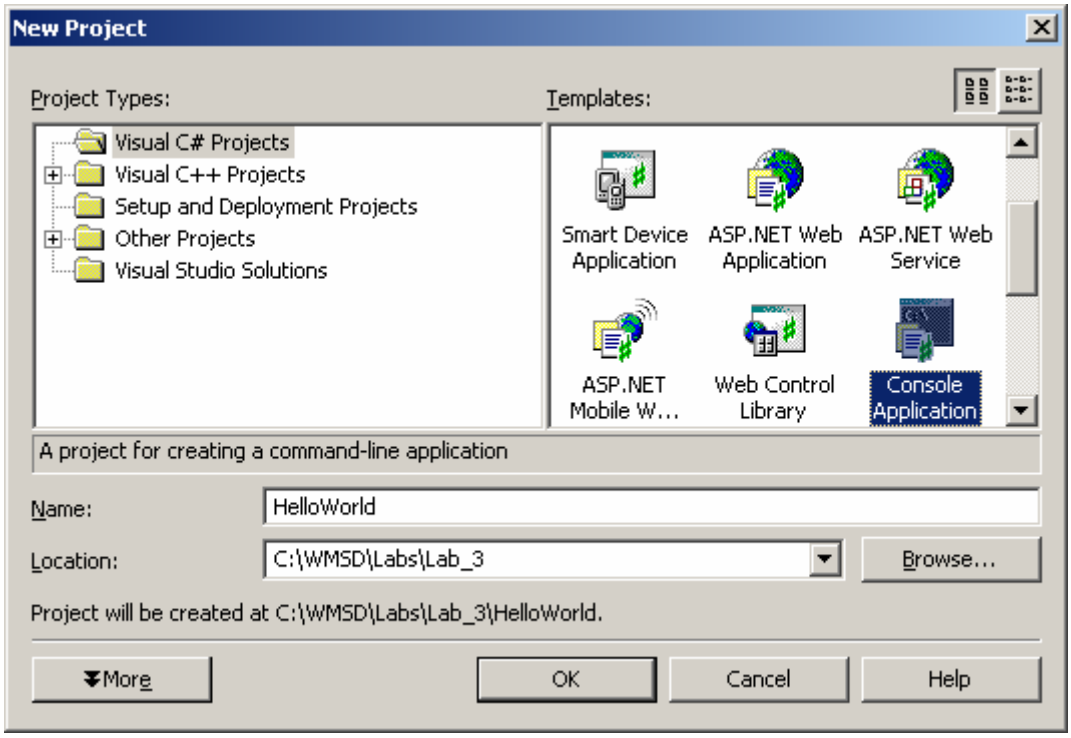
- Introduction to the .NET compact framework
    - Fundamentals of Microsoft .NET Compact Framework Development - http://www.msdn.microsoft.com/vstudio/default.aspx?pull=/library/en-us/dnnetcomp/html/net_vs_netcf.asp
- .NET Compact Framework reference
    - Unfortunately, there isn't really a good class list reference for the .NETcf. The best way to find information on a class is to search for it in the MSDN library and look at the full .NET framework documentation for it. The full .NET documentation will specify which methods and attributes are implemented in the .NETcf environment.

Note: For an in-depth study of the .NET Compact Framework there are a few books. The one used in developing this course is Microsoft .NET Compact Framework by Wigley and Wheelwright, from Microsoft Press.

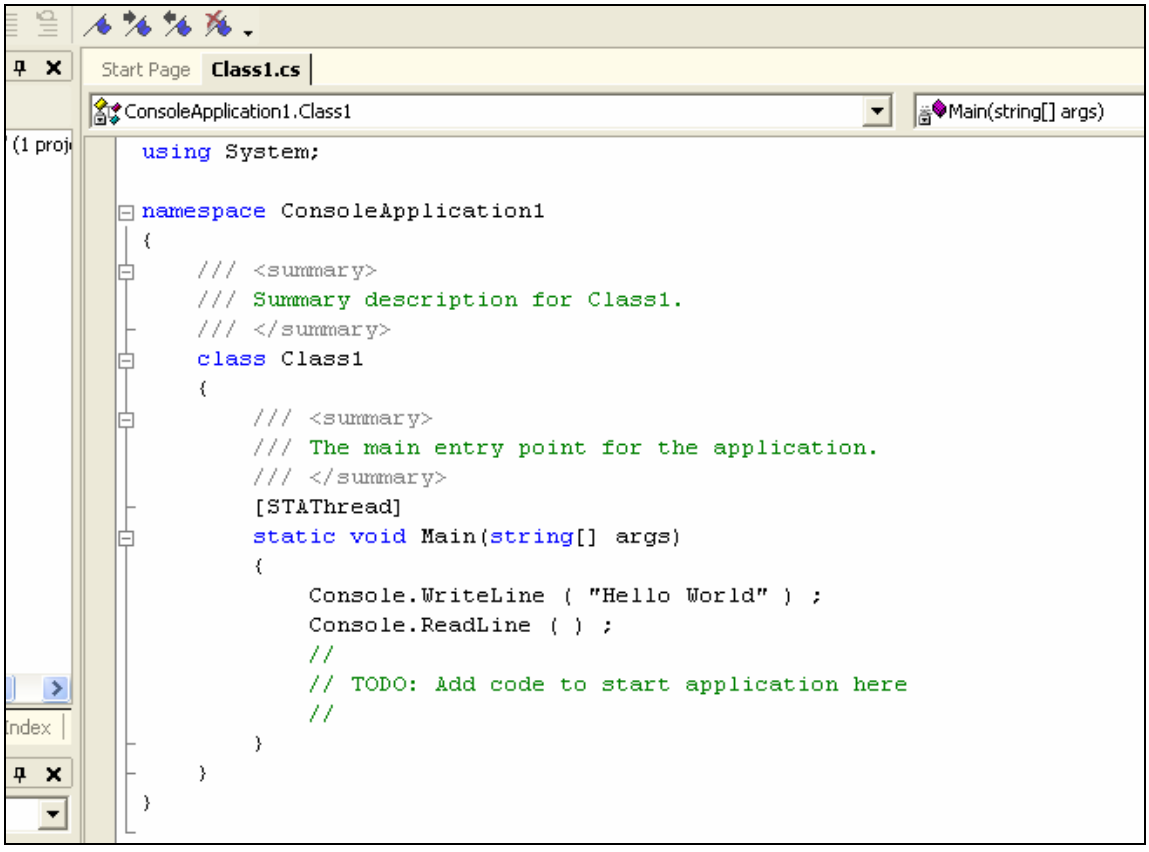
Part III – In-class Lab Assignment

1. Program Structure

Like C++, C# is case-sensitive. Semi colon (;) is the statement separator. Unlike C++, there are no separate declaration (header) and implementation (CPP) files in C#. All code (class declaration and implementation) is placed in one file with extension *cs*. You will see this with Hello world program. Start VS .NET. Select "Console Application" and change "Name" and "Location" value in the New Project window.



In the Main () function add the traditional "Hello World" statement as shown below. To keep the window from closing, also add a ReadLine statement.

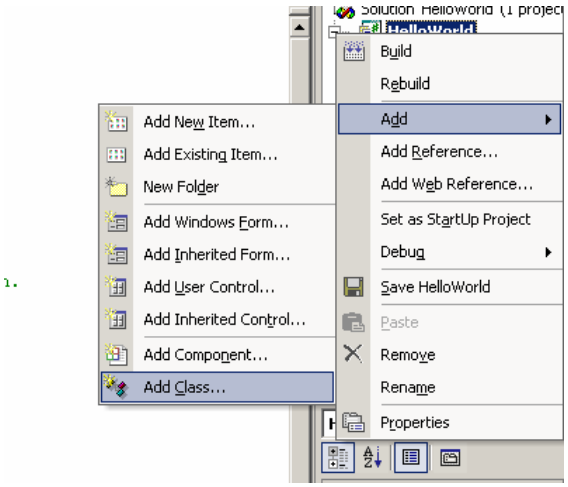


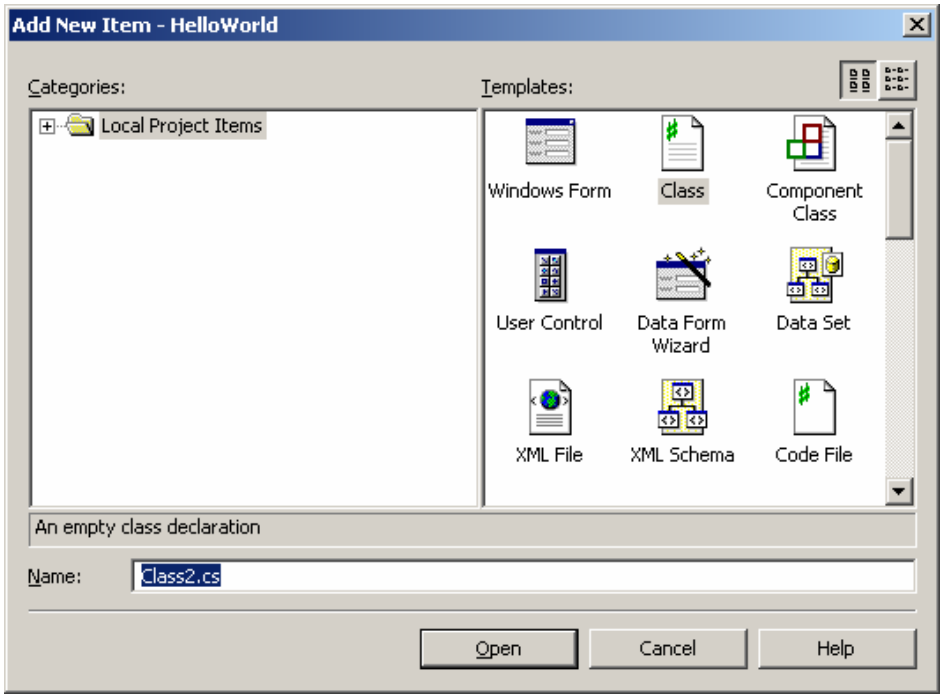
Build your 'Hello World' program by selecting Build->Build Solution. To run the console application program you made select Debug-> Start. Press <enter> to close the application.

2. Namespace

Every class is contained in a namespace. A namespace usually has multiple classes, but a class may only have on namespace. Namespaces are exactly the same concept as in C++, but in C#, namespaces are used more frequently. You can access a class in a namespace using dot (.) qualifier. HelloWorld is the namespace in hello world program above.

Now consider you want to access the HelloWorld class from some other class in some other namespace. Right click in the "Solution Explorer" and click "Add Class" to add another class.





Modify the generated code with the code given below.

```
using System;
namespace AnotherNameSpace
{
        public class AnotherClass
        {
                public void func()
                {
                        Console.WriteLine("Hello World 2");
                }
        }
}
```

There are 2 ways to access classes in other namespaces.

The first is to specify them using the (.) qualifier. An example of this is given below.

```
using System;

namespace ConsoleApplication1
{
        class Class1
        {
                [STAThread]
                static void Main(string[] args)
                {
                        Console.WriteLine ( "Hello World" ) ;
                        AnotherNameSpace.AnotherClass obj =
                                new AnotherNameSpace.AnotherClass() ;
                        obj.func ( ) ;
                        Console.ReadLine ( ) ;
                }
        }
}
```

The second is the 'using' keyword. The using keyword tells the compiler to look in this class when trying to find a class or method. An example of this is with writing to the console. I could type out System.Console.WriteLine(…) without using the 'using' keyword. Or at the top of the program, I could add 'using System;' and then in order to output all I need is Console.WriteLine (…); . The same goes for our example classes.

```
using System;
using AnotherNameSpace ;

namespace ConsoleApplication1
{
        class Class1
        {
                [STAThread]
```

```
                static void Main(string[] args)
                {
                        Console.WriteLine ( "Hello World" ) ;
                        AnotherClass obj = new AnotherClass () ;
                        obj.func ( ) ;
                        Console.ReadLine ( ) ;
                }
        }
}
```

Take screen shots of your programs running and include these in your report.

3. Variable types and conversion

C# makes dealing with strings much simpler than with C++. Because of this, I makes frequent use of strings for reading in data, especially when it comes to console input. In C++ where I could declare an integer variable and read something directly into it, C# does not allow me to do this. The Read and ReadLine methods read things as chars and strings. This may seem somewhat more of a pain, but it can be dealt with easily.

System.Convert converts most of the .NET base types to other base types. To test this out enter a number and convert it to a number. Yes, it does sound strange.

Add the following code to convert to different types. (We have no exception handling for input data to simplify. Input value should be integer or float)

```
string org = Console.ReadLine();
double num = System.Convert.ToDouble(org);
Console.WriteLine(num);

Int32 iNumber = System.Convert.ToInt32(num);
Console.WriteLine(iNumber);

Char chrNumber = System.Convert.ToChar(org[0]);
Console.WriteLine(chrNumber);
```

Take a screenshot of your program running and include it in your report.

4. The in-class portion of this lab will be to implement the traditional hello world program (plus a small twist) using C# and the .NETcf. This exercise will show how to create, debug, and deploy a mobile application.

- ❑ See the tutorial.