Week 2 - GUI Programming

In Part 1 of this course, we are going to focus on 2 things:

- Writing a user interface in XAML
- Writing code using the Solid Principles

We will look at principle 2 later today.

Intro to GUI Programming

Within the dotnet framework there are several UI frameworks that can be used to create the user interface for your program. In this course we are going to focus on using WPF.

Why?

- Windows Forms - Not responsive / adaptive, using bitmap to draw out your interface.
- UWP - Only works on Windows 10 eco system.
- WPF (and UWP) use XAML that allows you to code your user interface and this language looks very similar to HTML.

XAML stands for eXentisble Application Markup Language.

WPF based applications can only be rendered on a Windows based machine, therefore we need to use Visual Studio (for Windows).

Event driven programming

Event driven programming is a paradigm where the events of your program are a consequence of the actions that the user takes. i.e. You cannot predict what happens next.

It is therefore important that you understand the flow of each method.

When you link an event to an object [like a click on a button] you allow for an action to do something then that may give your user an interface of what they could do next, but it does not 100% predict it.

Let's create a simple application where we can click on a button to change some text. This will also introduce you to some basic GUI programming with an event driven example.

SOLID Principle 2 - Open Closed Principle

The principle of this week is the Open Closed Principle.

This is arguably the most difficult principle to get your head around.

Watch a video about it here

The Open Closed principle stands for "open for extension, but closed for modification"

The question you would ask when applying this principle is: If you have a working program that people are using how would you change it, without running the risk that you will break what works?

So far you may have done something like this:

```
class Program
{
  private void Main(string[] args)
  {
    Console.WriteLine("What is your name?");
    string name = Console.ReadLine();
    Console.WriteLine($"Your name is {name}.");
  }
}
```

and then you may have changed it into this.

```
public class Program
{
  public void Main(string[] args)
  {
    Console.WriteLine("What is your name?");
    string name = Greeting();
    Console.WriteLine($"Your name is {name}.");
  }

  public static string Greeting()
```

```
    {
      string name = Console.ReadLine();

      return name;
    }
  }
```

Looking at these 2 code samples - what has happened?

Does this break SRP? (Yes/No) - Why?

What could you perhaps do to make this code better?

So how do we change this program so that we do not have to break it?

Let's ask the following questions:

- What is the role of the Main method?
- What is the role of the Program.cs file?
- What should the `Greeting()` method do? - how any things etc.?

To fix this read get the answer here...