Lets look at OO initialisation, references and using cases.

Access members by first creating an instance unless its static. In that case we use the class name.

Objects represent things. Defines one thing created from the class template. Created at runtime with new keyword.

Classes represent types of things. Defines the template describing that type. Created at Dev.

Methods of Object Initialisation (initialising data members):

* Setting properties. New keyword to create an object and then set all properties with public access. Best when populating from database values so data conversion is easy. Debugger can point us to the exact property if there’s a problem. Can also be used to modify values anytime after initialisation.
* Parameterised Constructor. Data member values are passed in the constructor. Good when setting a basic set of properties. Initialise to a valid state.
* Object Initialisers. Uses new keyword but uses curly braces. At runtime it executes default constructor and then sets the values of the properties inside the curly braces. Basically a short cut to setting properties technique. Best used when readability is important. When initialising a subset or superset of properties.

Objects depend on other objects sometimes.

If an object is only needed in one method of a class, instantiate that object in only the method.

If always, create as a member. Instantiate in constructor.

If sometimes use lazy loading. Put the object instantiation in the getter of the class. First check for null. Create object if null. Need a property for object.

Propfull <tab> <tab> generates property fields.

Null Checking:

Classic is to use if statement. E.G.

If(object == null)

Can be tedious and hard to read. C# 6 has a better way: Null Conditional operators (?.) removes the need for explicit null checks. It works as follows:

Var companyname = currentProduct?.ProductVendor?.Name;

If null then null if not then dot. If its null then the expression is null.

FAQ’s:

1. What is the diff between an object and a class?

An object is an instance of a type, a class is a template definition for a type.

1. What is lazy loading and when would you use it?

Lazy loading is creating an instance of an object as a member within a class only when it is needed. Use it when the object is only used sometimes within the class.