OOP power lies in its reuse.

Extract similar functionality.

Building reusable classes/components.

Reduces the amount of new code you have to write.

Build it once. Test it once. Update it once. Enhance it once.

Techniques for reuse:

Collaboration. Sharing common classes.

Inheritance. Extract common code into a base class.

Components.

Every class created inherits from Object class. .NET defines this relationship implicitly.

The existing ToString() method will return the full class name of the object that called it.

Reuse with inheritance is implemented by the properties and methods of any parent class being available to the child class.

override *methodToBeOverriden() –* the auto generator creates a ref to the parent class (base. *methodToBeOverriden()).*

Polymorphism (many shaped) – the concept that a same named method can behave differently depending on the type of object that calls it. A method with a given name may result in different shapes. It allows us to use classes in a uniform way.

Inheritance based polymorphism – overriding methods.

Standard Entity properties:

IsNew

HasChanges

IsValid

EntityState – if deleted, it’s not really deleted merely its status is changed

When creating a base class you can make irt abstract or concrete. Abstract can’t be instantiated.

Abstract is only intended as a base. Can’t use the new keyword with it.

Sealed? – sealed class is a class that can’t be extended through inheritance. No other class can inherit. Prevents extension and customisation.

By default members of a base class are not overrideable. Mark the base class method with either virtual or abstract.

Virtual has its own definition but allows overriding.

Abstract states it MUST be overridden. It has no statement body.