This course examines the Why and the When but also introduces the How.

Comparable to a carpenter who knows how to use the tools but doesn’t yet know how to build a house. He must go through an apprenticeship. This course is an apprenticeship.

**Defining Proper Classes**

Solution – a container for the entire application

Project – defines a component of the application. Each project is compiled into it’s own executable or DLL.

First step in creating a C# application is to define the component s of the C# application which will become projects. Written code should be located in it’s correct project.

Visual studio doesn’t allow technologies to be mixed in the same project.

Having a business logic component allows the separation of functionality from display. That way different UI components can request data using the same interface.

Library component – common code not directly related to any layer

Data access component – responsible for all communication with the database. Created as a class library project.

Entity framework – an object related mapper built into .NET for working with relational data. It eliminates the need for most data access code. If you’re using entity framework you may not need the data access component.

C# Application deals with things. E.G. vendors, products etc. These are known as domain entities.

A class – template or mold that provides the spec for the objects the application works with at runtime. Specifies the traits or data and behaviour or operation.

Types pf classes:

UI classes, Domain Entity classes, library classes

VM (view model) class is a UI class that handles the data interaction with a form. Search MVVM. Basically data stored at UI layer

When creating the domain layer we can create a domain model. The domain model defines the domain components and how they are related. For each domain component we might need a class to manage each type of component. These are called repository classes. They will most likely interact with the data access layer

Creating a domain model allows us to consider how the domain logic will look, what it will require and how the logic can be divided into classes.

Unit testing – tests pieces of code. Not features but individual functionality like a method. It’s automated. Sometimes known as Automted Code Testing. It’s defined with code so unit testing involves writing more code. The idea is to identify errors. Unit testing makes sure methods produce correct results from valid and invalid inputs.

Apparently … code tested with unit code has a higher level of quality. Why?

1. We have to write our methods in a way that’s testable.
2. We have to consider test cases and scenarios may come to light that weren’t accounted for.
3. Helps us find bugs or logic errors immediately.
4. May take longer to code but save time with debugging.
5. Repeatable

Unit testing is provided as part of visual studio: MSTest

Creating a unit test:

* Define the scenarios. Happy path test is a scenario that should work.
* Generate the tests
* Execute the tests

Each test method goes through 3 phases: arrange, act, assert. A class defined as a test class has [TestClass()] above it. A method has [TestMethod()]

To run a test, right click in a test and select run tests. To view the results, open the results window. Test->windows->test explorer

String.trim() – removes leading or trailing spaces.

Why is layered architecture important?

* Logical components are easier to create, maintain, extend and change.
* Code is easier to reuse