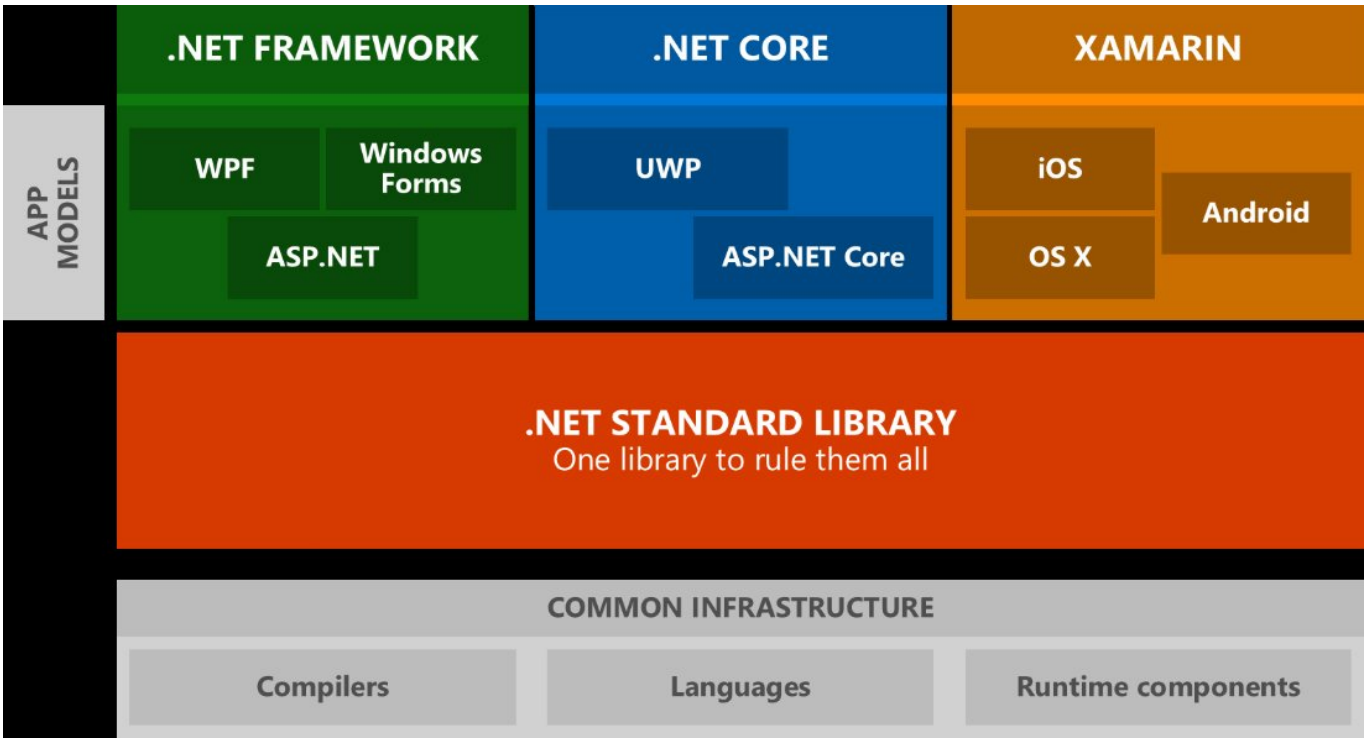
What is GUI Programming

GUI programming as the name implies is the programming of Graphical User Interfaces. GUI Objects are anything from buttons, lists, switches, labels and more.

The image below shows you all of the dotnet areas that you can develop in.



dotnet ecosystem

Framework	Description
Windows Forms and WPF	These are used in desktop applications that are compatible with all of the Windows Operating systems
UWP	Is used for applications of all sizes, all the way from the Hololens to Windows Phone (RIP, January 2019), Desktops, Laptops, the Surface Hubs and the Xbox interface.
ASP.net	Is the web framework that Microsoft has developed and uses C# in the backend.
Xamarin	Allows you to create mobile apps for Android and iOS.

Separating Code from the UI

When you are writing (good) software you need to write the code platform independent. This means that your "business logic" off the application should be able to fit in any GUI type project.

This week we are going to focus on making applications like that use this setup.

- Your classes should be UI free

- Your Methods outside of your Main Method should be UI free

The benefit of this is that you will be able to use the same code in multiple projects.

SOLID Principles

Each week in this section of the course we will look at 1 or 2 of the SOLID principles.

The principles stand for:

- Single Repsonsibility Principle
- Open Closed Principle
- Liskov Substitution Principle
- Interface Segregation Principle
- Dependency inversion principle

The idea behind using this principles is that you look at how you strucutre and organise your code so that it is easier to maintain. Although it will look like you need to do more work (and it is) for things that are relatively simple, it will help you to read and update your code better once it is in production.

The other takeaway from these principles is that you are going to go through a process of decoupling your code from the other code in your application.

SOLID Principle 1 - Single Responsibility Principle

Watch a video about it here

The Single responsibility principle can be summed like this:

Each class and method should only be responsible for a single thing. As soon as it does 2 or more things, it should be split up into multiple classes and methods respectively.

A couple of "guidelines" you can quickly see if you are breaking the rule

- Is your method longer than 10 lines?
- Has your class got more than 10 things in it?
- Do you have more than a single code indentation in your method?

Exercises

Open up the following repository and see how you can refactor these classes and methods to fit into the Single Responsibility Principle.