Every project starts with words. An informal spec or change request or email.

From request to classes:

1. Analyse the business problem
2. Start with nouns for class names
3. Define appropriate members.

Pillars of OOP: Abstraction, Encapsulation

From a spec we identify classes. These are the business entities that represent the problem domain. These business entities will have data and behaviour.

Abstraction – focus on the features of the entity that are relevant to the application. Simplifying reality.

Encapsulation – a technique to hide data and implementation details of a class. Hiding complexity. No other code in the app needs to know the details. Encapsulated units allow for large apps to be broken down.

Data hiding – protects the data from being corrupted. Allows processing of data before it’s presented to outside objects.

Implementation hiding – breaks down complexity, implementation knowledge not needed outside of the class, implementation can be changed without affecting the rest of the application.

Applictaion structure – most have a layered structure (project for each layer). UI(exe), Business logic(dll), Common(dll), Data(dll).

This separation allows components to be changed without affecting other layers.

Dll – a class library project in Vis Studio

Ensure a class has public access modifier. Default is no access modifier.

Internal – access to any entity in the component only.

Auto-implemented properties for simple Data members. If you don’t need data in the getter or setter.

Snippets – edit->intellisense->insert snippet. Or right click->insert snippet.

Propg = Property with getter and private setter

Automated code test is great for testing DLL code as at certain points of development it can’t be executed.

Add a folder to solution, call it tests, add a project to the folder of unit test which is an option. You can add test classes to this project. Call the class the name of the component you’re testing.

Var – defines an implicitly typed local variable. Shortens the syntax. Use only when the type is obvious.

Static – belongs to the class not an object.

As you write code immediately consider writing the test code.

Signature doesn’t include return type. Each must be unique.

Overloading – methods that have same name but different params.

Contract (class interface)– all members with public form a contract that provides these properties and methods to other classes

Ctor <tab><tab> = constructor snippet.

DateTimeOffset –type to use when the time can be set in different time zones.

Separating Responsibilities:

Separation of concerns – responsibilities divided amongst components. Minimises coupling, maximises cohesion.

Coupling – how dependent classes are on each other. Go for low coupling. This encourages better reuse and maintainability and testability. When there’s low coupling a change to a class is less likely to have an effect on other classes.

Cohesion – a measure of how related everything in a class is to the purpose of the class. Go for high cohesion. Well defined and complete making it easier to understand and test.

YAGNI – you ain’t gonna need it. Focus on what is needed now.

Repository class – a class responsible for retrieving and saving data. Handle the interaction with the data store.

Design pattern – repository pattern. Common practices for defining classes and their associated relationships.

Object equality only returns true if the same object.

Establishing Relationships:

Relationships determine how the objects interact to perform the operations of the application.