Partial

Indicates that a type declaration is a partial definition of the type.

You can divide the definition of a type among several declarations by using the `Partial` keyword. You can use as many partial declarations as you want, in as many different source files as you want. However, all the declarations must be in the same assembly and the same namespace.

Syntax

[<attrlist>] [accessmodifier] [Shadows] [MustInherit | NotInheritable] _

Partial { Class | Structure | Interface | Module } name [(Of typelist)]

    [Inherits classname]

    [Implements interfacenames]

    [variabledeclarations]

    [proceduredeclarations]

{ End Class | End Structure }

Parts

Term	Definition
`attrlist`	Optional. List of attributes that apply to this type. You must enclose the Attribute List in angle brackets (`< >`).
`accessmodifier`	Optional. Specifies what code can access this type.
`Shadows`	Optional.
`MustInherit`	Optional.

Term	Definition
`NotInheritable`	Optional.
`name`	Required. Name of this type. Must match the name defined in all other partial declarations of the same type.
`Of`	Optional. Specifies that this is a generic type.
`typelist`	Required if you use Of.
`Inherits`	Optional.
`classname`	Required if you use `Inherits`. The name of the class or interface from which this class derives.
`Implements`	Optional.
`interfacenames`	Required if you use `Implements`. The names of the interfaces this type implements.
`variabledeclarations`	Optional. Statements which declare additional variables and events for the type.
`proceduredeclarations`	Optional. Statements which declare and define additional procedures for the type.
`End Class` or `End Structure`	Ends this partial `Class` or `Structure` definition.

Example

The following example splits the definition of class `sampleClass` into two declarations, each of which defines a different `Sub` procedure.

```
Partial Public Class sampleClass

    Public Sub sub1()

    End Sub

End Class

Partial Public Class sampleClass

    Public Sub sub2()

    End Sub

End Class
```

The two partial definitions in the preceding example could be in the same source file or in two different source files.

Advantages

The biggest use of partial classes is that using Partial Classes multiple programmers can work on the same class easily. Partial classes are mainly used by code generator because the code can be added to the class without having to recreate the source file. Moreover, it's easier to categorize code with partial classes.

Consider the following points while implementing the partial classes:

The partial keyword indicates that other parts of the class can be defined in the same namespace. All the parts should use the partial keyword. All the parts should be available at compile time to form the final type. All the parts should have the same accessibility, such as public, private, and so on.

If you inherit a class or interface on a partial class, then it should be inherited on all parts of a partial class. If a part of the partial class is sealed, then the entire class will be sealed. Also if a part of the partial class is abstract, then the entire class will be an abstract class.