**Extension Methods**

**Purpose:**

An extension method allows for a class – even a sealed class! – to have another method added to it

**Implementation:**

To create an Extension method the method to be extended must meet the below requirements:

* Class method resides must be static
* Class members must be public
* Method must be static (prerequisite for members of static classes)
* Import namespace with class
* First parameter is the type to extend – cannot be group or non-CLS structure (i.e. Synergy structure) Note CLS structure is a .NET value type structure

Signature:

Accessibility **static extension** method name ,return\_type

Example:

public static extension method MyMethod ,int

arg1 ,string

endparams

Creates an extension method named MyMethod that extends the string type (first parameter) and returns an integer

proc

data str ,string

data int ,int

int = str. MyMethod ()

**NOTES**

Manual - Page 296:

The EXTENSION modifier can only be specified in conjunction with the STATIC modifier.

It enables you to extend a class declared in another assembly, even if the class

is declared as SEALED. An extension method must be declared in a non-generic, static

class, and the first parameter type should be of the type you want to extend. The

first parameter type cannot be a group or non-CLS structure.

To use an extension method, you’ll need to import the namespace that contains the

class that contains the extension method and then call it using an instance of the

first parameter of the extension method. (See Examples on page 4-128.) The class

members you are using must be public. If an extension method has the same

signature (minus the first parameter of the extension method) as an existing method

in the class, the compiler resolves to the existing class method.

Murach’s “C# 2008” book on extension methods:

Unlike inheritance you don’t have to create a new derived type to add functionality

MSDN:

Extension methods enable you to "add" methods to existing types without creating a new derived type, recompiling, or otherwise modifying the original type. Extension methods are a special kind of static method, but they are called as if they were instance methods on the extended type. For client code written in C# and Visual Basic, there is no apparent difference between calling an extension method and the methods that are actually defined in a type.

The most common extension methods are the LINQ standard query operators that add query functionality to the existing [System.Collections.IEnumerable](http://msdn.microsoft.com/en-us/library/system.collections.ienumerable.aspx) and [System.Collections.Generic.IEnumerable<T>](http://msdn.microsoft.com/en-us/library/9eekhta0.aspx) types. To use the standard query operators, first bring them into scope with a using System.Linq directive. Then any type that implements [IEnumerable<T>](http://msdn.microsoft.com/en-us/library/9eekhta0.aspx) appears to have instance methods such as [GroupBy](http://msdn.microsoft.com/en-us/library/system.linq.enumerable.groupby.aspx), [OrderBy](http://msdn.microsoft.com/en-us/library/system.linq.enumerable.orderby.aspx), [Average](http://msdn.microsoft.com/en-us/library/system.linq.enumerable.average.aspx), and so on. You can see these additional methods in IntelliSense statement completion when you type "dot" after an instance of an [IEnumerable<T>](http://msdn.microsoft.com/en-us/library/9eekhta0.aspx) type such as [List<T>](http://msdn.microsoft.com/en-us/library/6sh2ey19.aspx) or [Array](http://msdn.microsoft.com/en-us/library/system.array.aspx).