VB: Inheritance and Interfaces

Simple as PIE

Overview

- Inheritance
- Protected Members
- Shared Members
- Abstract Classes and Members
- Virtual Members
- Constructors
- Interfaces

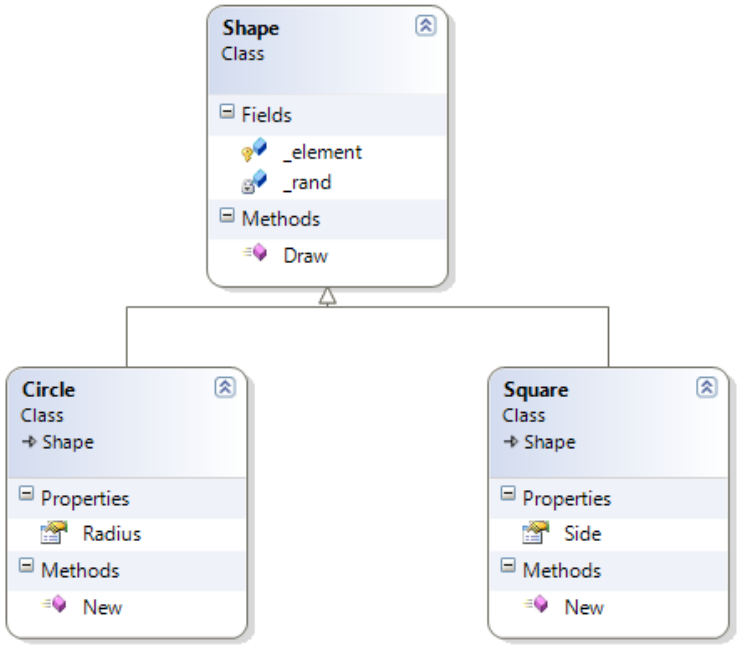
Inheritance

- **Create classes to extend other classes**
  - Classes inherit from System.Object by default
  - Gain all the state and behavior of the base class

```
Class Shape
    Public Sub Draw()
End Class

Class Square
    Inherits Shape
End Class

Class Circle
    Inherits Shape
End Class
```



Access Modifiers – Part II

- **Review of access modifiers – with inheritance in mind**
  - Public: member can be used internally and externally
  - Private: member can be used internally in the declaring type
  - Protected: member can only be used internally in the declaring type and its derived types
- **Examples:**
  - A private member declared in Shape could only be used inside Shape
  - A protected member in Shape could be used inside Shape or its derived types – Square and Circle

Shared Members

- Shared members are members of the type
- Public shared members can be used without creating an instance
- The value of shared fields are shared across all instances of the type
- All members of a Module are shared
  - Cannot instantiate a module

```
Public Shared Property Diameter As Double

Public Shared Function Circumference()
    Return Diameter * Math.Pi
End Function
```

Abstract Classes and Members

- **The MustInherit keyword**
  - Apply to classes
- **The MustOverride keyword**
  - Apply to members (methods, properties, indexers, events)
- **Abstract class cannot be instantiated**
  - Abstract class is designed as a base class
  - Must implement abstract members to make a concrete class

```
Public MustInherit Class Shape
    Public MustOverride Function Area() As Integer
End Class
```

```
Public Class Square : Inherits Shape
    Public Overrides Function Area() As Integer
    End Function
End Class
```

Virtual Members

- The Overridable keyword creates a virtual member
- Use Overrides keyword to override the member in derived class
- Use MyBase to call implementation from base class

```
Protected Overridable Sub SetColors(shape As System.Windows.Shapes.Shape)
    shape.Fill = New SolidColorBrush(Colors.Green)
    shape.Stroke = New SolidColorBrush(Colors.Black)
End Sub
```

```
Protected Overrides Sub SetColors(shape As System.Windows.Shapes.Shape)
    MyBase.SetColors(shape)
    shape.Fill = New SolidColorBrush(Colors.Red)
End Sub
```

Constructors in Derived Types

- Construction is done "inside out"
- Constructors in derived types call base class constructor
  - Passing required parameters

```
Private _canvas As Canvas

Public Sub New(canvas As Canvas)
    _canvas = canvas
End Sub
```

```
Public Sub New(canvas As Canvas, side As Integer)
    MyBase.New(canvas)
    Dim rect As New Rectangle()
    rect.Width = side
    rect.Height = side
    SetColors(rect)
    _element = rect
End Sub
```

Interfaces

- **An interface defines a group of related methods and properties**
  - Classes can implement interfaces
    - They must implement the complete set of methods and properties
    - Classes can implement more that one interface
- **Interfaces are about defining something an object can do**
  - While types are about defining what an object is

```
Interface IXmlExport
    Function GetXml() As String
End Interface
```

```
Class Shape
    Implements IXmlExport

    Public Function GetXml() As String _
        Implements IXmlExport.GetXml

        ' ...
    End Function
End Class
```

```
Class Employee
    Implements IXmlExport

    Public Function GetXml() As String _
        Implements IXmlExport.GetXml

        ' ...
    End Function
End Class
```

Summary

- Inheritance
- Protected Members
- Shared Members
- Abstract Classes and Members
- Virtual Members
- Constructors
- Interfaces