**Building Good Classes**

Covers best practices for good C# Classes.

* Signatures
* Fields (member)
* Constructors (member)
* Methods (member)
* Properties (member)

Default class access modifier is **internal** which means they are accessible from within the component they are defined within (the project).

Always best practice to add XML comments to the class. These comments are used by intellisense ///

Field – a variable in a class. Holds the data associated with the class. Set them to private to ensure no outside code can access it. Start with LC

Properties – are getter and setter functions that guard access to fields. Encapsulation. It’s best practice to define a property for each field in the class that needs to be accessed from outside of the class. Start with UC

Class Best Practices – DO:

* Meaningful name. Noun, PascalCasing.
* Add XML comments
* Use properties to encapsulate fields
* Ensure class has a well defined purpose
* 1 class per code files

AVOID:

* Abbrev
* Prefixes
* Underscores
* Prefixing with ‘C’ used to be recommended, no longer!
* Large classes. Maybe doing too much

Propfull <tab> <tab> creates a field and property automatically.

To create a Unit Test right click inside a method and select the option.

Var keyword uses implicit typing.

Assert.AreEqual(….,….)

NamesPace – Namespace provides a unique address for the class and organises the division of code. Logic hierarchy. Provides explicit references for larger applications with repeat names for classes. Default namespace can be changed in the properties of a project. Only new classes will be set by this default, not existing.

Namespace DO:

* Follow: <company><technology><feature>
* Use Pascal casing

AVOID:

* Using system. System is reserved for .NET framework.
* Avoid using the same name as a class in the namespace.

Static classes – a cohesive set of services provided by a set of methods often without any internal data. A static class can only have static members. We can’t instantiate a static class.

A static class provides a container for utility features. E.G. Logging or email generation. Most likely to find them in the Common code.

A static class cannot implement an interface.

Static Class DO:

* Use sparingly as Supporting classes.
* Use in common code library components.

AVOID:

* Using as a bucket of AD HOC methods. Each class should have a purpose.

Singleton Class – a class that provides only one instance. How to define? Change constructor accessibility modifier to private. This allows the code in the class to create an instance but no other code can create an instance. Add a static property that creates and provides the one instance of the class. The code in the static property checks if the instance has been created. If not, it creates it and returns the instance. Check Design Patterns on Plural site. Or Design Patterns on Ramp.

Advantages:

* A singleton has an instance and can be passed to other code as needed. A singleton can have child objects.
* Supports OOP features. Can implement an interface and can be inherited from.

FAQ’s

1. Whats the diff between property and method?

Properties are the gate keepers providing access to the data. Methods are the operations.

1. What is a constructor?

A method called to create an instance of a class. Usually to initialise data.

1. What is the purpose of a namespace?

Namespaces provide logical division of code. Prevent class name collision.

1. What is a static class?

A class that can’t be instantiated. Usually for utility features and exists in common librarty.

1. What is a Singleton?

A class with one instance. Supports all OOP features.