

7SENG001W Tutorial 1 Intro to Visual Studio and C#

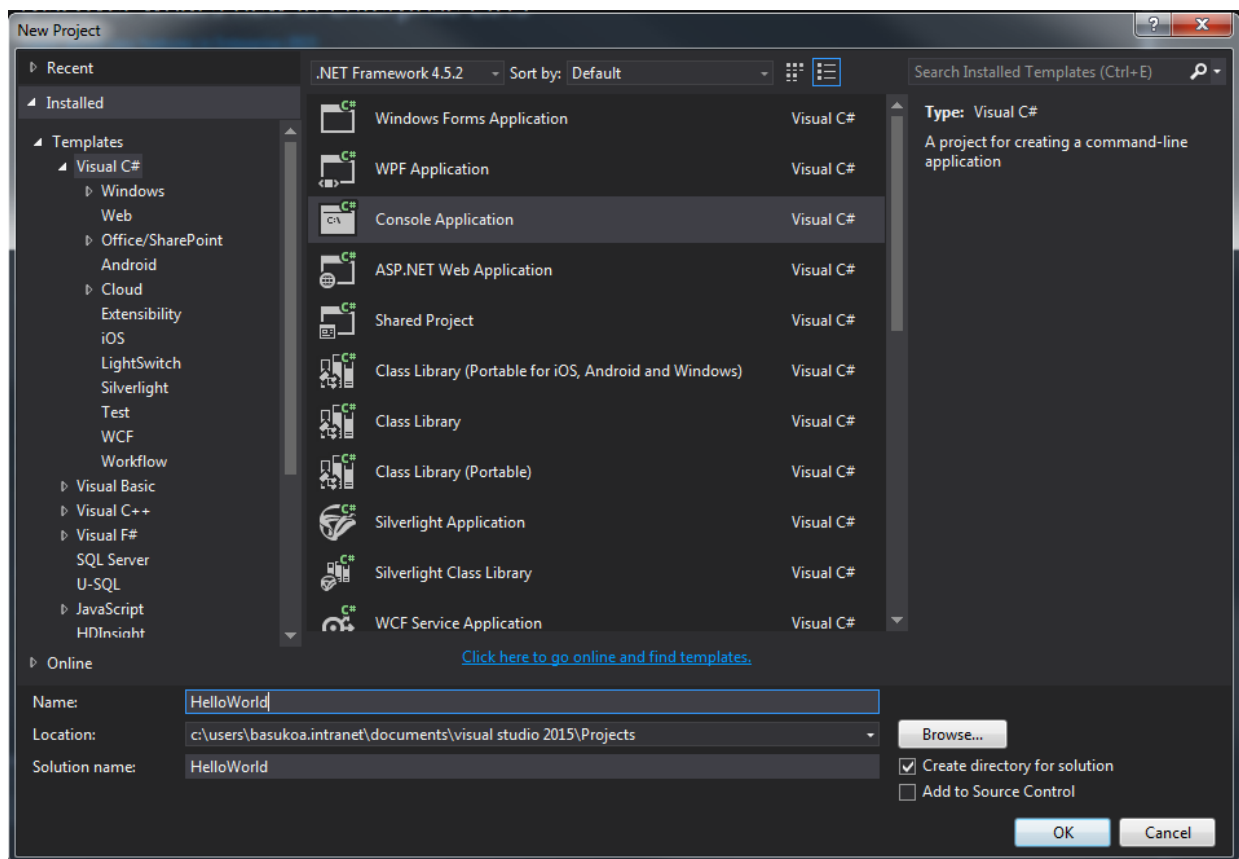
- Understand the basic structure of a C# program
- Obtain a basic familiarization of "Namespace"
- Learn how to obtain command-line input
- Learn about console input/output (I/O)
- Creating new classes
- Getting and Setting parameters
- Familiarization with Visual Studio Community 2019

Note that we will revisit and continue this tutorial next once we have discussed more of the C# language.

There are some new language features you won't have seen yet but we will explore these in detail over the next few weeks. This is likely to be the only console application we create as we shall do only event-driven programming from next week.

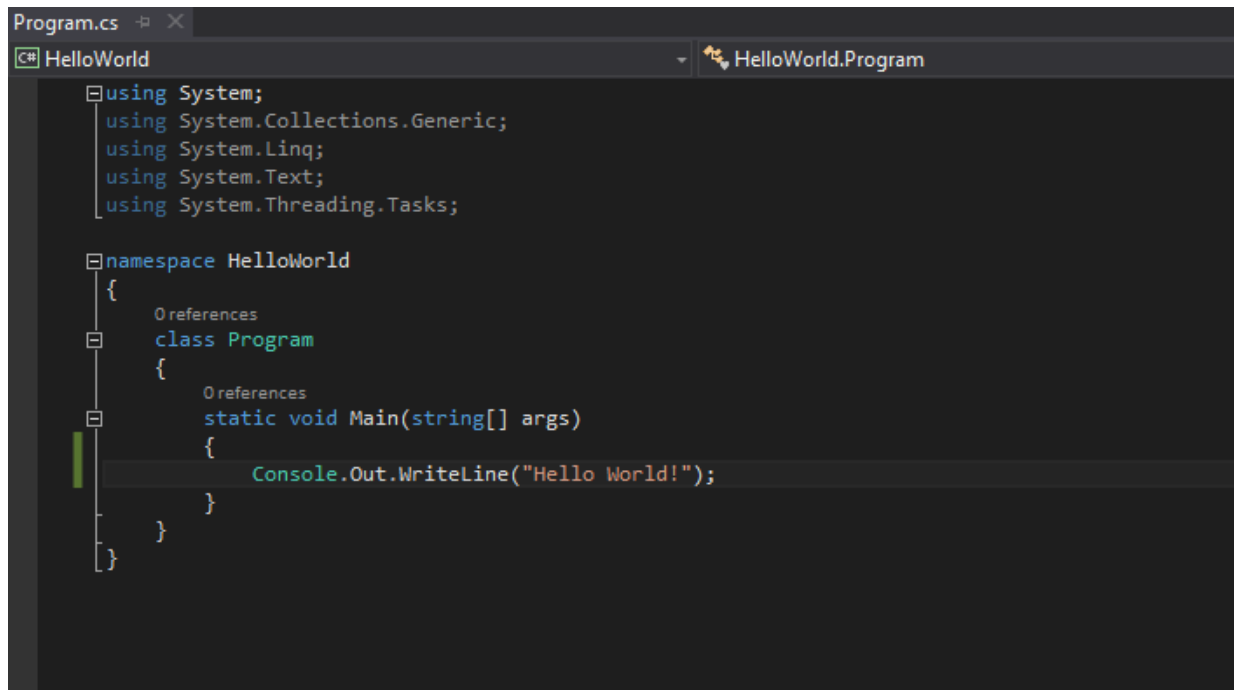
Note: When you start Visual Studio for the first time, set the environment for C# development (you will get a dialog box asking you for a preference at start-up).

1) Create a new C# console project called **HelloWorld**. Make sure your Location is writable by you.



2) Inspect the created project files and changes the name of the default cs file containing the main method to hello.cs

Note that unlike Java, the file name and contained class name need not be the same. Also notice that like java everything is contained inside a class. Run it with Ctrl-F5



```
Program.cs
C# HelloWorld HelloWorld.Program

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace HelloWorld
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

3) Note: The namespace declaration, **using System;** indicates that you are referencing the System namespace. Namespaces contain groups of code (class libraries) that can be called upon by C# programs. With the using System; declaration, you are telling your program that it can reference the classes in the System namespace without prefixing the word **System** to every reference. This is much like Java's package directive.

4) Add the code shown above to the **Hello.cs** file (note this is an image so you can't copy the code)

5) Build and execute the program

6) We will now modify the code to add some functionality (i.e. to collect the user's name and other details). Add the code below:

```

Console.Out.WriteLine("Hello World!");
String name = null;
name = Console.ReadLine();
Console.Out.WriteLine("Hello " + name);

```

7) Build and execute the program

8) Add a new class to the project called **Person** (write down at least two ways of achieving this within Visual Studio). Notice that it will put it in the project namespace.

9) Type in the following code for the Person class

```

using System;
using System.Collections.Generic;
using System.Text;
namespace HelloWorld
{
    class Person
    {
        String name;
        public String getName() {

            return name;
        }

        public void setName(String _name)
        {
            name = _name;
        }
    }
}

```

10) Use the new **Person** class to store the user's name and then output the name with the message "Hi Phil" for example (so add another method to the class).

Q. Should this class be a static class with static methods or a class we can instantiate? Can you think of a useful class method we might include in this class?

11) Add some breakpoints into your code and follow the thread of execution

12) Add some more fields to the person class.

- a) Age (int)
- b) Email (String)
- c) Address line 1 (String)
- d) Address line 2 (String)
- e) Postcode

13) Modify the HelloWorld class' main method to collect the required data from the user and display the output. For example:

Name: Ford Prefect

Age: 34

Address:

10 Park Drive

London

W1 1AA

Email: fprefect@betageuse.com

14) Display a class diagram of your application using the visual studio IDE

15) This isn't very satisfactory as we have the Main method doing work best moved elsewhere, so modify the Person class to include a new method called **printDetails()** that has the equivalent functionality.

16) Modify the **HelloWorld** class to call this new method

17) Note that C# has a special construct for getting and setting fields and looks like this for the case of the name parameter:

```
String name;  
public String Name  
{  
    get { return name; }  
    set { name = value; }  
}
```

The name value is retrieved by using:

p.Name

It can be set using:

p.Name = "Phil" for example

Modify your **Person** code to use these constructs.

End of Tutorial