

Module 9: Developing Components in Visual Basic .NET

Overview

- Components Overview
- Creating Serviced Components
- Creating Component Classes
- Creating Windows Forms Controls
- Creating Web Forms User Controls
- Threading

◆ Components Overview

- Types of Components
- Using Modules As Components
- Using Classes As Components
- Using Components in Unmanaged Client Applications
- .NET Remoting Overview

Types of Components

- Structures
- Modules
- Classes
- Component classes
- Serviced components
- User controls
 - Windows Forms user controls
 - Web Forms user controls

Using Modules As Components

- Declare the module as public
- Reference and import the assembly into client code

```
Public Module MyMathFunctions
    Public Function Square(ByVal lng As Integer) As Long
        Return (lng * lng)
    End Function
    ...
End Module

'Client code
Imports MyAssembly
...
Dim x As Long = Square(20)
```

Using Classes As Components

- Declare the class as public
- Reference and import the assembly into client code

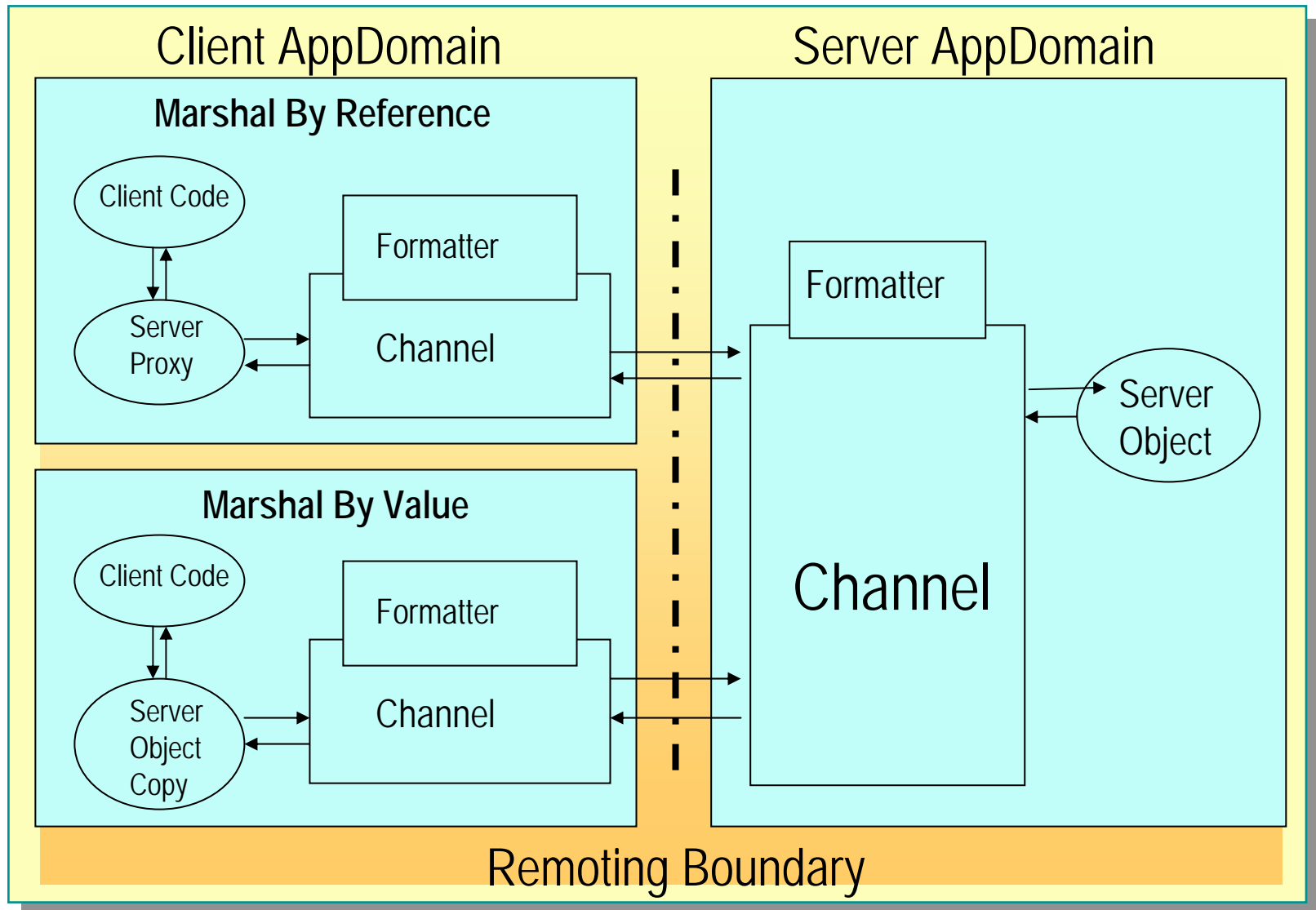
```
Public Class Account
    Public Sub Debit(ByVal AccountId As Long, Amount As Double)
        'Perform debit action
    End Sub
    Public Sub Credit(ByVal AccountId As Long, Amount As Double)
        'Perform credit action
    End Sub
End Class
```

```
'Client code
Imports MyAssembly
Dim x As New Account( )
x.Debit(1021, 1000)
```

Using Components in Unmanaged Client Applications

- **Setting assembly properties**
 - Generate a strong name
 - Select **Register for COM Interop** in **Build** options
- **Exposing class members to COM and Component Services**
 - Define and implement interfaces
 - Use the **ClassInterface** attribute with **AutoDual** value
 - Use the **COMClass** attribute

.NET Remoting Overview



◆ Creating Serviced Components

- Hosting Components in Component Services
- Using Transactions
- Using Object Pooling
- Using Constructor Strings
- Using Security
- Using Other Component Services
- Configuring Assemblies for Component Services

Hosting Components in Component Services

- Add a reference to `System.EnterpriseServices` in your assembly
- The `System.EnterpriseServices` namespace provides:
 - `ContextUtil` class
 - `ServicedComponent` class
 - Assembly, class, and method attributes

Using Transactions

- Transaction attribute specifies how a class participates in transactions
- ContextUtil class provides transaction voting
- AutoComplete attribute avoids using the SetAbort, SetComplete, and ContextUtil methods

```
<Transaction(TransactionOption.Required)> Public Class Account
  Inherits ServicedComponent
  Public Sub Debit(...)
    'Perform debit action
    ContextUtil.SetComplete( )
  End Sub
  <AutoComplete( )> Public Sub Credit(...)
    'Perform credit action
    'No SetComplete because AutoComplete is on
  End Sub
End Class
```

Using Object Pooling

- Object pooling allows objects to be created in advance
- ObjectPooling attribute specifies MinPoolSize and MaxPoolSize
- ServicedComponent provides CanBePooled method

```
<ObjectPooling(Enabled:=True, MinPoolSize:=5, _  
                MaxPoolSize:=50)> _  
Public Class Account  
    Inherits ServicedComponent  
    ...  
    Protected Overrides Function CanBePooled( ) As Boolean  
        Return True  
    End Function  
End Class
```

Using Constructor Strings

- Specify the `ConstructionEnabled` attribute to indicate that a construction string is required
- Override the `Construct` method to retrieve information

```
<ConstructionEnabled(True)>Public Class Account
    Inherits ServicedComponent
    Protected Overrides Sub Construct(ByVal s As String)
        'Called after class constructor
        'Use passed in string
    End Sub
End Class
```

Using Security

- Security configuration attributes enable security and role configuration
- SecurityCallContext class provides role checking and caller information

```
<ComponentAccessControl(True), SecurityRole("Manager")> _  
Public Class Account  
    Inherits ServicedComponent  
    Public Function GetDetails( ) As String  
        With SecurityCallContext.CurrentCall  
            If .IsCallerInRole("Manager") Then  
                Return .OriginalCaller.AccountName  
            End If  
        End With  
    End Function  
End Class
```

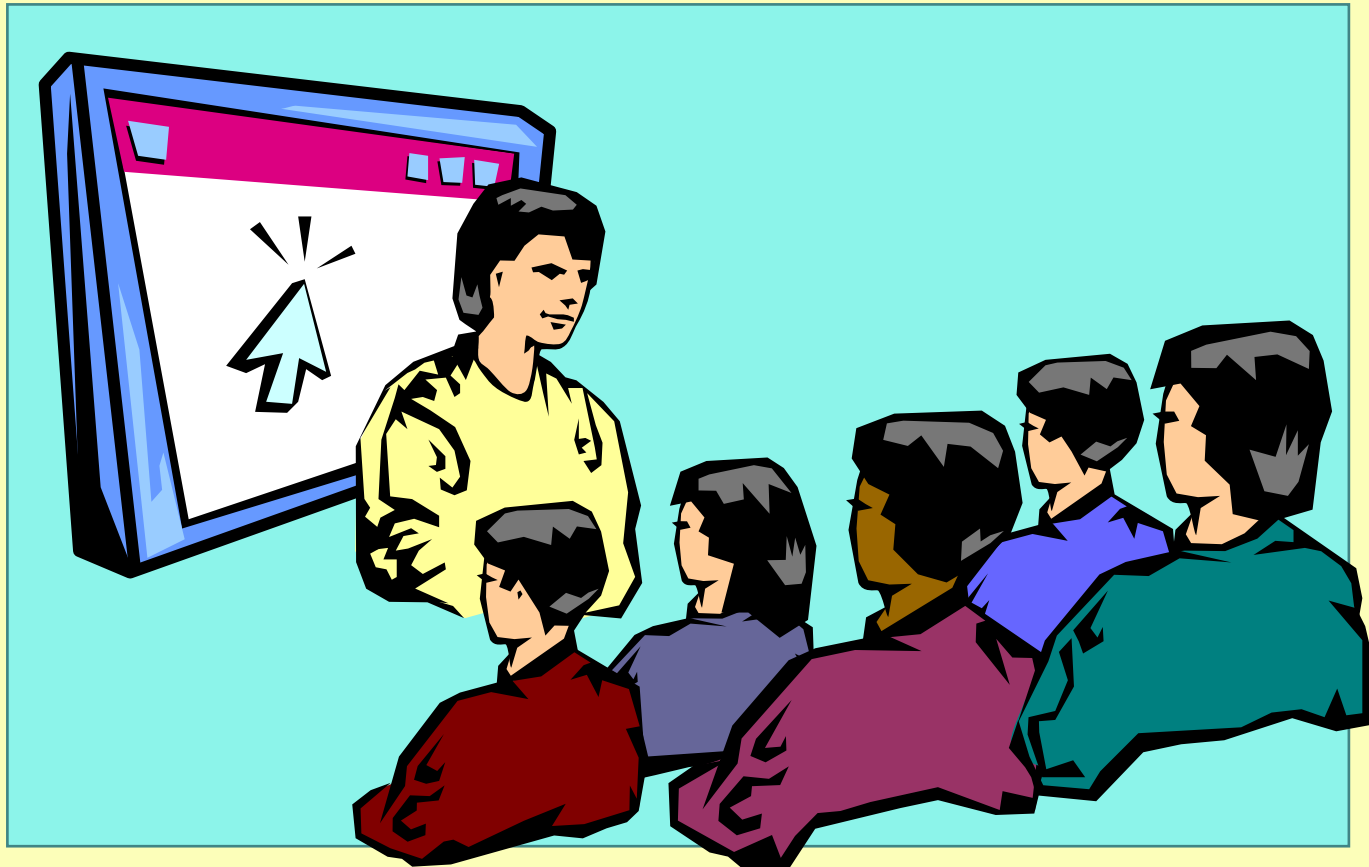
Using Other Component Services

- **Other Component Services include:**
 - Just-in-time activation
 - Queued components
 - Shared properties
 - Synchronization

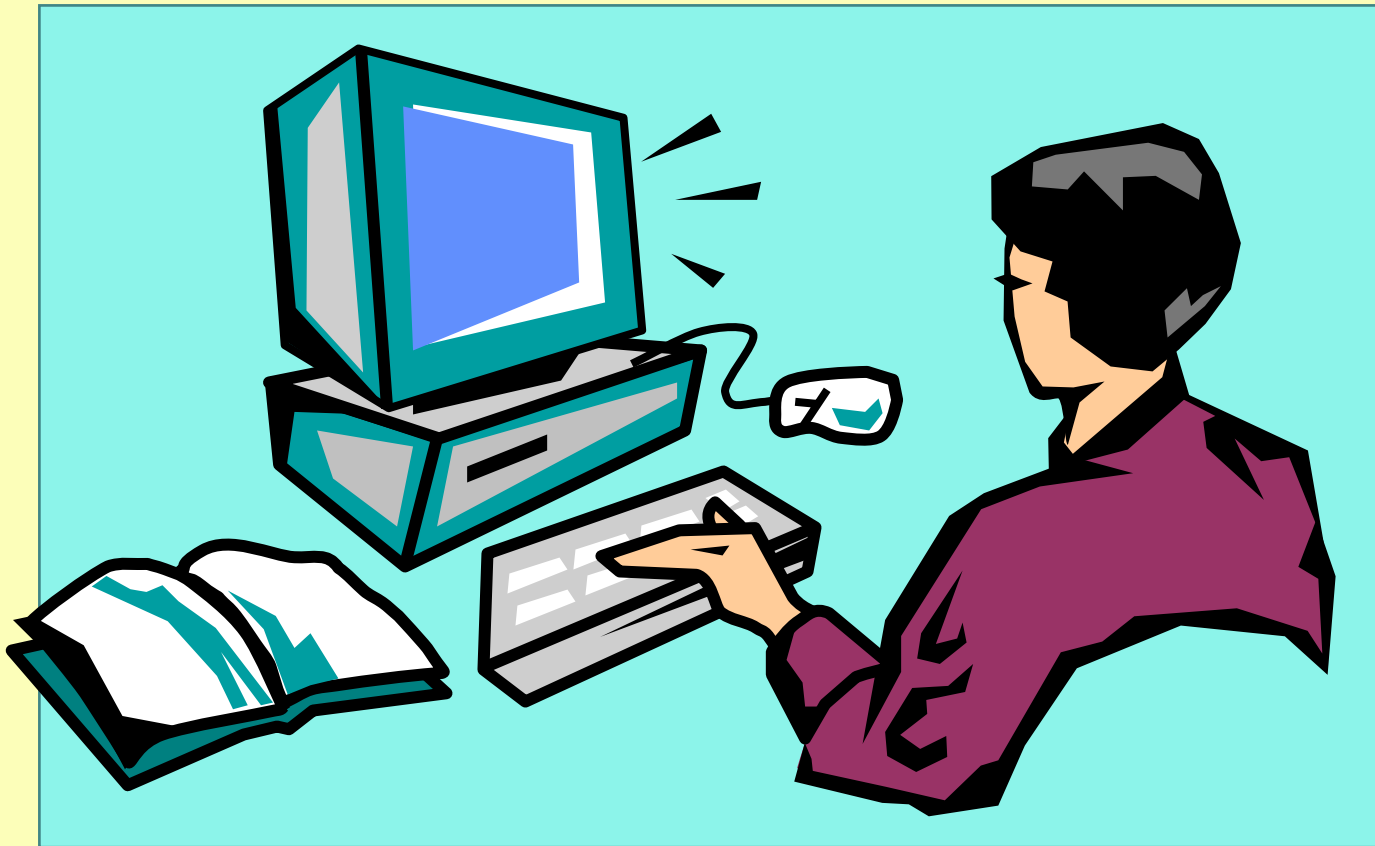
Configuring Assemblies for Component Services

- Setting assembly attributes
 - ApplicationName
 - Description
 - ApplicationActivation: library or server application
 - AssemblyKeyFile
- Using Regsvcs to register and create Component Services applications
 - Regsvcs.exe myApplication.dll
- Using Lazy Registration
 - Application registered on first use by client

Demonstration: Creating a Serviced Component



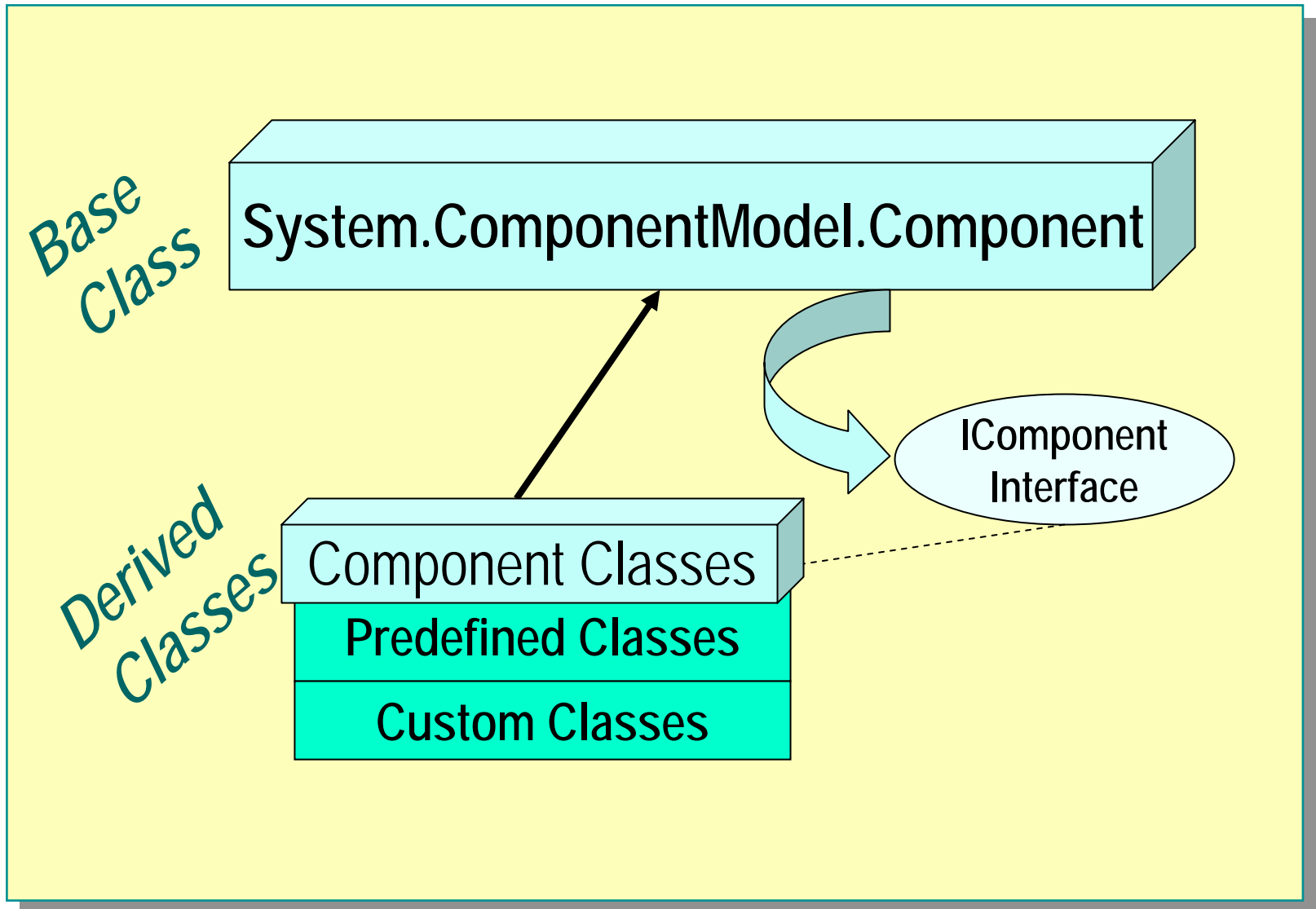
Lab 9.1: Creating a Serviced Component



◆ Creating Component Classes

- Architecture of a Component Class
- Creating a Component Class

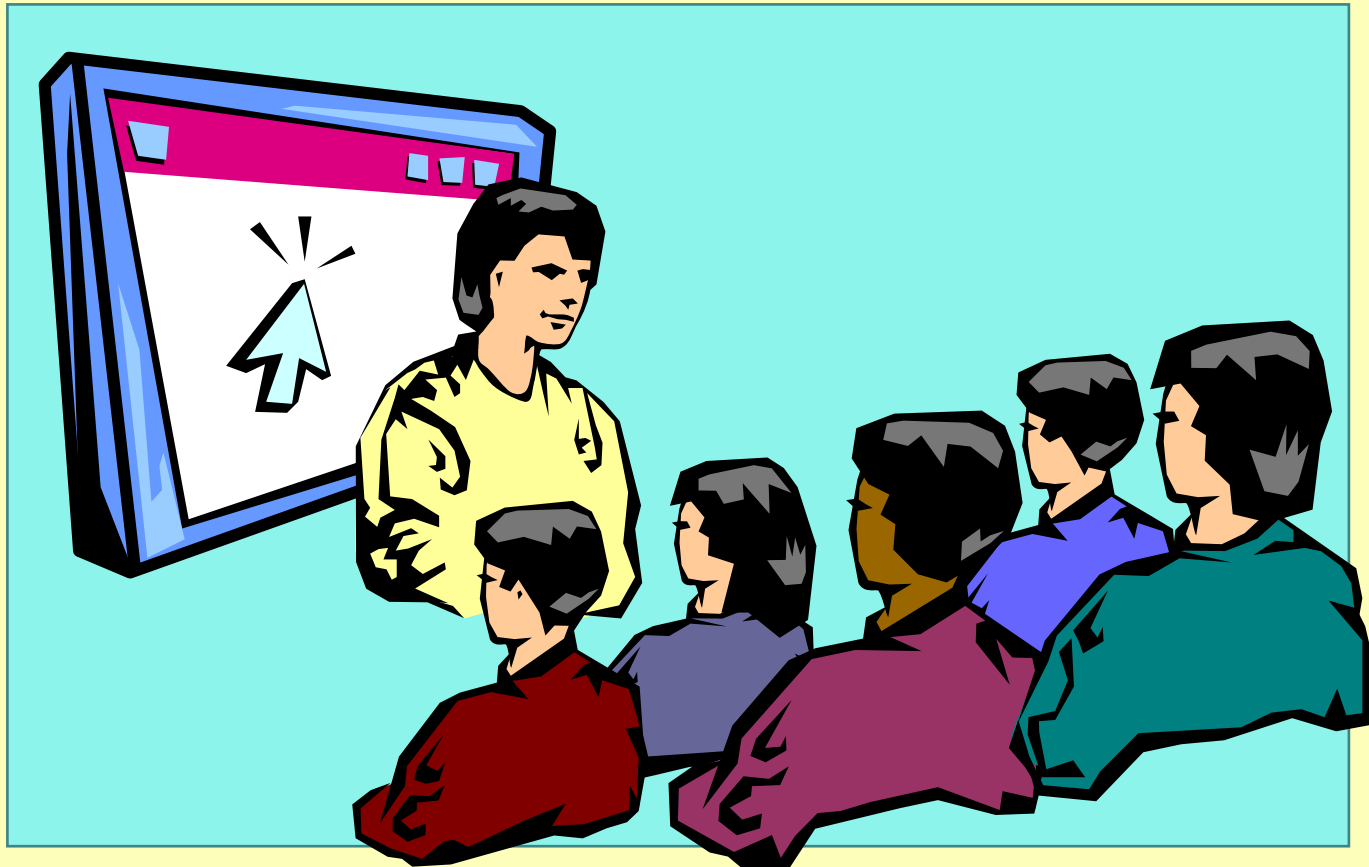
Architecture of a Component Class



Creating a Component Class

1. Inherit the `System.ComponentModel.Component`
 - Perform any initialization in constructor
 - Override **Dispose** method
2. Add any sited components
 - Use Server Explorer or Toolbox items
3. Create required functionality
 - Properties, methods, and events
4. Build the assembly

Demonstration: Creating a Stopwatch Component



◆ Creating Windows Forms Controls

- Inheriting from the UserControl Class
- Inheriting from a Windows Forms Control
- Providing Control Attributes

Inheriting from the UserControl Class

- Inherit from `System.Windows.Forms.UserControl`
- Add required controls to designer
- Add properties and methods that correspond to those of constituent controls
- Add any additional properties and methods
- No `InitProperties`, `ReadProperties`, or `WriteProperties`
 - Property storage is automatic

Inheriting from a Windows Forms Control

- Allows enhanced version of a single control
- Inherit from any System.Windows.Forms control

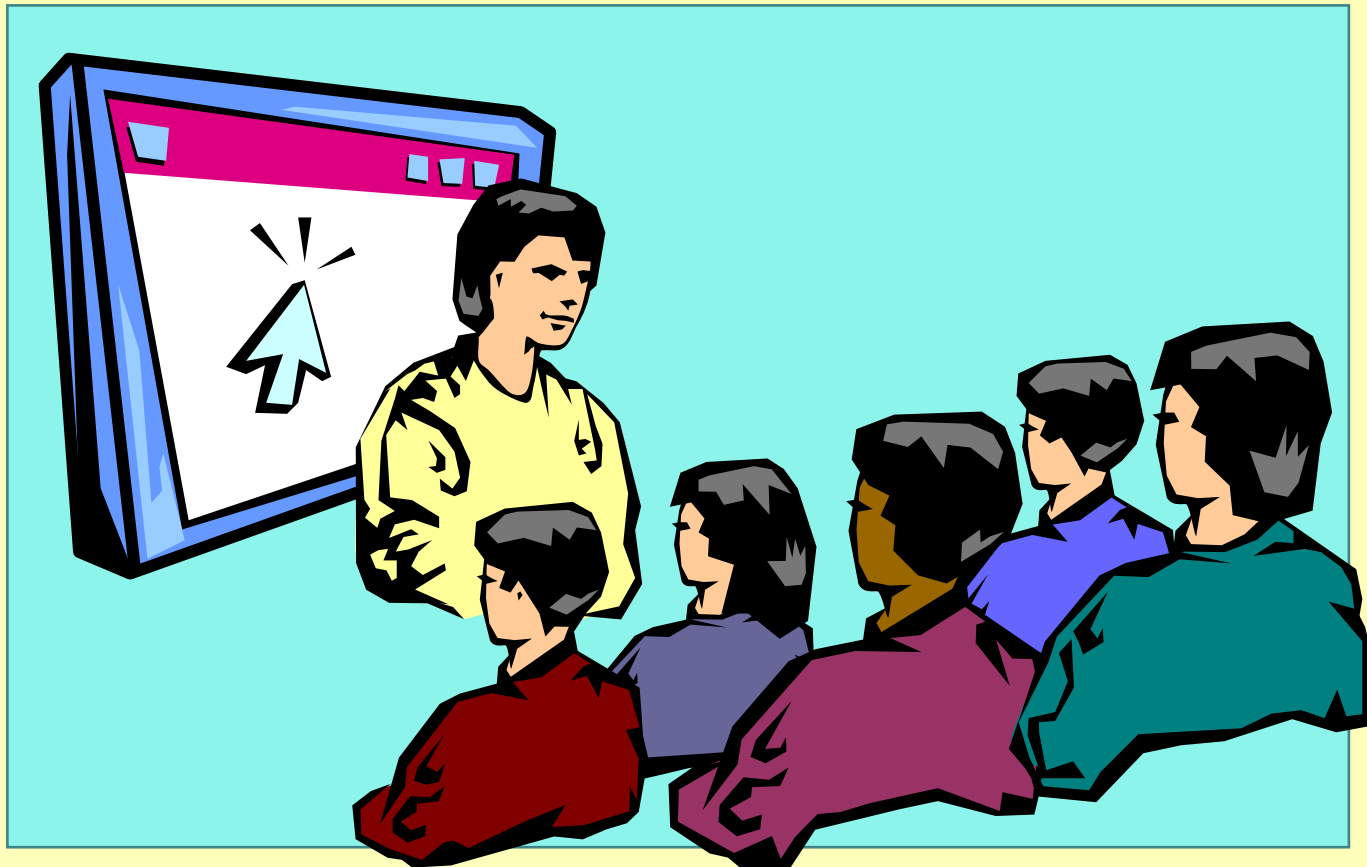
```
Public Class MyTextBox
    Inherits System.Windows.Forms.TextBox
    Private strData As String
    Public Property HiddenData( ) As String
        Get
            Return strData
        End Get
        Set(ByVal Value As String)
            strData = Value
        End Set
    End Property
    ...
End Class
```

Providing Control Attributes

- System.ComponentModel provides control attributes
- Class level – DefaultProperty, DefaultEvent, ToolboxBitmap
- Property level – Category, Description, DefaultValue

```
Imports System.ComponentModel
<ToolboxBitmap("C:\txticon.ico"), DefaultEvent("Click")> _
Public Class MyTextBox
    Inherits System.Windows.Forms.UserControl
    <Category("Appearance"), _
        Description("Stores extra data"), _
        DefaultValue("Empty")> _
    Public Property HiddenData( ) As String
        ...
    End Property
    ...
End Class
```

Demonstration: Creating an Enhanced TextBox



◆ Creating Web Forms User Controls

- Extending Existing Controls
- Creating Web User Controls

Extending Existing Controls

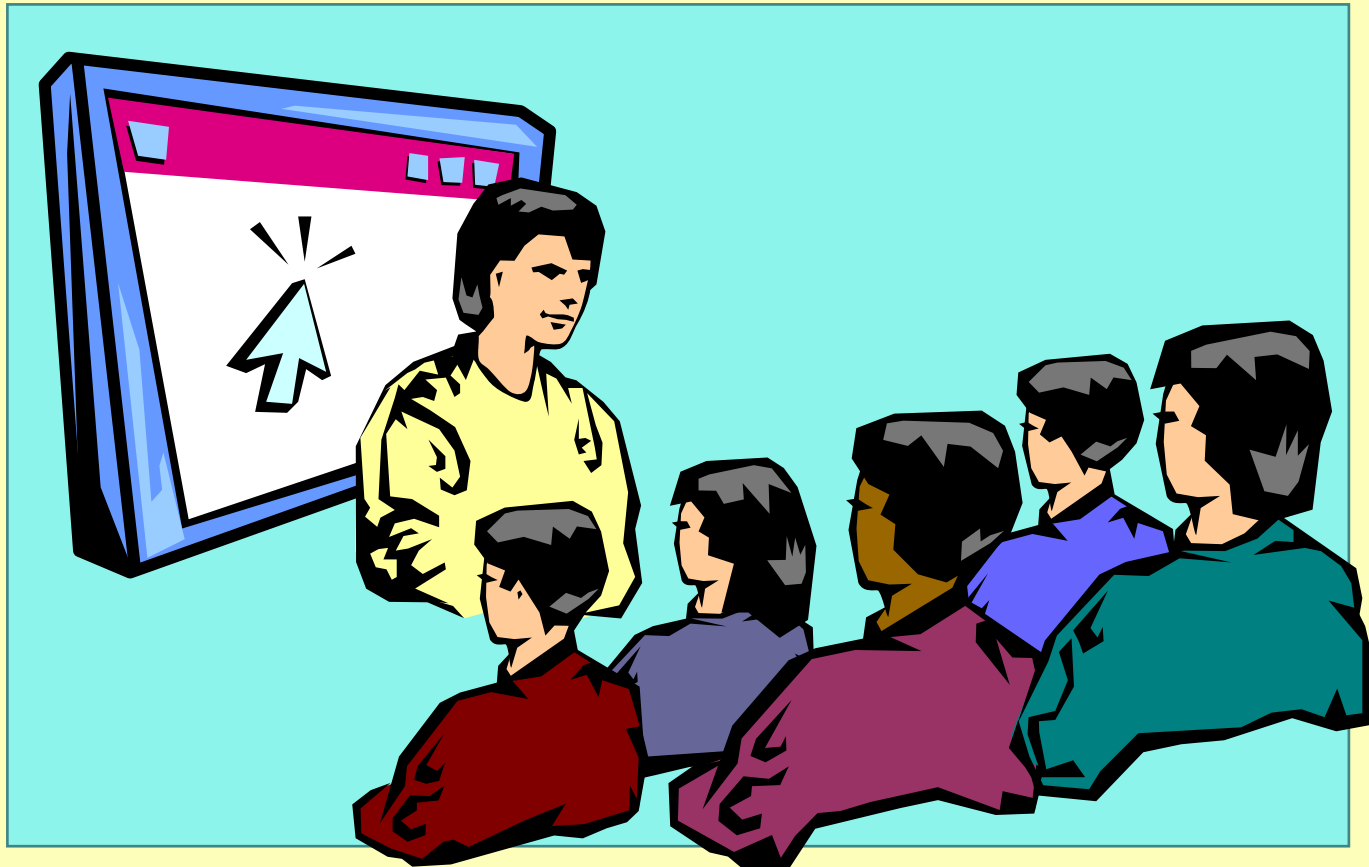
1. Add a Web user control to an ASP.NET Web project
2. Use the Toolbox to drag existing controls to the Web user control designer
3. Add properties and methods
4. Save the .ascx file
5. Drag the .ascx file from Solution Explorer to the Web Forms Designer
6. Create Web Form code as usual

Creating Web User Controls

```
<%@ Control Language="vb" AutoEventWireup="false"  
  Codebehind="SimpleControl.ascx.vb"  
  Inherits="MyApp.SimpleControl"%>  
<asp:TextBox id="TextBox1" runat="server"></asp:TextBox>
```

```
Public MustInherit Class SimpleControl  
  Inherits System.Web.UI.UserControl  
  Protected WithEvents TextBox1 As System.Web.UI.WebControls.TextBox  
  Public Property TextValue( ) As String  
    Get  
      Return TextBox1.Text  
    End Get  
    Set(ByVal Value As String)  
      TextBox1.Text = Value  
    End Set  
  End Property  
End Class
```

Demonstration: Creating a Simple Web Forms User Control



Lab 9.2: Creating a Web Forms User Control

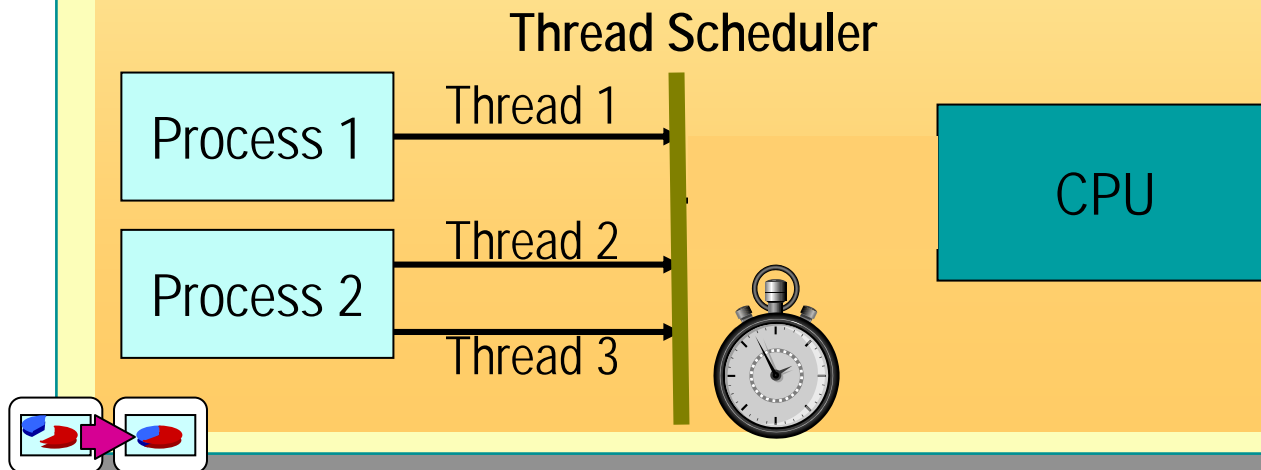


◆ Threading

- What Is a Thread?
- Advantages of Multithreading
- Creating Threads
- Using Threading
- When to Use Threading

What Is a Thread?

- The unit of execution that the CPU processes
 - All application processes contain at least one thread
- Threads are scheduled
 - The computer appears to perform multiple tasks at one time
 - Every thread contains its own call stack and storage



Advantages of Multithreading

- Improved user interface responsiveness
 - Example: a status bar
- No blocking
- Asynchronous communication
- No thread affinity
 - Objects are not tied to one thread

Creating Threads

- Use the `System.Threading.Thread` class
 - Constructor specifies delegate method
 - Methods provide control of thread processing
 - Properties provide state and priority information
- Use a class if parameters are required
 - Allow public access to class variables
 - Raise an event when finished

Using Threading

```
Class Calculate
  Public iValue As Integer
  Public Event Complete(ByVal Result As Integer)
  Public Sub LongCalculation( )
    'Perform a long calculation based on iValue
    ...
    RaiseEvent Complete(iResult)      'Raise event to signal finish
  End Sub
End Class
```

```
Sub Test( )
  Dim calc As New Calculate( )
  Dim th As New Threading.Thread(AddressOf calc.LongCalculation)
  calc.iValue = 10
  AddHandler calc.Complete, AddressOf CalcResult
  th.Start( )
End Sub

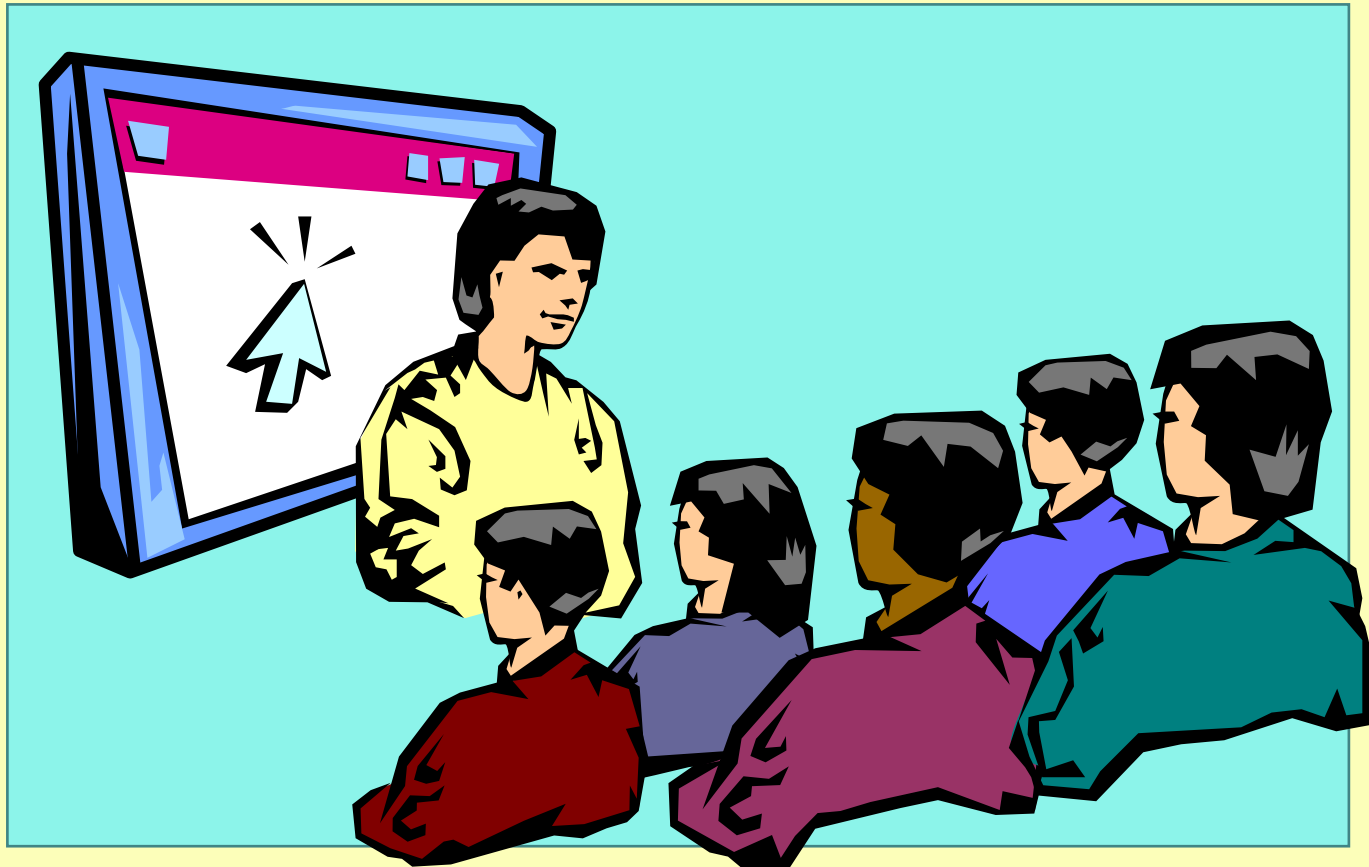
Sub CalcResult(ByVal Result As Integer)
  ...
End Sub
```

When to Use Threading

- Use threads carefully
 - Using more threads requires more system resources
- Synchronize access to shared resources
 - Prevent two threads from accessing shared data simultaneously
 - Use **SyncLock** statement to block sections of code

```
Sub Worker( )  
    SyncLock(theData)           'Lock this object variable  
        theData.id = iValue  
        'Perform some lengthy action  
        iValue = theData.id  
    End SyncLock                'Unlock the object variable  
End Sub
```

Demonstration: Using the SyncLock Statement



Review

- Components Overview
- Creating Serviced Components
- Creating Component Classes
- Creating Windows Forms Controls
- Creating Web Forms User Controls
- Threading

Course Evaluation

