Tutorial - Introduction to C#

In this exercise you will create a new Visual C# project and make use of some of the techniques discussed in the lecture.

The Basics:

Exercise:

Create a new C# console project in Visual Studio by following these steps:

1. Open Visual Studio
2. From the installed templates options, select New Project -> Visual C#, Windows
3. Select "Console Application" and give the project a suitable name

Visual studio should create a **Program.cs** file for you with your project name as a **namespace**, a class called **Program** and a static function within it called **Main**. This is the entry point for the application when it is run. You can run the program but nothing will happen at the moment.

Questions:

1. What happens if you have more than one static function called Main?
2. What does the static keyword do? Why is it needed here?
3. At the top of the program are a number of lines starting with the keyword **using**. What do these do? Research these so you can understand what they are for.

We are going to start by creating a "Hello World" program because that is what our fathers and fore-fathers did, and tradition is important! We are going to output the text to the console.

Exercises:

1. Add the System.IO namespace to your program. You should be able to work out how to do this. (hint: use the **using** keyword)
2. Add these lines to the main function:

```
Console.Write("Hello World");
while (true) ; //this just keeps our console window open
```

3. Run the program. You should see "Hello World" written on the console.
4. Modify the code so it reads input from the keyboard using the Console.ReadLine() function and use this to produce a program which prompts the user for their name and writes Hello <name> back. You will need to research how string handling works in .Net and you should find tit's really simple and logical. Your program should be really simple. This is the magic of .NET.
5. You should experiemt with the program flow controls which you are familiar with from C++: *for, while, if else,* etc. They mostly work as you would expect.

Classes in C#:

Next, we are going to look at Classes.

Classes are defined in a very similar way to C++. There are no header files in C# so we do not split a class between a header and a program file. It's generally good practice to put your classes in separate files.

Exercises:

1. Create a new file called "MyClass.cs" and create a class in it called "MyClass". You can either create a .cs file manually and add the class to it or use the option to create a class which does most of the work for you.
2. In Program.cs, instantiate the class using the **new** keyword and assign a variable to it. The line to instantiate the class should look something like this:

```
MyClass myClass = new MyClass();
```

3. The syntax is very similar to C++, so it should be easy to understand. The only important difference to note is that the variable myClass would be a pointer in C++, but in C# it is a reference. Under the hood, it's still a pointer but you don't need to worry about that as a programmer. The other thing to note I that you do not need to match the **new** keyword with a **delete** keyword as you would in C++ - it is handled automatically for you.

Add a simple function to your new class:

```
public int SimpleAdditionFunction(int num1, int num2)
{
    return num1 + num2;
}
```

And call it from Program.cs:

```
int answer = myClass.SimpleAdditionFunction(3, 4);
```

Note the similarities with C++, but also note the difference in the use of the public keyword. Every class member in C# has to be proceeded by the public keyword if you want to access it from outside the class.

Exercises:

1. Experiment with the class. Add more members, constructors. Turn it into a simple calculator program that can add, subtract, multiply and divide.