Interfaces provide a clean way for your application to interact with classes, components, applications and systems. Used in many design patterns.

Whats an interface?

Common .NET interfaces

How to define your own.

Interface: point of contact between 2 systems.

API – Application Programming Interface.

Class interface – public members of a class are the implicit class interface.

Other parts of the application can only communicate via the interface.

The interface is ONLY the definition, not the implementation.

In C# there is the explicit interface which is a type you create by using the **interface** keyword. It is reusable for any class that wants to use the interface.

Think of an explicit interface as a role that an object can play. When the object needs to use a role it must implement the interface.

.NET provides several built in explicit interfaces recognisable by the I prefix. E.G.

IDisposable

IEquatable

IEnumerable

Thinking of them as roles, you can decide if you want to implement the Interface.

IEquatable is an interface used to define equality params. Implementing this interface allows you to write your own Equals() operation.

Example of a logging interface:

Classes that implement the logging interface are responsible for what info is logged. Then a class in the Common section will write the information.

A LoggingService class is passed a list of objects that can log data. The class is static so an instance doesn’t have to be instantiated. The list given can be of type ILog. This guarantees the object given will implement the ILog interface and will have the .Log() method.

Adding method signatures to an interface allows different object types to be passed as the same type and exposes the appropriate interface for the use of the object in that context.

Members of an interface don’t need access modifiers, they’re automatically public.

To add an interface to a class, add it to the class signature then implement every member.

Each member must be public, non-static and have exactly the same signature.

Interface-based polymorphism – many shapes. Allows to to pass different objects in the same list as long as they implement the same interface.