

**Module 1:  
Introducing Windows  
Forms**

# Overview

- Creating a Form
- Adding Controls to a Form
- Creating an Inherited Form
- Organizing Controls on a Form
- Creating MDI Applications

# Lesson: Creating a Form

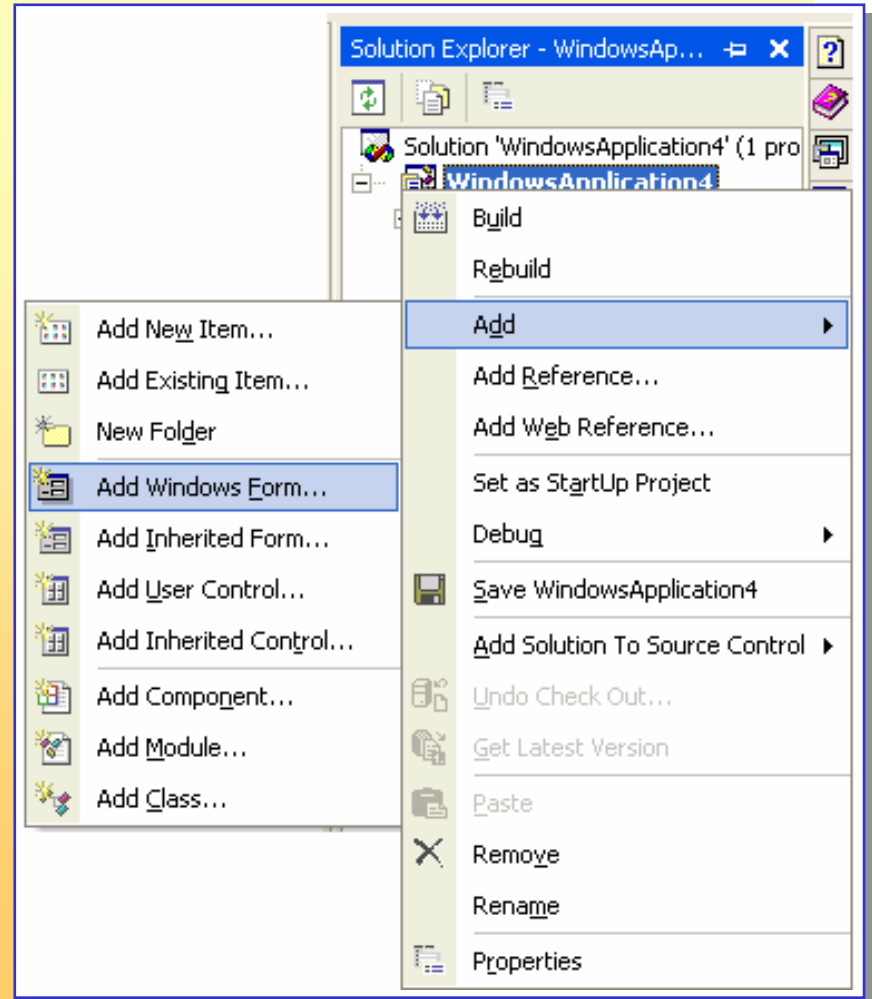
- Windows Forms vs. Web Forms
- How to Create a Form
- How to Set Form Properties
- Form Life Cycle
- How to Handle Form Events
- Windows Form Designer-Generated Code

# Windows Forms vs. Web Forms

Feature	Windows Forms	Web Forms
Deployment	Can be run without altering the registry	No download required
Graphics	Includes GDI+	Interactive or dynamic graphics require round trips to the server for updates
Responsiveness	Provide the quickest response speed for interactive applications	Can take advantage of the browser's dynamic HTML to create rich UI
Platform	Requires .NET Framework running on the client computer	Require only a browser
Programming model	Based on a client-side, Win32-based message-pump mode	Applications components are invoked via HTTP
Security	Code-based and role-based security	Role-based security

# How to Create a Form

- A base form is created when you create a new project
- To create a new form
  1. Right-click the project in Solution Explorer
  2. Click **Add**
  3. Click **Add Windows Forms**



# How to Set Form Properties

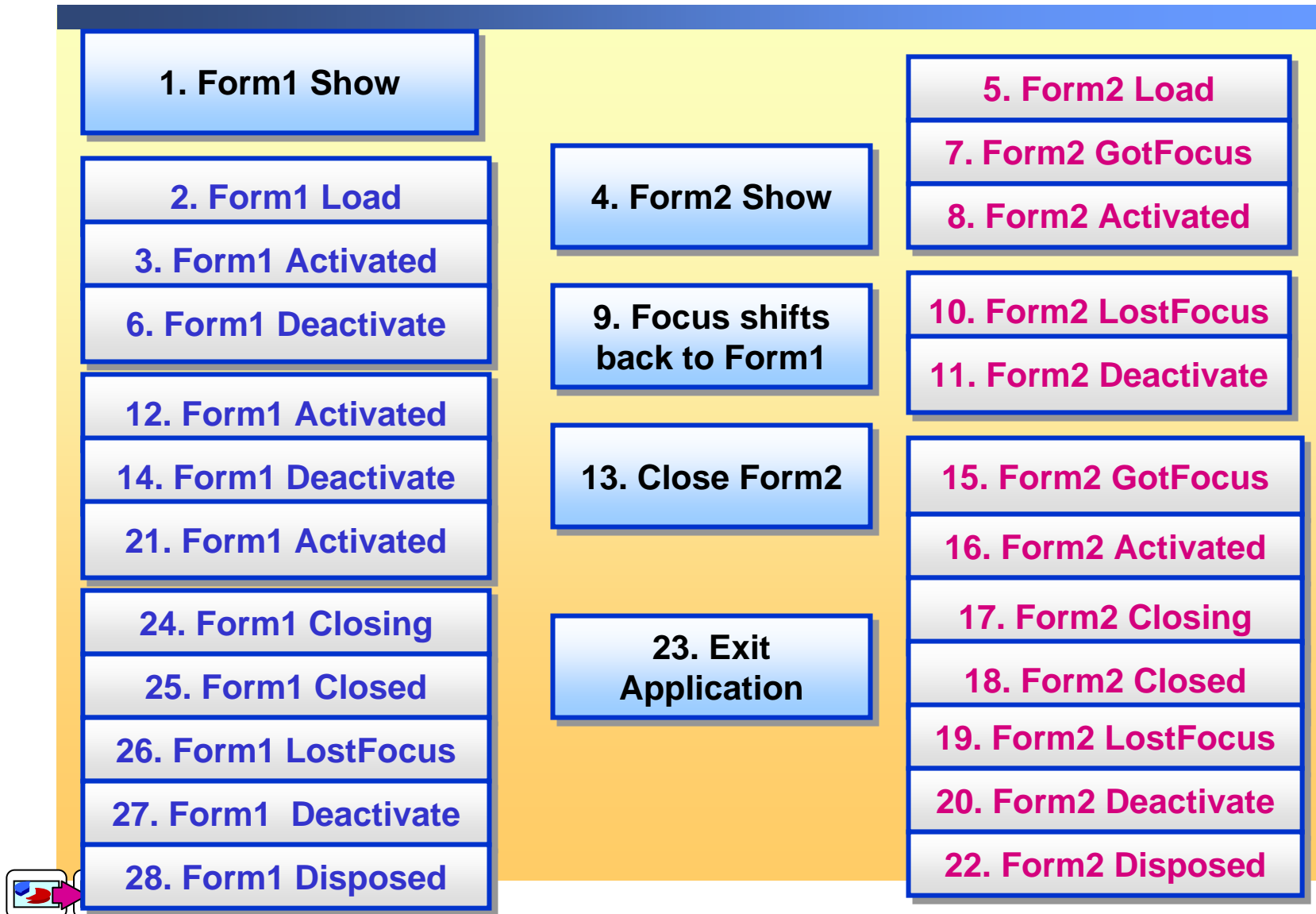
The image shows a screenshot of the Visual Studio Properties window for a form named 'Form1'. The window is titled 'Properties' and shows the following structure:

- Form Name:** A callout box labeled 'Form Name' points to the 'Form1' text in the top-left corner of the Properties window.
- Categorized Button:** A callout box labeled 'Categorized Button' points to the 'A-Z' icon in the toolbar of the Properties window.
- Alphabetic Button:** A callout box labeled 'Alphabetic Button' points to the 'AcceptButton' property in the list of properties.
- Description Pane:** A callout box labeled 'Description Pane' points to the text description of the 'AcceptButton' property at the bottom of the window.

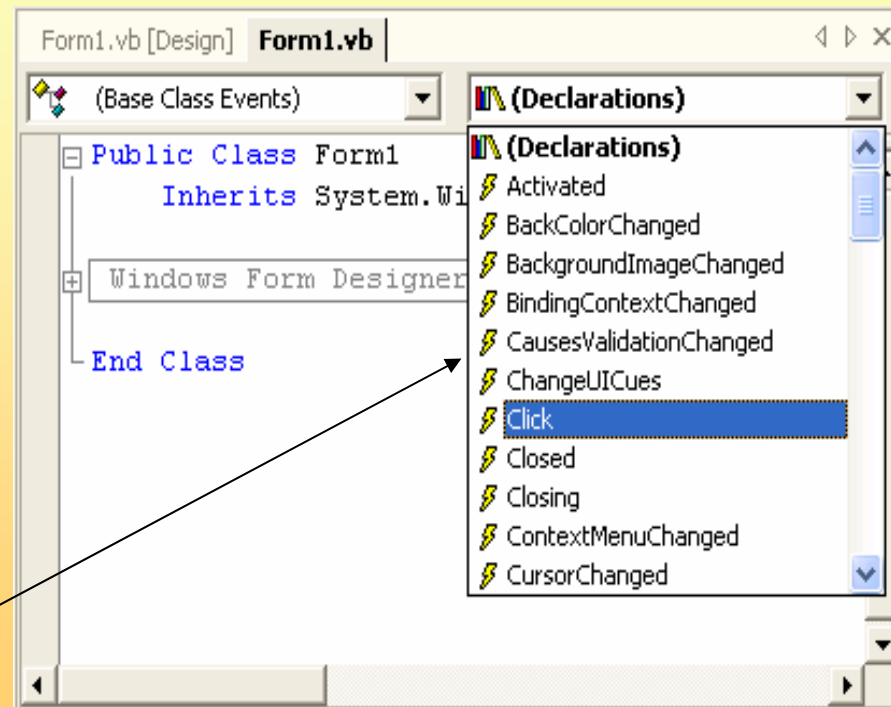
Property	Value
(DataBindings)	
(DynamicProperti	
(Name)	<b>Form1</b>
<b>AcceptButton</b>	(none)
AccessibleDescrip	
AccessibleName	
AccessibleRole	Default
AllowDrop	False

**AcceptButton**  
The accept button of the form. If this is set, the button is 'clicked' whenever the user presses the 'ENTER' key.

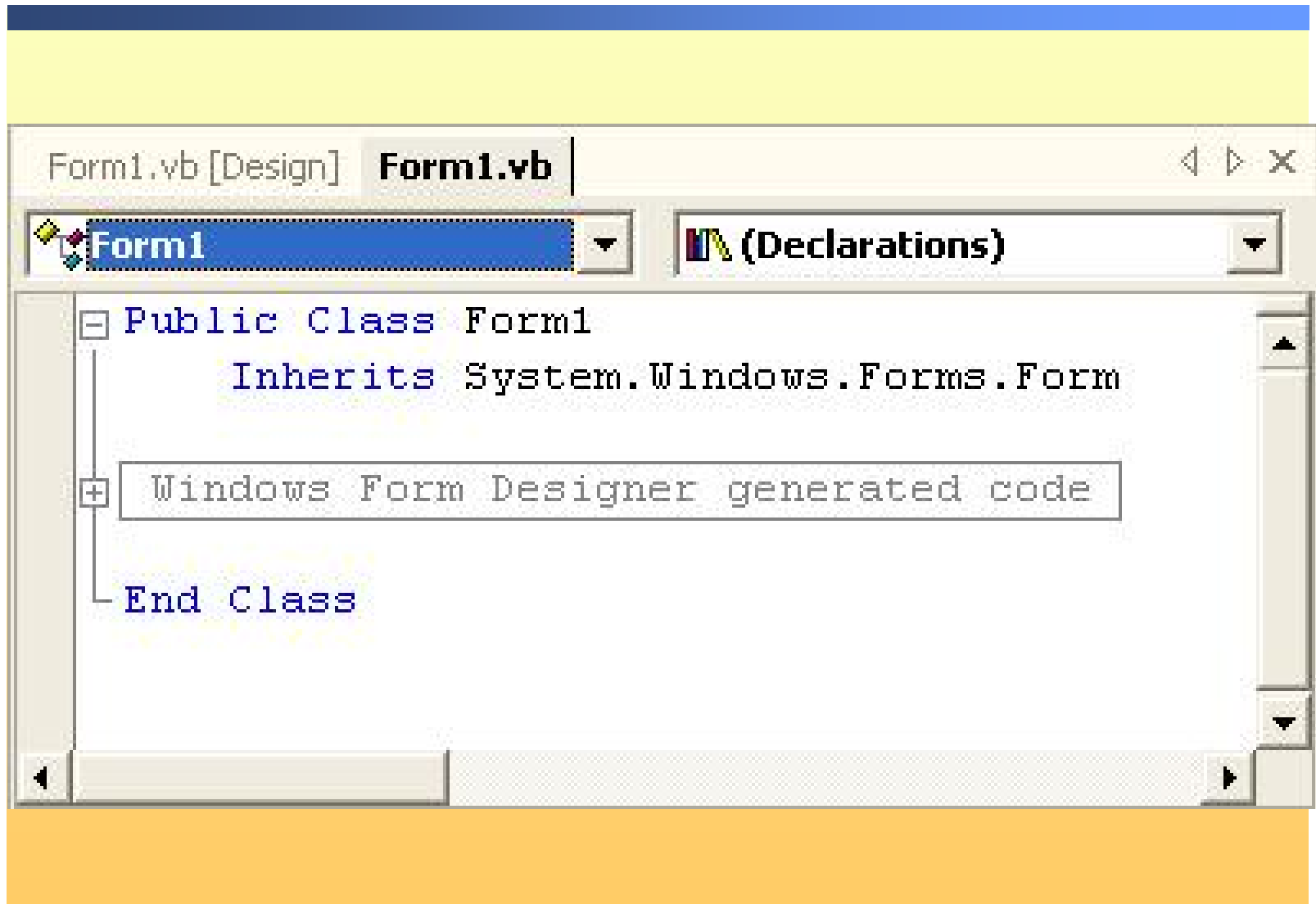
# Form Life Cycle



# How to Handle Form Events



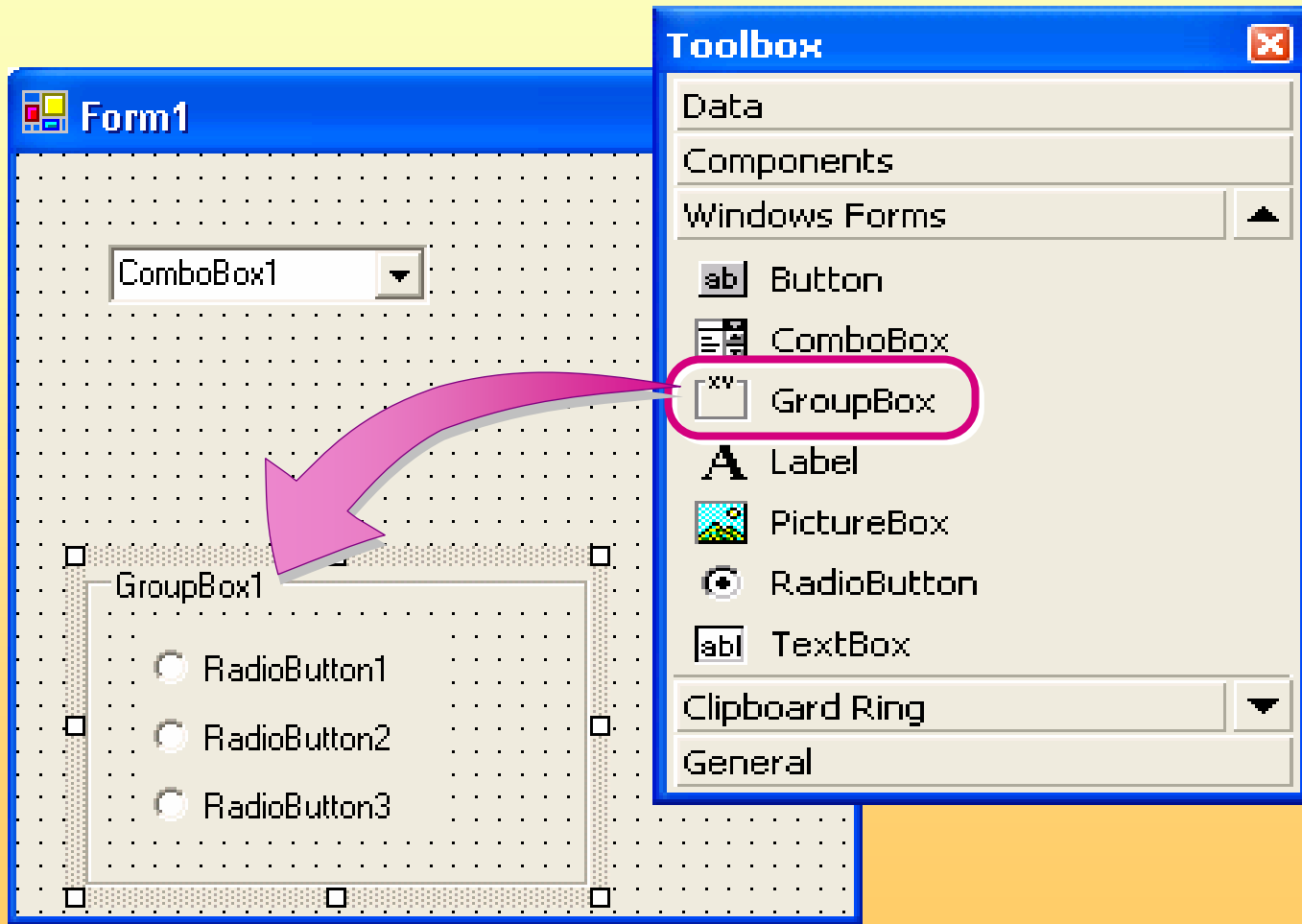
# Windows Form Designer-Generated Code



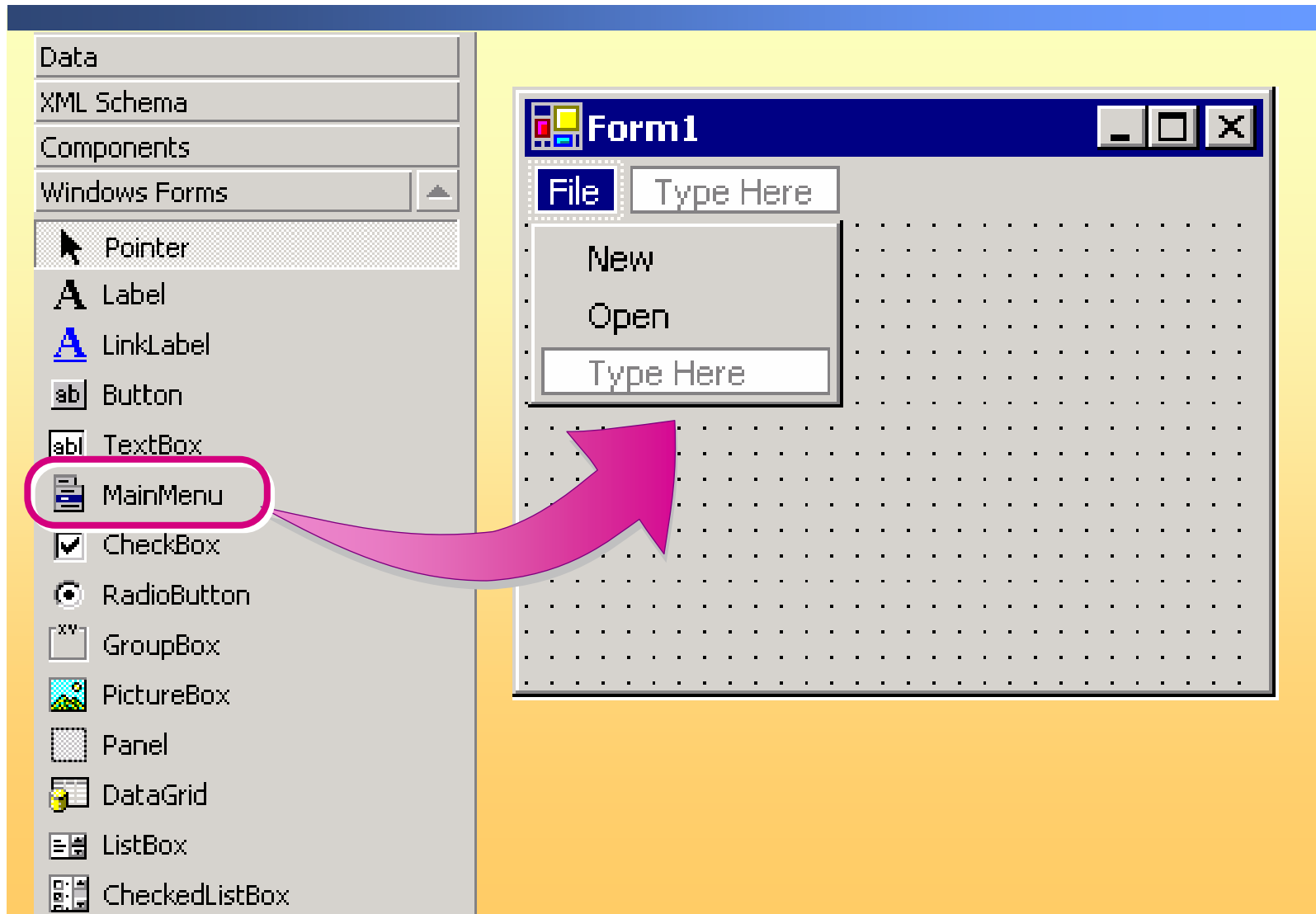
# Lesson: Adding Controls to a Form

- How to Add Controls to a Form
- How to Add Menus to a Form
- How to Customize the Controls Toolbox
- Practice: Creating a Form and Adding Controls

# How to Add Controls to a Form



# How to Add Menus to a Form

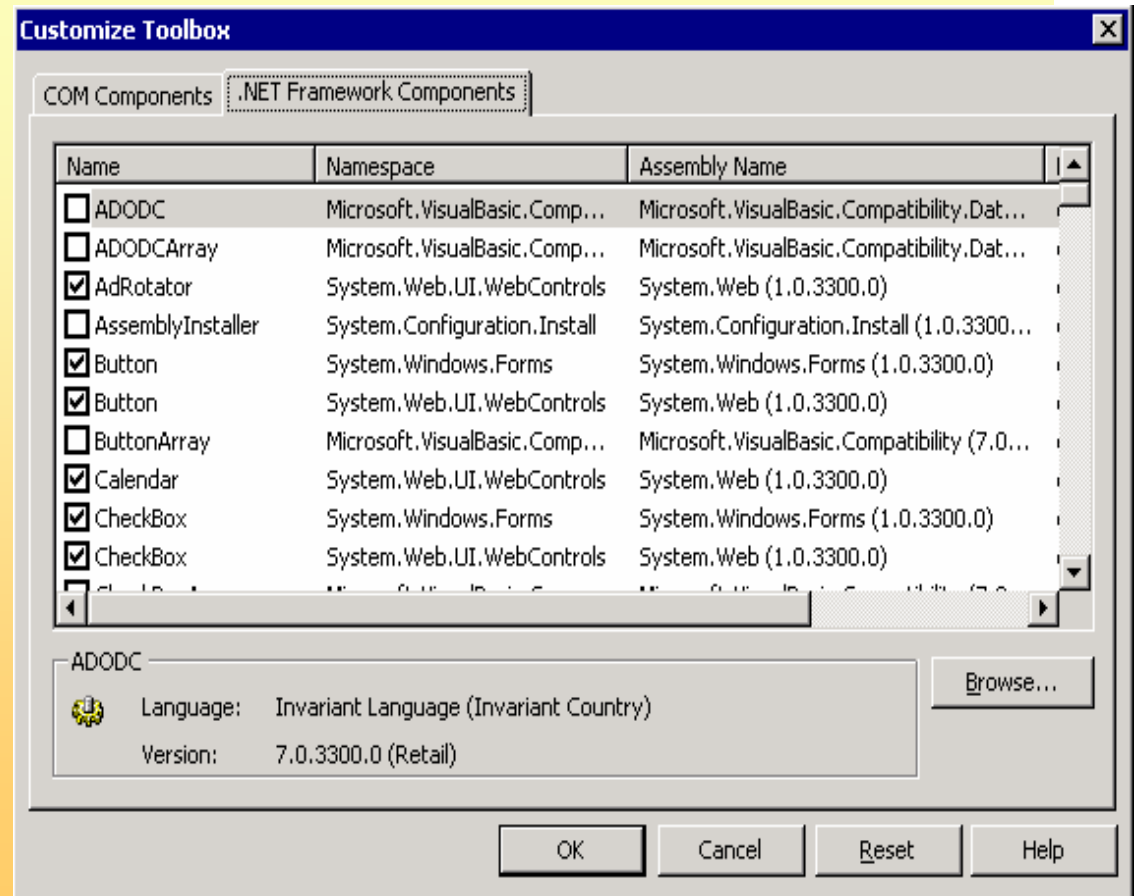


# How to Customize the Controls Toolbox

**1** Right-click the Toolbox

**2** Click Customize Toolbox

**3** Select the required control on the .NET Framework Components page





# Lesson: Creating an Inherited Form

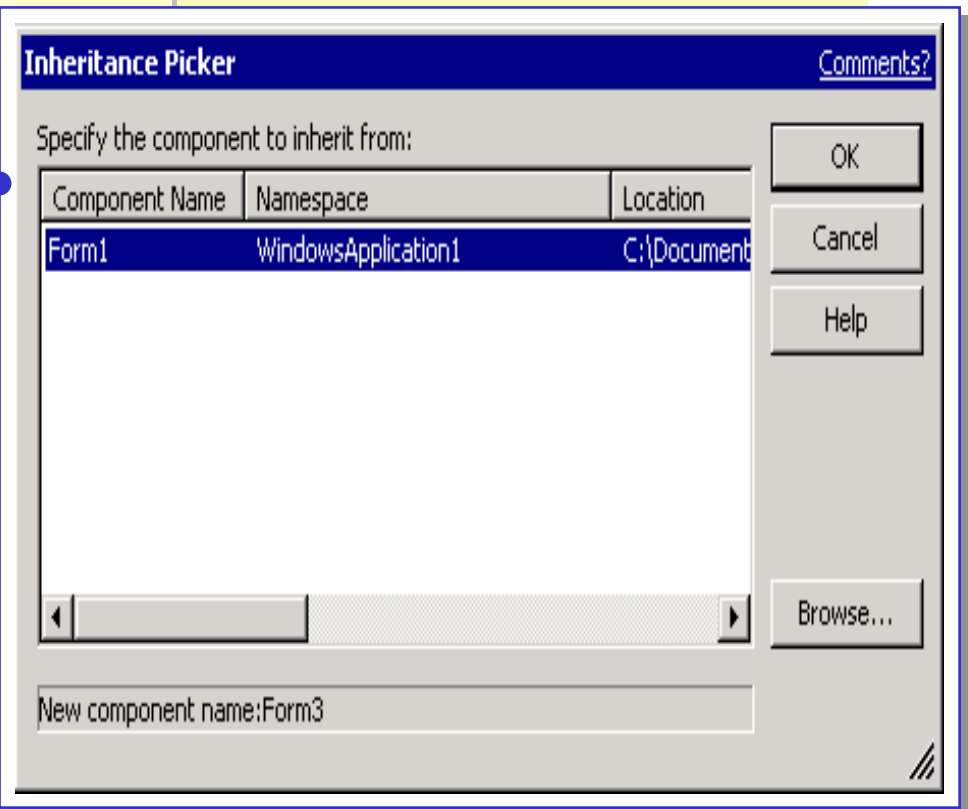
- Access Modifiers
- How to Create an Inherited Form
- Practice: Creating an Inherited Form

# Access Modifiers

Access Modifier	Description
<b>Private</b>	Read-only to a child form, all of its property values in the property browser are disabled
<b>Protected</b>	Accessible within the class and from any class that inherits from the class that declared this member
<b>Public</b>	Most permissive level. Public controls have full accessibility

# How to Create an Inherited Form

Create an inherited form by using the Inheritance Picker dialog box



Create an inherited form programmatically

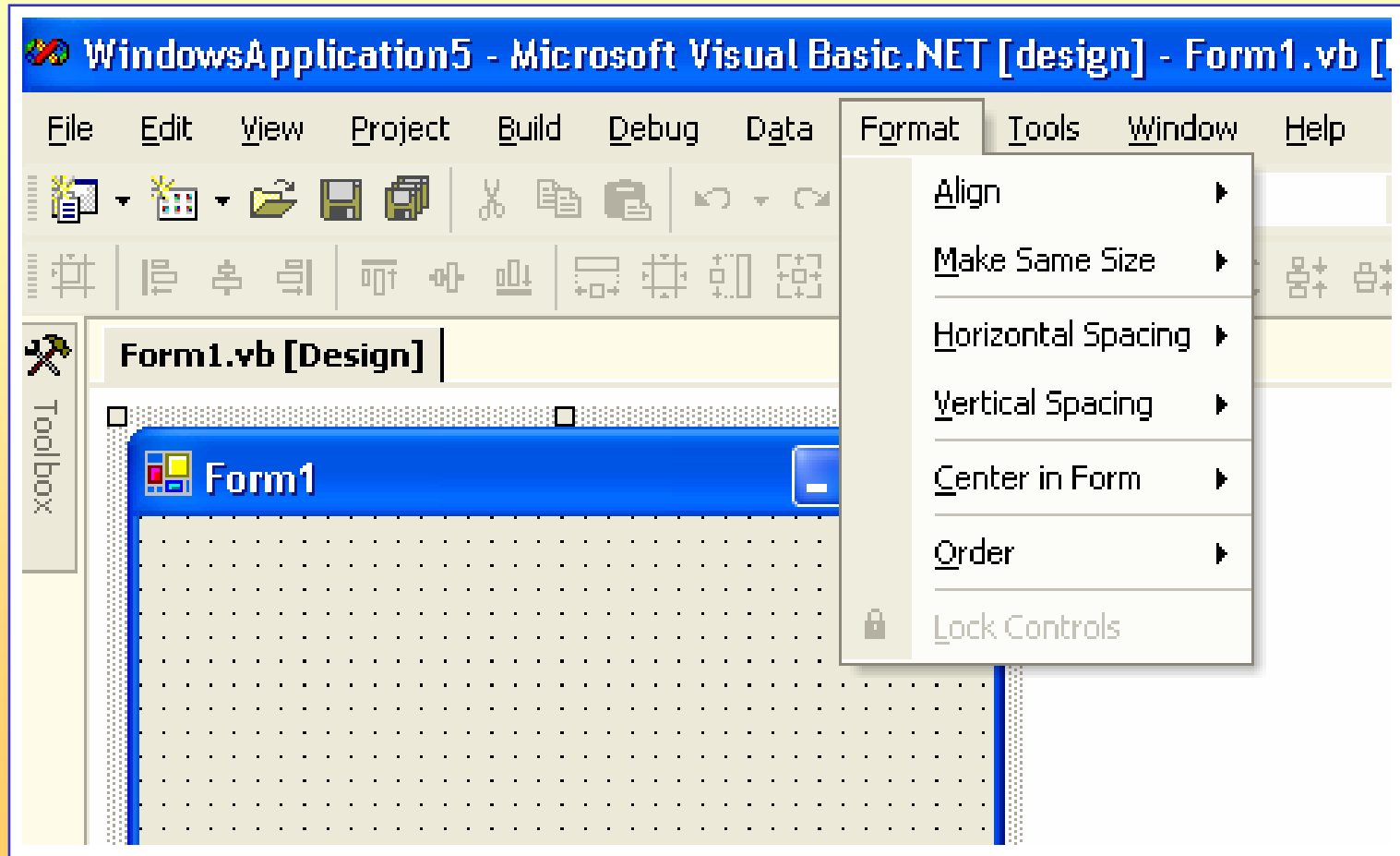
```
Public Class Form2  
    Inherits Namespace1.Form1
```



# Lesson: Organizing Controls on a Form

- How to Arrange Controls on a Form by Using the Format Menu
- How to Set the Tab Order for Controls
- How to Anchor a Control in Windows Forms
- How to Dock a Control in Windows Forms
- Demonstration: Organizing Controls on a Form

# How to Arrange Controls on a Form by Using the Format Menu



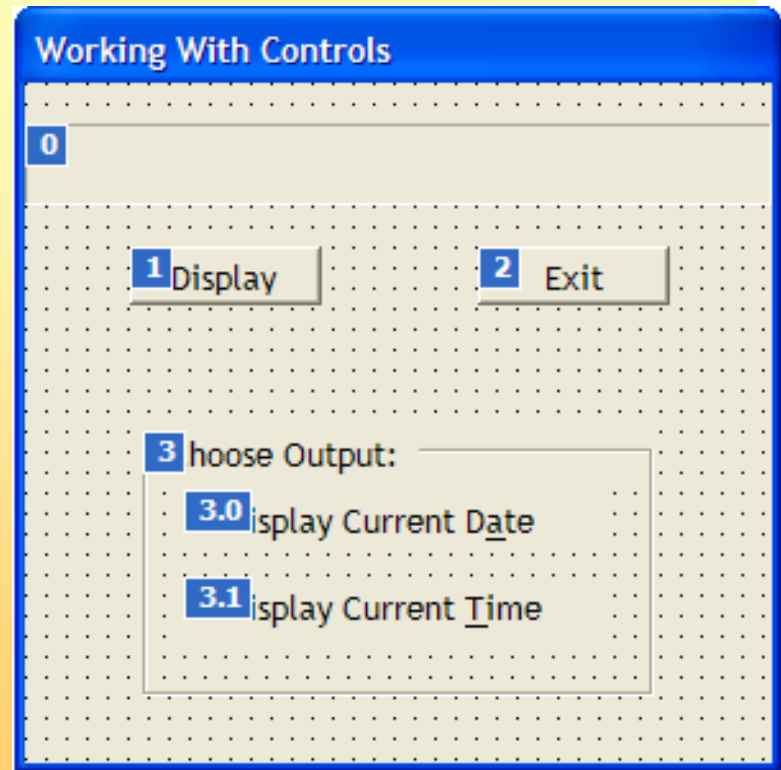
# How to Set the Tab Order for Controls

- To set the tab order for controls

- On the **View** menu, select **Tab Order**
- Click a control to change its tab order

-- OR --

- Set the **TabIndex** property
- Set the **TabStop** property to **True**



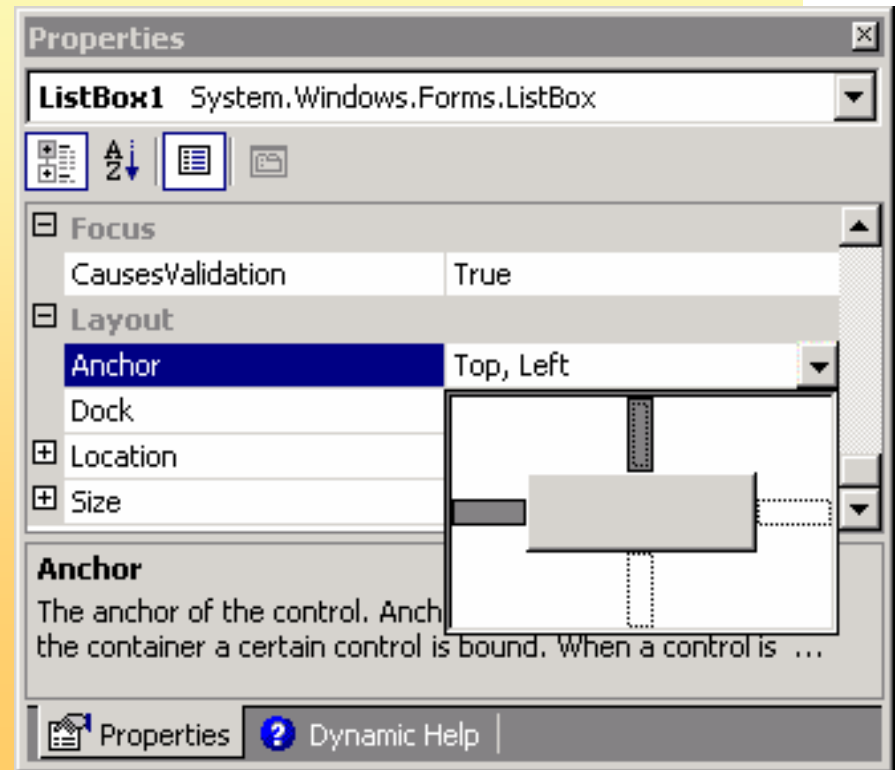
# How to Anchor a Control in Windows Forms

## ■ Anchoring

- Ensures that the edges of the control remain in the same position with respect to the parent container

## ■ To anchor a control to the form

- Set its **Anchor** property
- Default value: **Top, Left**
- Other Styles: **Bottom, Right**



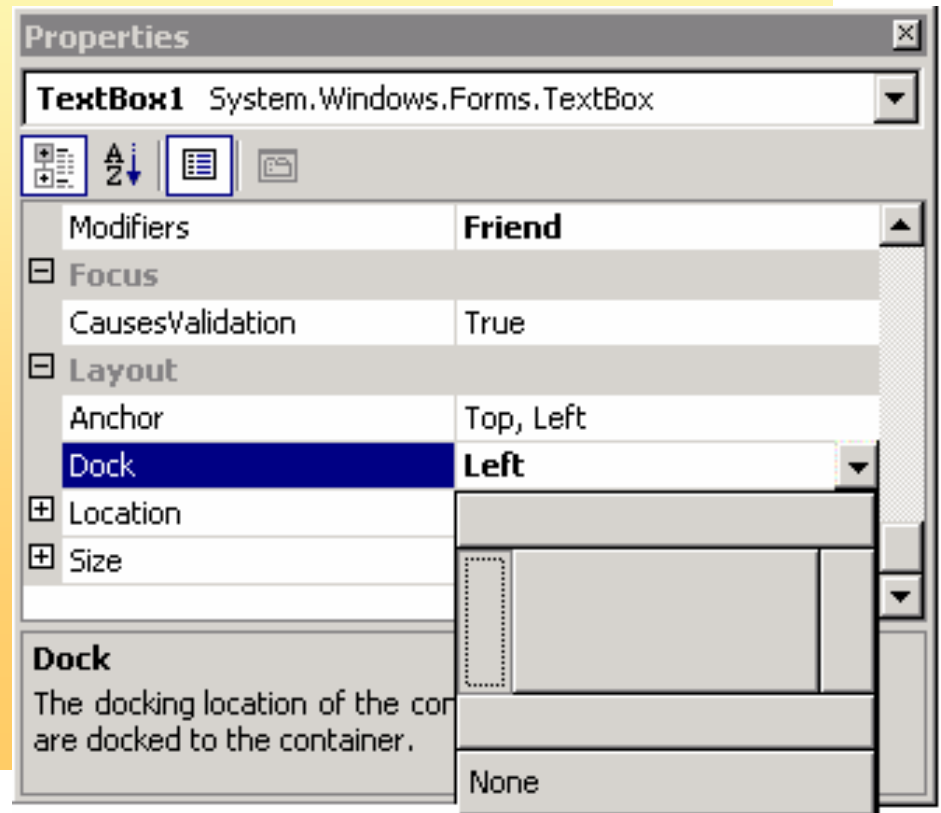
# How to Dock a Control in Windows Forms

## ■ Docking

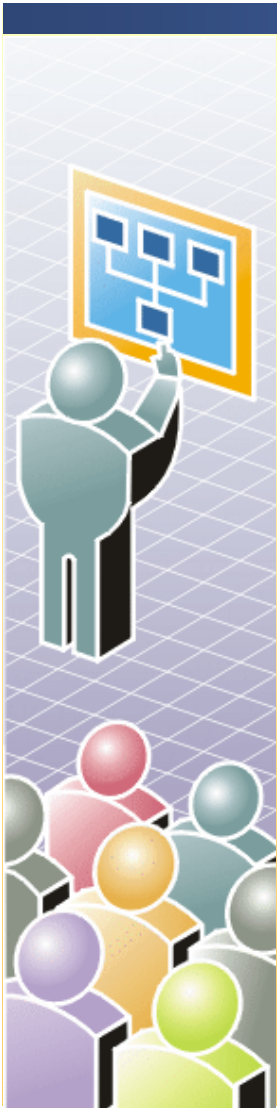
- Enables you to glue the edges of a control to the edges of its parent control

## ■ To dock a control

- Set the **Dock** property



# Demonstration: Organizing Controls on a Form



In this demonstration, you will see how to

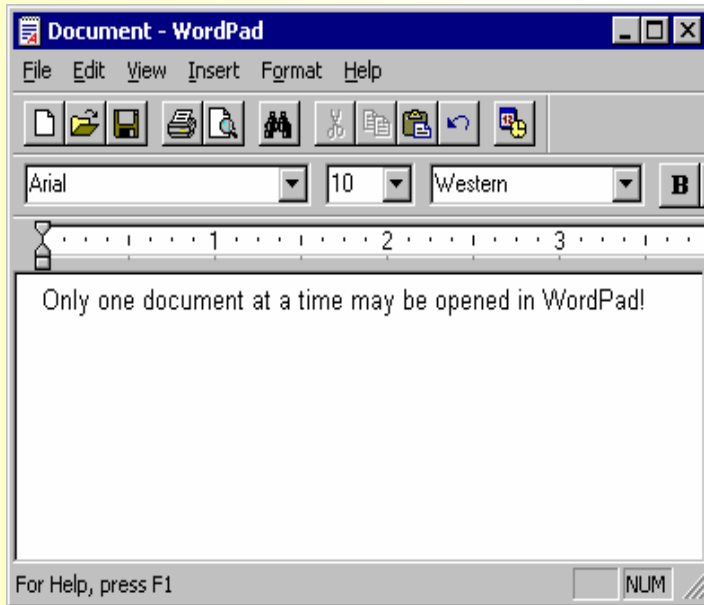
- Align controls on a form
- Layer controls on a form
- Anchor controls to a form
- Dock controls on a form

# Lesson: Creating MDI Applications

- SDI vs. MDI Applications
- How to Create MDI Applications
- How Parent and Child Forms Interact
- Practice: Creating an MDI Application

# SDI vs. MDI Applications

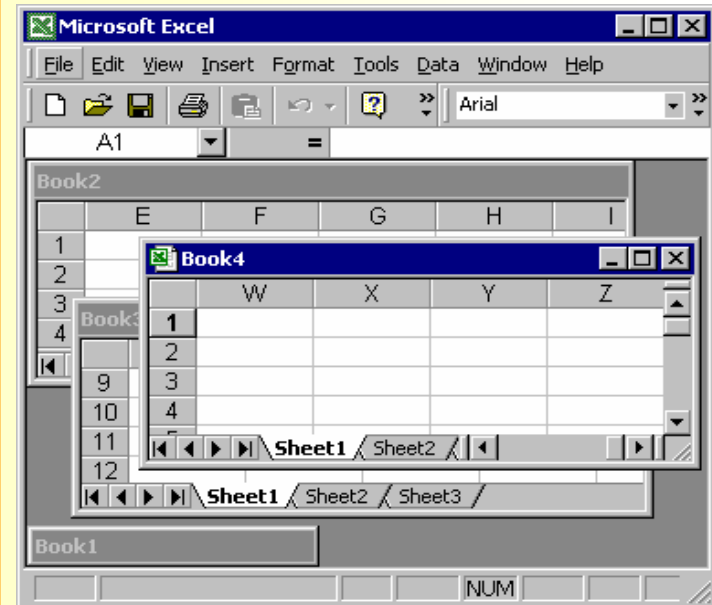
## SDI



Only one document is visible

You must close one document before you open another

## MDI



Displays multiple documents at the same time

Each document is displayed in its own window

# How to Create MDI Applications

- To create a parent form
  - Create a new project
  - Set the **IsMdiContainer** property to **True**
  - Add a menu item to invoke the child form
- To create a child form
  - Add a new form to the project
- To call a child form from a parent form

```
Protected Sub MenuItem2_OnClick(ByVal sender As System.Object, ByVal  
e As System.EventArgs) Handles MenuItem2.Click  
Dim NewMdiChild As New Form2()  
'Set the Parent Form of the Child window.  
NewMdiChild.MdiParent = Me  
'Display the new form.  
NewMdiChild.Show()  
End Sub
```

# How Parent and Child Forms Interact

- To list the available child windows that are owned by the parent
  - Create a menu item (Windows) and set its **MdiList** property to **True**
- To determine the active MDI child
  - Use the **ActiveMdiChild** property

```
Dim activeChild As Form = Me.ActiveMdiChild
```

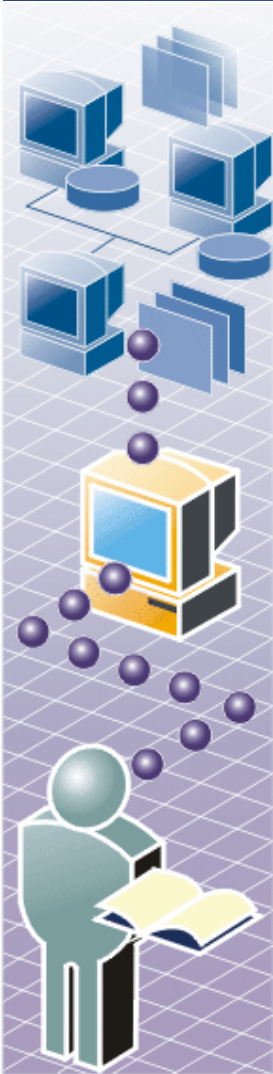
- To arrange child windows on the parent form
  - Call the **LayoutMdi** method



# Review

- Creating a Form
- Adding Controls to a Form
- Creating an Inherited Form
- Organizing Controls on a Form
- Creating MDI Applications

# Lab 1.1: Creating Windows Forms



- Exercise 1: Creating a New Windows Form
- Exercise 2: Inheriting a New Form from an Existing Windows Form