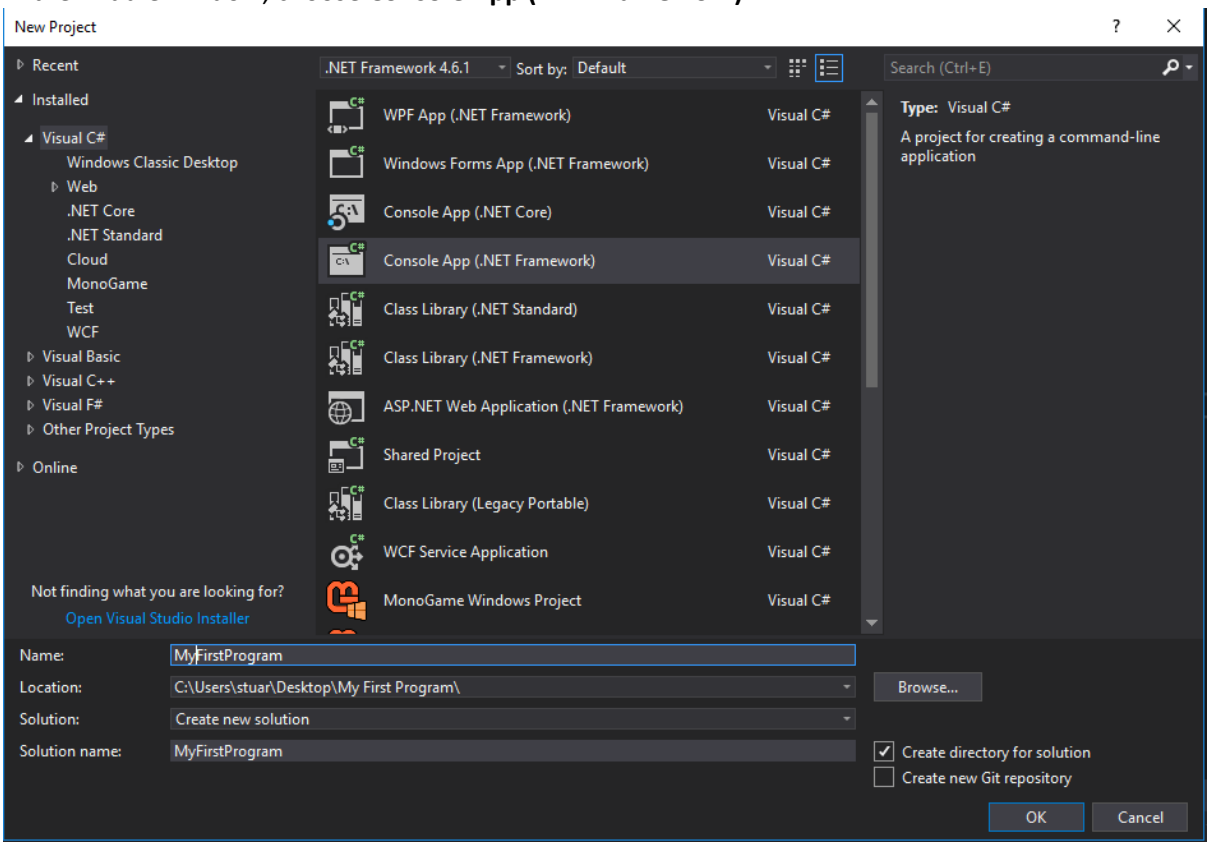
Tutorial - Introduction To Programming: Your First Console Program

In this tutorial we're going to look at how to create your first project in C# and talk about the basic structure of a file, and how to start creating programs.

Create A New Project

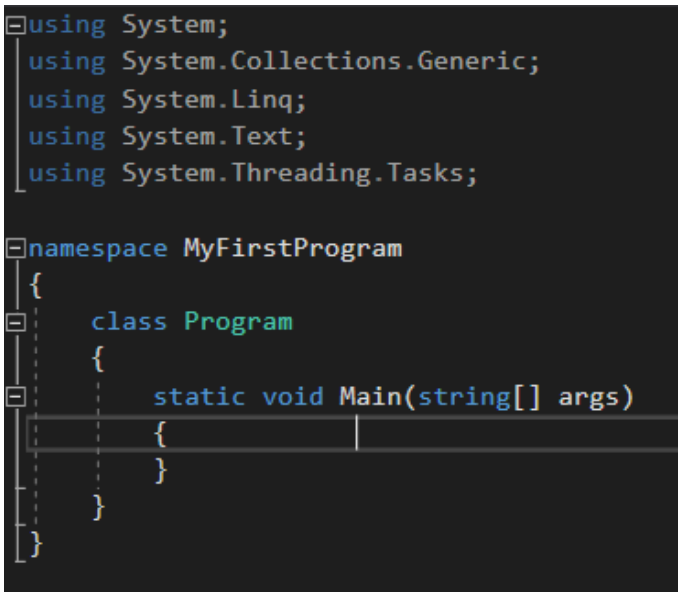
Before we talk about how to use variables, let's open Visual Studio and create a new project.

1. Open Visual Studio
2. Click **File > New > Project**
3. On the left window, select **Visual C#**
4. In the middle window, choose **Console App (.NET framework)**



5. Call the projects name **MyFirstProgram**
6. Set the **Location** to where you want to store it on your PC
7. Click **OK**

This will create a new project for you. You will now see a screen that looks something like this:



Explaining the New Screen

Code can be split up into different sections, just like this document has each section separated by a different heading.

The first code block looks like this:

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

These are called the libraries. **using** literally means that we are using a library. The words that come after **using** are the libraries that we are using. The first one is using the **System** library.

All libraries are groups of code that other people have written that we can re-use. This saves us having to write code to do common, simple tasks, because someone else wrote it for us.

Next we have this:

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

The first part is the **namespace**. You don't need to worry too much about what name spaces are now, but notice how the name space is the same name as the project you created. For now, know

that later on you can create multiple coding files that can all talk to each other, and ideally they should have the same namespace – for your projects, at least.

After the **namespace** are some curly braces **{ }**. These are how you define a section of code.

```
namespace MyFirstProgram
{
        ...
}
```

The curly brace that comes immediately after the **namespace** is what we call the **opening** brace. Everything that comes after this belongs to the **namespace MyFirstProgram**. The three dots **. . .** illustrate where the code would go that belongs to the namespace.

After that we have the **closing** curly brace. That tells us when the code block ends. Anything after the closing brace is outside the namespace.

If we want code to belong the the **MyFirstProgram** namespace, it has to be written between the opening and closing braces of the namespace.

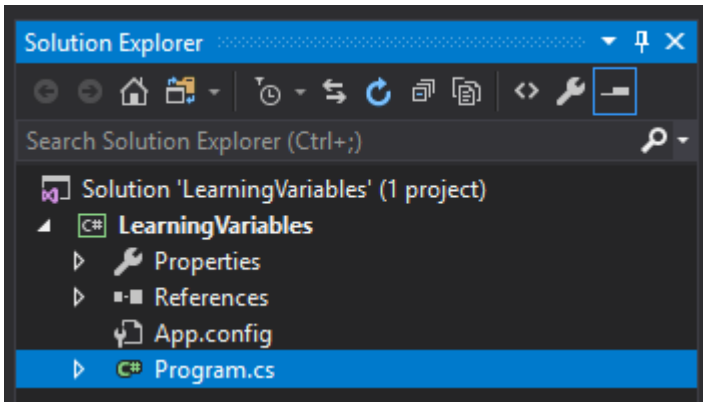
You'll get used to this stuff as you do it, so don't stress if it's sounding a bit strange so far.

The next bit of code we look at is called the **class**:

```
class Program
{
        ...
}
```

Classes are special groups of code. The simplest way for you to think about it for now is, each file that has code inside it in our program is a **class.** If you have multiple code files in your project, it means you have multiple **classes.** This isn't the strict definition, but it should help you understand the basic concept.

You can see this illustrated by looking at the **Solution Explorer** on the right window of your screen.



The file **Program.cs** is the code file that we are currently editing. It has the same name as our class. Classes always have the same name as the file.

Each class has to go inside a **namespace**. And because our class called **Program** is inside the opening and closing curly braces of the namespace **MyFirstProgram**, it belongs to that namespace.

Notice that the class has opening and closing curly braces after it. That is because we also have to specify what code belongs inside of the class.

Finally, we have our first **function**:

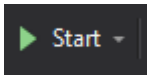
```
static void Main(string[] args)
{
}
```

This block of code is called a **function**, and the functions' name is **Main**. Again, notice how it has it's own opening and closing curly braces, and how it fits inside the curly braces of the **Program** class.

We don't need to worry about functions too much yet, but just know that classes can have many functions, and later will be making our own functions to run code for us.

Make Your Program Do Something

Now we are going to write some code. First, try and run your application. At the top of the screen, you should see a button with a green **play** symbol that says **Start**.



Press it. You should see a black window open briefly then close again. That pop-up window is known as the console, and that is where we can look at our text-based projects. To keep it open, we need to add a line of code. Modify your code as follows:

```
    class Program
    {
        static void Main(string[] args)
        {
            Console.ReadLine();
        }
    }
```

From now on, code that gets highlighted in bright green is code for you to add to your project.

Now test your project again by pressing **Start.**

You'll notice that the black screen appears but doesn't close. This is because we've paused the application, until you manually press the **Enter** key on your keyboard. That's all that line of code we added does.

Console refers to the console window we added. And **ReadLine()** will force the program to wait for us to press the **Enter** key. There are other uses for **ReadLine**, but that's all we need it to do for now.

Notice how at the end of the line of code, there is a **semi colon ;**

Whenever we write a line of code to do something, we end it with semi colons **;**

There are exceptions, such as when we are setting up our classes, because we use the **curly braces** to define the start and end of a section of code instead. But in programming, you need ways to tell the program where certain parts of your code begin and end.

The **semi colon ;** tells our program, **"Okay, this line of code is finished."**

Now add the following code:

```
class Program
{
    static void Main(string[] args)
    {
        Console.Write("Hello, world!");
        Console.ReadLine();
    }
}
```

Test your program by pressing **Start** again. You should see the black **Console** window appear with the words, **"Hello, world!"** on the screen.

The way that the word **Console** is highlighted in colour shows us that **Console** is what is called a **key word**. **Console** is part of those **using** libraries we talked about earlier. Someone else wrote code that allows us to print text to the console.

Console.Write() allows us to print text. Any words that we put between the quotation marks instead of **Hello, world!** would print to the console. Try printing something else, like this:

```
class Program
{
    static void Main(string[] args)
    {
        Console.Write("Hello, world!");
        Console.Write("This is my first program. Neat!");
        Console.ReadLine();
    }
}
```

Generally speaking, when you place quotation marks in C# code, you are defining what is called a **string**, which just means human readable text. Strings are generally used to display text to the user, or you might store strings in your code to sort data by some form of text recognition, for example, re-ordering things alphabetically based on the first letter.

This is also why Visual Studio changes the colour of code between quotation marks, to let you know that you are defining a **string**, or text. We'll talk more about strings in a later tutorial.

Conclusion

For now, try printing out some more text and possibly re-reading the tutorial to become familiar with the basic structure of a C# code file. In later tutorials, we'll start looking at more ways to build your own programs.