Nesting a Class

A class can be created inside of another class. A class created inside of another is referred to as nested. To nest a class, simply create it as you would any other. Here is an example of a class called Inside that is nested in a class called Outside:

```
Public Class Outside
    Public Class Inside

    End Class
End Class
```

In the same way, you can nest as many classes as you wish in another class and you can nest as many classes inside of other nested classes if you judge it necessary. Just as you would manage any other class so can you exercise control on a nested class. For example, you can declare all necessary fields, properties, or methods in the nested class or in the nesting class. When you create one class inside of another, there is no special programmatic relationship between both classes: just because a class is nested does not mean that the nested class has immediate access to the members of the nesting class. They are two different classes and they can be used separately as you judge it necessary.

The name of a nested class is not "visible" outside of the nesting class. To access a nested class outside of the nesting class, you must qualify the name of the nested class anywhere you want to use it. For example, if you want to declare an Inside variable somewhere in the program but outside of Outside, you must qualify its name. Here is an example:

```
Public Module Exercise
    Public Class Outside
        Public Class Inside
            Public Sub New()
                MsgBox(" =- Inside -=")
            End Sub
        End Class

        Public Sub New()
            MsgBox(" -= Outside =-")
        End Sub
    End Class

    Public Function Main() As Integer
        Dim Out As Outside = New Outside

        Dim Ins As Outside.Inside = New Outside.Inside

        Return 0
    End Function
End Module
```

This would produce:

Because there is no programmatically privileged relationship between a nested class and its "container" class, if you want to access the nested class in the nesting class, you can use its static members. In other words, if you want, you can declare static all members of the nested class that you want to access in the nesting class. Here is an example:

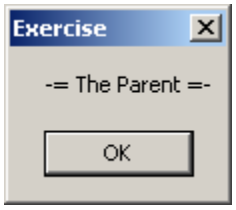
```
Public Module Exercise
    Public Class Outside
        Public Class Inside
            Public Shared InMessage As String

            Public Sub New()
                MsgBox("=- Insider -=")
                InMessage = "Sitting inside while it's raining"
            End Sub

            Public Shared Sub Show()
                MsgBox("Show me the wonderful world of VB.NET Programming")
            End Sub
        End Class

        Public Sub New()
            MsgBox("-= Outside =-")
        End Sub

        Public Sub Display()
            msgbox(Inside.InMessage)
            Inside.Show()
        End Sub
    End Class

    Public Function Main() As Integer
        Dim Recto As Outside = New Outside
        Dim Ins As Outside.Inside = New Outside.Inside

        Recto.Display()

        Return 0
    End Function
End Module
```

In the same way, if you want to access the nesting class in the nested class, you can go through the static members of the nesting class. To do this, you can declare static all members of the nesting class that you want to access in the nested class. Here is an example:

```
Public Module Exercise
    Public Class Outside
        Public Class Inside
            Public Shared InMessage As String

            Public Sub New()
                MsgBox("=- Insider -=")
                InMessage = "Sitting inside while it's raining"
            End Sub

            Public Shared Sub Show()
                MsgBox("Show me the wonderful world of VB.NET Programming")
            End Sub
```

```
            Public Sub FieldFromOutside()
                msgbox(Outside.OutMessage)
            End Sub
        End Class

        Private Shared OutMessage As String

        Public Sub New()
            MsgBox(" -= The Parent =-")
            OutMessage = "Standing outside! It's cold and raining!!"
        End Sub

        Public Sub Display()
            MsgBox(Inside.InMessage)
            Inside.Show()
        End Sub
    End Class

    Public Function Main() As Integer
        Dim Recto As Outside = New Outside
        Dim Ins As Outside.Inside = New Outside.Inside

        Recto.Display()

        Ins.FieldFromOutside()
        Return 0
    End Function
End Module
```

This would produce:



Instead of static members, if you want to access members of a nested class in the nesting class, you can first declare a variable of the nested class in the nesting class. In the same way, if you want to access members of a nesting class in the nested class, you can first declare a variable of the nesting class in the nested class. Here is an example:

```
Public Module Exercise
    Public Class Outside
        REM A member of the nesting class
        Private OutMessage As String

        REM The nested class
        Public Class Inside
            REM A field in the nested class
            Public InMessage As String

            REM A constructor of the nested class
            Public Sub New()
                MsgBox("=- Insider -=")
                Me.InMessage = "Sitting inside while it's raining"
```

```
            End Sub

            REM A method of the nested class
            Public Sub Show()
                REM Declare a variable to access the nesting class
                Dim Outsider As Outside = New Outside
                MsgBox(outsider.OutMessage)
            End Sub
        End Class REM End of the nested class

        REM A constructor of the nesting class
        Public Sub New()
            Me.OutMessage = "Standing outside! It's cold and raining!!"

            MsgBox("-= The Parent =-")
        End Sub

        REM A method of the nesting class
        Public Sub Display()
            MsgBox(insider.InMessage)
        End Sub

        REM Declare a variable to access the nested class
        Dim insider As Inside = New Inside
    End Class

    Public Function Main() As Integer
        Dim Recto As Outside = New Outside
        Dim Ins As Outside.Inside = New Outside.Inside

        Ins.Show()
        Recto.Display()
        Return 0
    End Function
End Module
```

This would produce: