Portland State ACM
C# Hello World Tutorial
Author: Sean Walsh

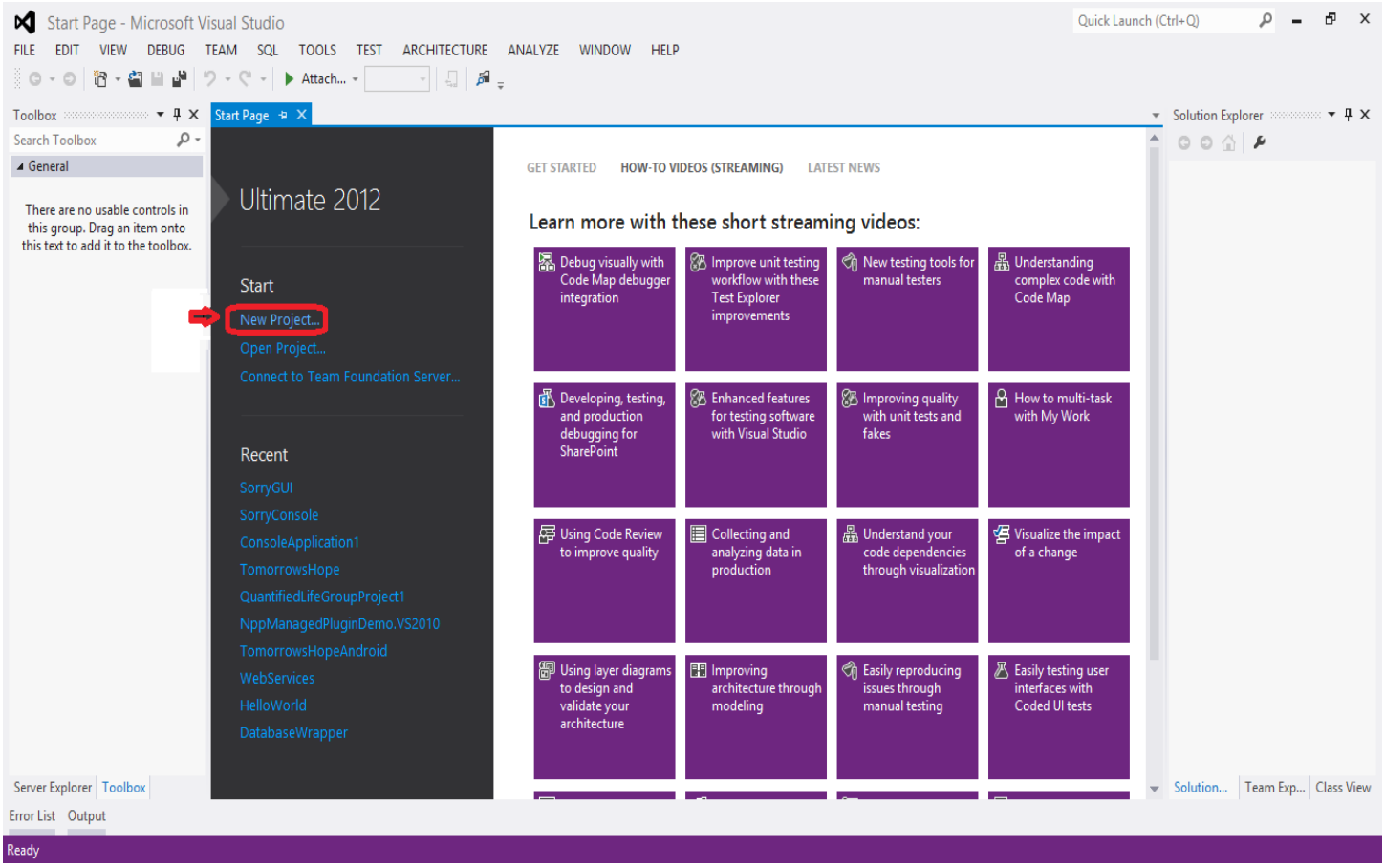
C# Hello World Tutorial

When you first start Visual Studio 2012, you will see the following screen. Depending on the edition you are using (Express, Professional, Ultimate), your screen may look a bit different than this screen, but will be very similar.

If you would like to get Visual Studio Ultimate 2012 (or any other recent version of Visual Studio), it is freely available through the Portland State University MSDNAA account.

Follow this link:

Just log in with your MCECS credentials.
Choose either File->New Project, or click the New Project link highlighted in the screenshot below.



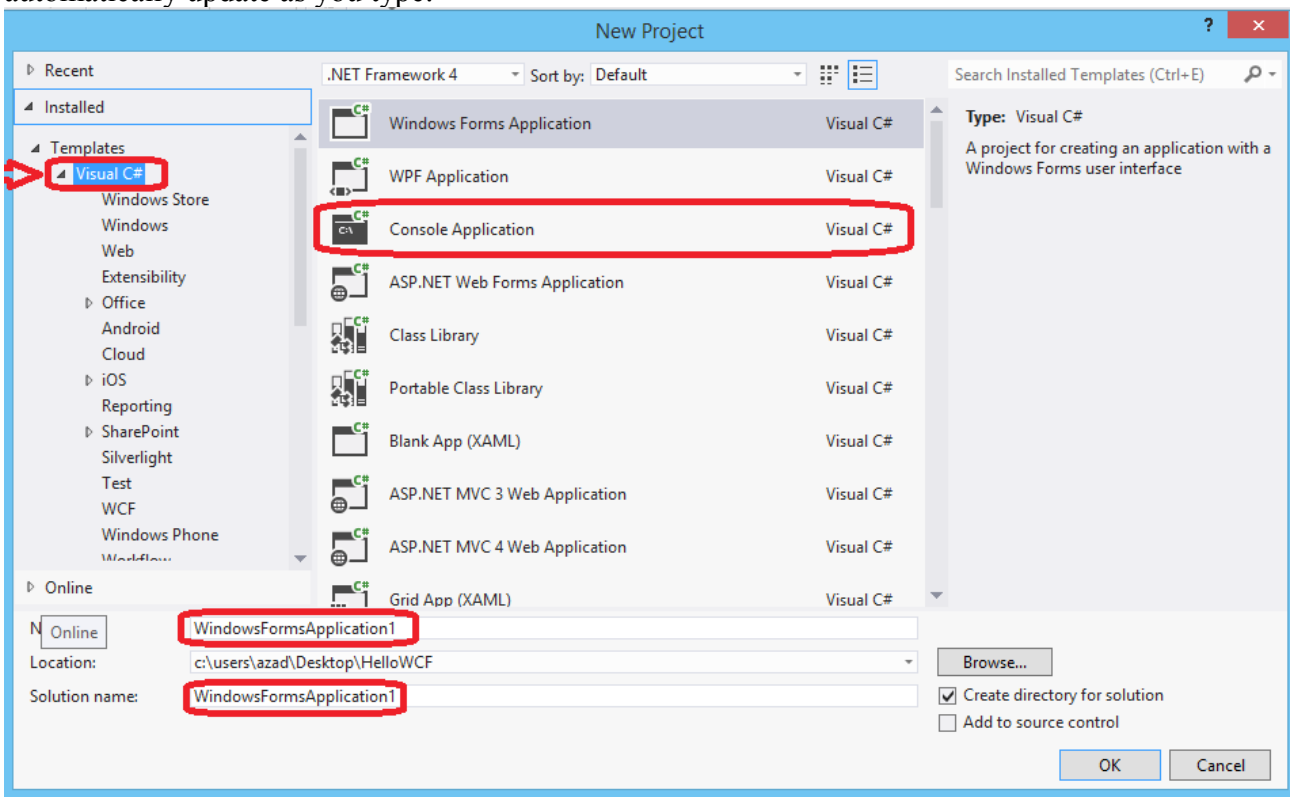
Once you have the New Project Dialog open, your screen should look like the following screenshot below.

Take note of the highlighted fields in Figure 2.

1. Choose Visual C# in the left pane as shown. If you do not see Visual C#, you are either using an Express edition and it was not installed or you had a faulty Visual Studio installation and should re-run the installer and choose “Repair”.

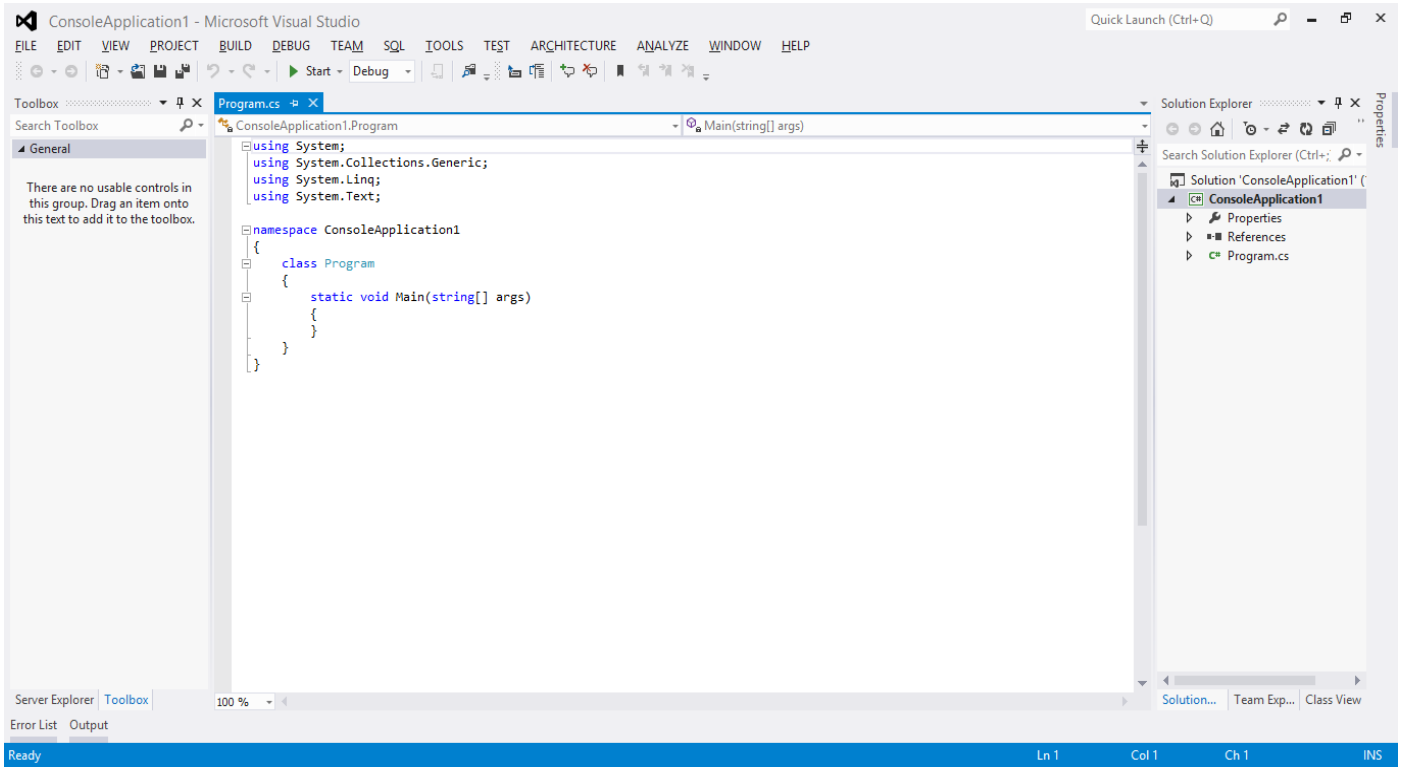
2. Choose Console Application from the center pane as shown.

3. Choose your Project Name and Solution Name as shown in the figure. The directory will automatically update as you type.



For our purposes, we will change where it says WidnowsFormsApplication1 to HelloCSharp.

Click OK, and you will be given the following screen.



Please pay notable attention to the center and right panes, as these are where you will be spending a lot of your time. Lesser noted, but still important is the left pane which is titled “Toolbox”. This pane is where you will do work with drag-n-drop GUI applications, sometimes known as What You See Is What You Get (or WYSIWYG – pronounced Wizzy Whig by many of us code monkeys.) In a Console Application, there is no default Window handler and therefore, no toolbox to work with. So if we direct our attention to the right pane, we see “Solution Explorer” at the top of the sub window. We also see a directory tree of our solution with its projects. You might be asking at this time what exactly the difference between a Solution and a Project is.

The answer is amazingly simple and intuitive. A Solution is the framework in which applications live. A Solution can have many Projects, which can communicate with each other. In an Integrated Development Environment (IDE) such as Visual Studio or Eclipse, this concept is quite common to find.

We see Properties and References, which for now, we will largely ignore. As the projects you make get larger and you start either importing C# native libraries or using outside Application Programming Interfaces (APIs), with examples including Connector/.NET (for MySQL interactivity), Skype API (for integrating Skype functionality into your program), and scores of other examples.

Finally, if we focus our attention on the center pane, we see our code file. C# files are suffixed with .cs, so we can identify our program in the Solution Explorer or the top of the center pane as Program.cs, which is the default name for a C# program, which is analogous to main.cpp if you are familiar with C++.

Now, if we examine the code, we find some similarities to other C-based languages such as C, C++, or Java.

First we see the code which is auto-generated for us:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }
}
```

The first four lines are using statements, which in C++ would look like this:

```
#include <iostream>;
```

Or in Java as:

```
import java.util.*;
```

The Java code highlighting example was taken from Eclipse, a very popular and free open-source Java editor.

Next, we see the namespace declaration. In C++, we have the `using namespace std` declaration, which tells C++ to not need to prefix calls such as cout and cin with std, an example being:

```
std::cout << "Hello World";
```

vs.

```
cout << "Hello World";
```

In C#, we have the same concept. If we have a class in a namespace, then it means that we can call that namespace using a **reference** (noted earlier in the Solution Explorer pane) to include the namespace in our program (accomplished with the **using** keyword.)

Classes are used the same as in C++, as is the main method. One notable difference about main in C# is that it is always written as follows:

```
static void Main(string[] args)
```

This means that the method Main only occurs once, and every variable declared within Main is also static, which is to say that there is only one copy of every variable declared within Main.

With all this background out of the way, we will now write and run a program in C#. Our program will take input from the user and output the information obtained from the user plus a string.

Enter the following lines into the Main function:

```
String name;
Console.Write("Hello! Please enter your name: ");
name = Console.ReadLine();
```

```
Console.WriteLine("You entered: " + name);
Console.WriteLine("Goodbye!");
```

We have done a lot here in just a few short lines of code.

First, we declared a String variable called name. This is synonymous with how you would do the same task in C++ or Java.

Next, we write output to the screen, similar to System.out.println() in Java, and quite differently from how we would do a cout in C++. This is because Console.WriteLine is a method call, and it takes as an argument a string.

Then we assigned the value of whatever line is read from the user to the variable name.

Finally, we print the output to the user using Console.WriteLine, which will write out the string within, and then print a new line afterwards. Instead of using << like C++ would, we use + to concatenate strings, just like we would in Java.

C# is smart enough to imply from the context that we want to make a string out of the two pieces of information.

Press CTRL+F5 or choose Debug->Start Without Debugging to start the program.

Congratulations! You have just written your first program in C#.

Exercise: See if you can modify the program to ask your favorite number and print the following: *Hello, your name is [name], and your favorite number is [number]*.

Where [name] is your name, and [number] is the number you entered.