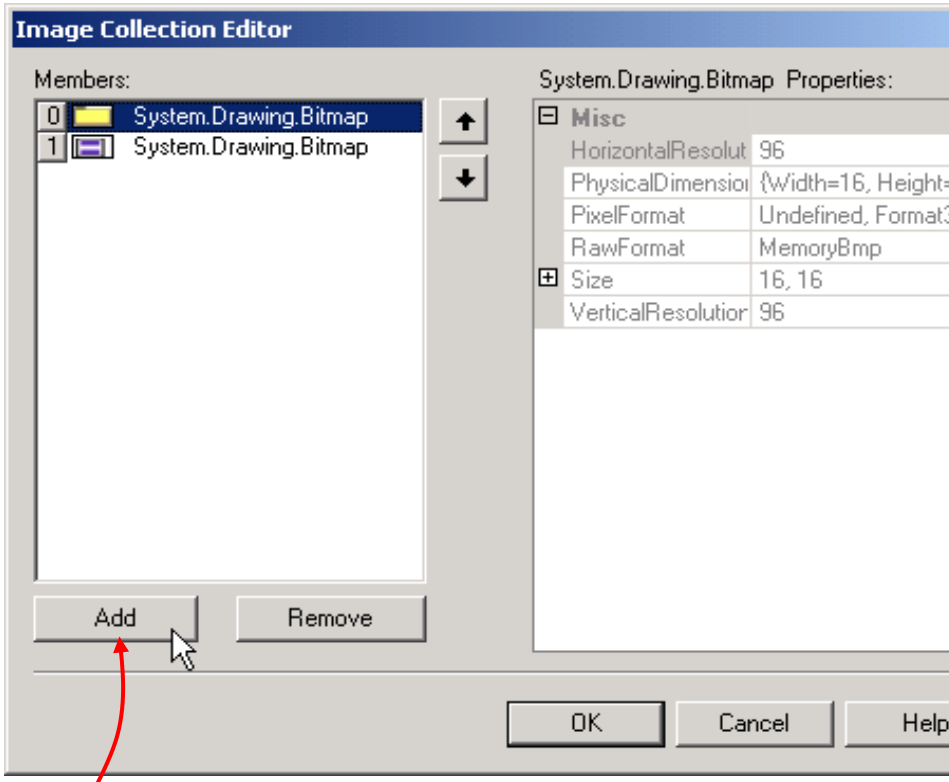
Chapter 13 – Graphical User Interfaces Part 2/2

13.7 ListViews

- Listviews to displays list of items
  - Can select one or more items from list
  - Displays icons to go along with items
- **ImageList** component in this program
  - dragging it from the **ToolBox**
  - click the **Images** collection in **Properties** window to display the **Image Collection Editor**
  - developers can browse for images that they wish to add to the **ImageList**, which contains an array of **Image**s.

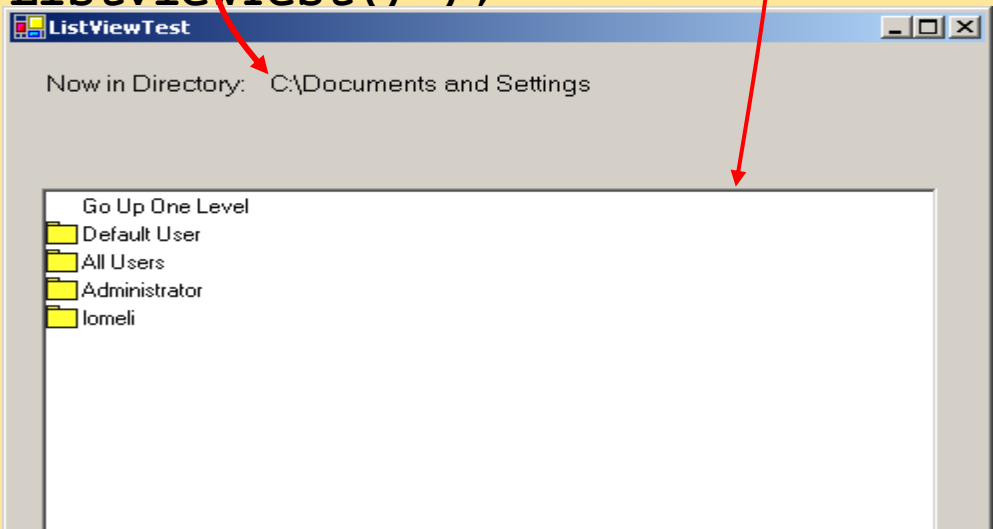


13.7 ListViews

ListView events and properties	Description / Delegate and Event Arguments
Common Properties	
Activation	Determines how the user activates an item. This property takes a value in the ItemActivation enumeration. Possible values are OneClick (single-click activation), TwoClick (double-click activation, item changes color when selected) and Standard (double-click activation).
CheckBoxes	Indicates whether items appear with checkboxes. True displays checkboxes. Default is False.
LargeImageList	Indicates the ImageList used when displaying large icons.

ListView events and properties	**Description / Delegate and Event Arguments**
Items	Returns the collection of **ListViewItem**s in the control.
MultiSelect	Determines whether multiple selection is allowed. Default is **True**, which enables multiple selection.
SelectedItems	Lists the collection of currently selected items.
SmallImageList	Specifies the **ImageList** used when displaying small icons.
View	Determines appearance of **ListViewItem**s. Values **LargeIcon** (large icon displayed, items can be in multiple columns), **SmallIcon** (small icon displayed), **List** (small icons displayed, items appear in a single column) and **Details** (like **List**, but multiple columns of information can be displayed per item).
Common Event	*(Delegate **EventHandler**; event arguments **EventArgs**)*
ItemActivate	Generated when an item in the **ListView** is activated. Does not specify which item is activated.

```
4     using System;
5     using System.Drawing;
6     using System.Collections;
7     using System.ComponentModel;
8     using System.Windows.Forms;
9     using System.Data;
10    using System.IO;
12    public class ListViewTest : System.Windows.Forms.Form {
16       private System.Windows.Forms.Label currentLabel;
17       private System.Windows.Forms.Label displayLabel;
20       private System.Windows.Forms.ListView browserListView;
23       private System.Windows.Forms.ImageList fileFolder;
26       string currentDirectory= Directory.GetCurrentDirectory();
29       [STAThread]
30       static void Main() {
32          Application.Run( new ListViewTest() );
33       }
34
```



```
36  private void browserListView_Click(
37      object sender, System.EventArgs e ) {
40    if ( browserListView.SelectedItems.Count != 0 ){
43      if ( browserListView.Items[ 0 ].Selected ) {
46        DirectoryInfo directoryObject =
47                  new DirectoryInfo( currentDirectory );
50        if ( directoryObject.Parent != null )
51          LoadFilesInDirectory(directoryObject.Parent.FullName);
53        }
56      else {
59        string chosen = browserListView.SelectedItems[0].Text;
63        if (Directory.Exists(currentDirectory + "\\"+chosen)){
69          if ( currentDirectory == "C:\\" )
70            LoadFilesInDirectory(currentDirectory + chosen );
72          else
73            LoadFilesInDirectory(currentDirectory+"\\"+chosen);
75        }
77      }
80    displayLabel.Text = currentDirectory;
82  }
84  }
```

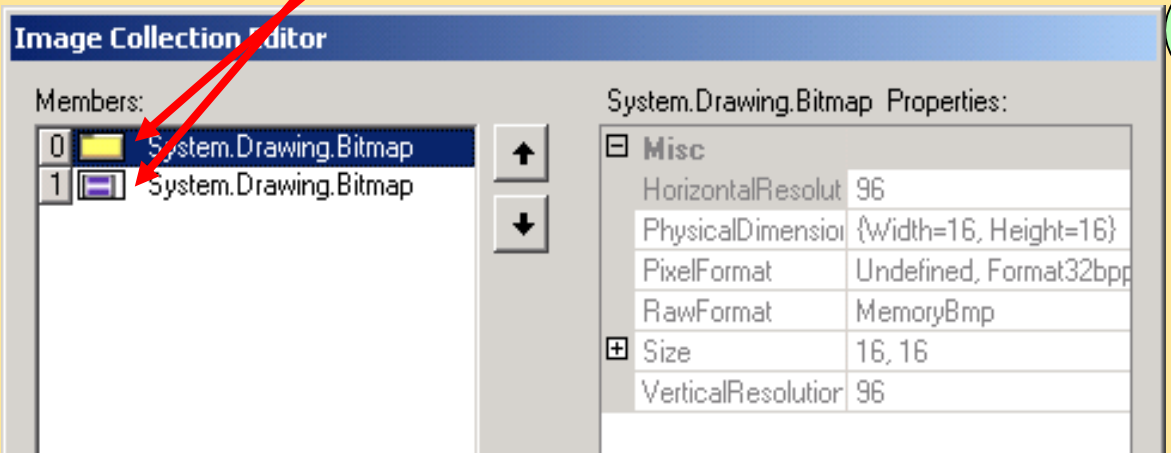
select “go up one level” ?

only root directory has “\” in its suffix

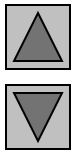
```
87   public void LoadFilesInDirectory(
88     string currentDirectoryValue ) {
91   try {
94     browserListView.Items.Clear();
95     browserListView.Items.Add( "Go Up One Level" );
98     currentDirectory = currentDirectoryValue;
99     DirectoryInfo newCurrentDirectory =
100           new DirectoryInfo(currentDirectory );
103    DirectoryInfo[] directoryArray =
104          newCurrentDirectory.GetDirectories();
106    FileInfo[] fileArray = newCurrentDirectory.GetFiles();
110    foreach ( DirectoryInfo dir in directoryArray ) {
113      ListViewItem newDirectoryItem =
                  browserListView.Items.Add(dir.Name);
117      newDirectoryItem.ImageIndex = 0; // set directory image
118    }
121    foreach ( FileInfo file in fileArray ) {
124      ListViewItem newFileItem =
                  browserListView.Items.Add( file.Name );
127      newFileItem.ImageIndex = 1;  // set file image
128    }
129  }
```

Method returns subdirectories of current directory.

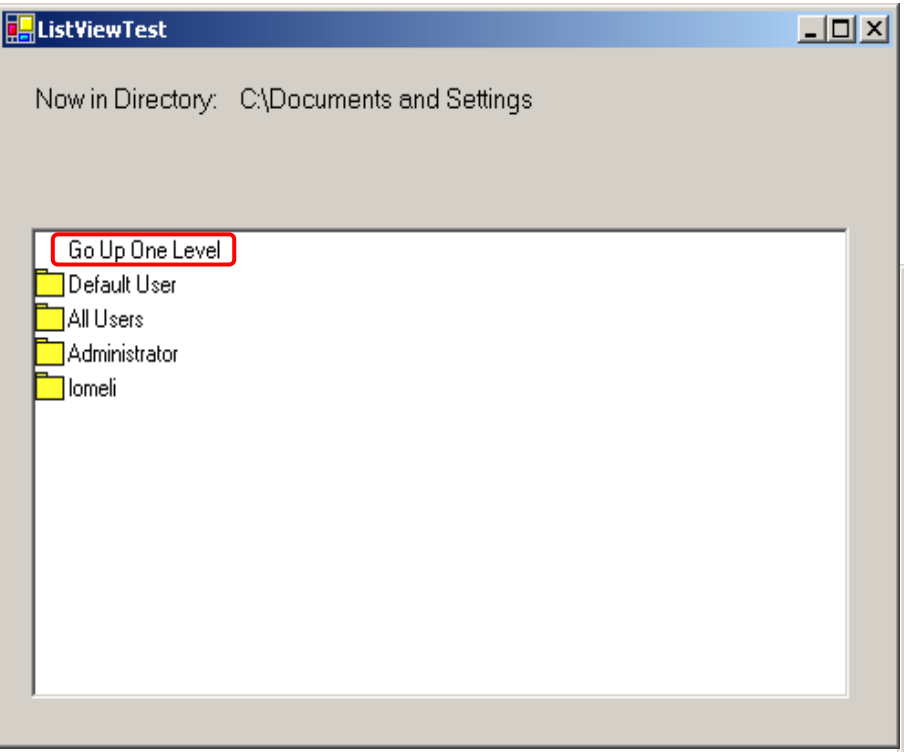
```
132         catch ( UnauthorizedAccessException exception ) {
134            MessageBox.Show(
135               "Warning: Some fields may not be " +
136               "visible due to permission settings",
137               "Attention", 0, MessageBoxIcon.Warning );
138         }
140      }
143      private void ListViewTest_Load(
144         object sender, System.EventArgs e ){
147         Image folderImage = Image.FromFile(
148            currentDirectory + "\\images\\folder.bmp" );
150         Image fileImage = Image.FromFile( currentDirectory +
151            "\\images\\file.bmp" );
153         fileFolder.Images.Add( folderImage );
154         fileFolder.Images.Add( fileImage );
157         LoadFilesInDirectory( currentDirectory );
158         displayLabel.Text = currentDirectory;
160      }
162   }
```

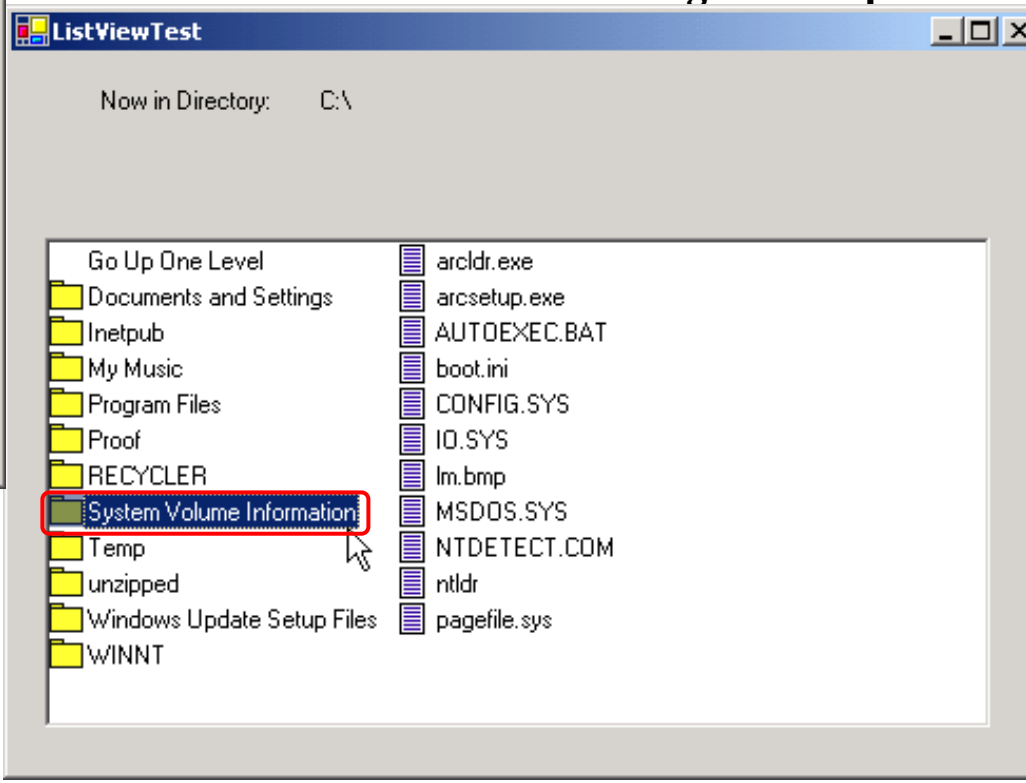


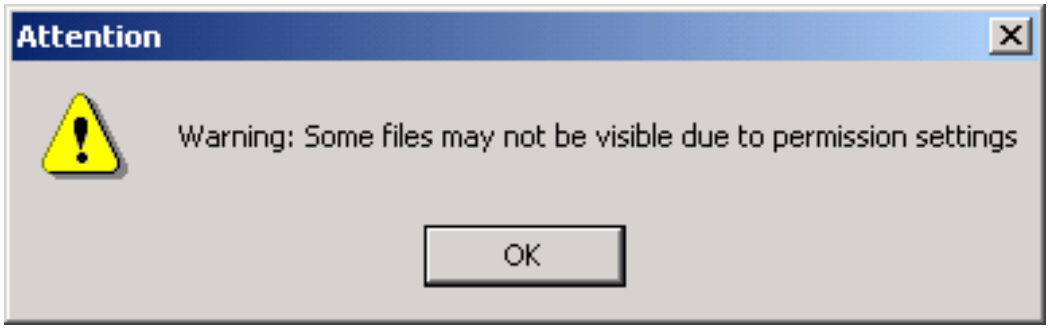
Written by programmer



`ListViewTest.cs`
Program Output

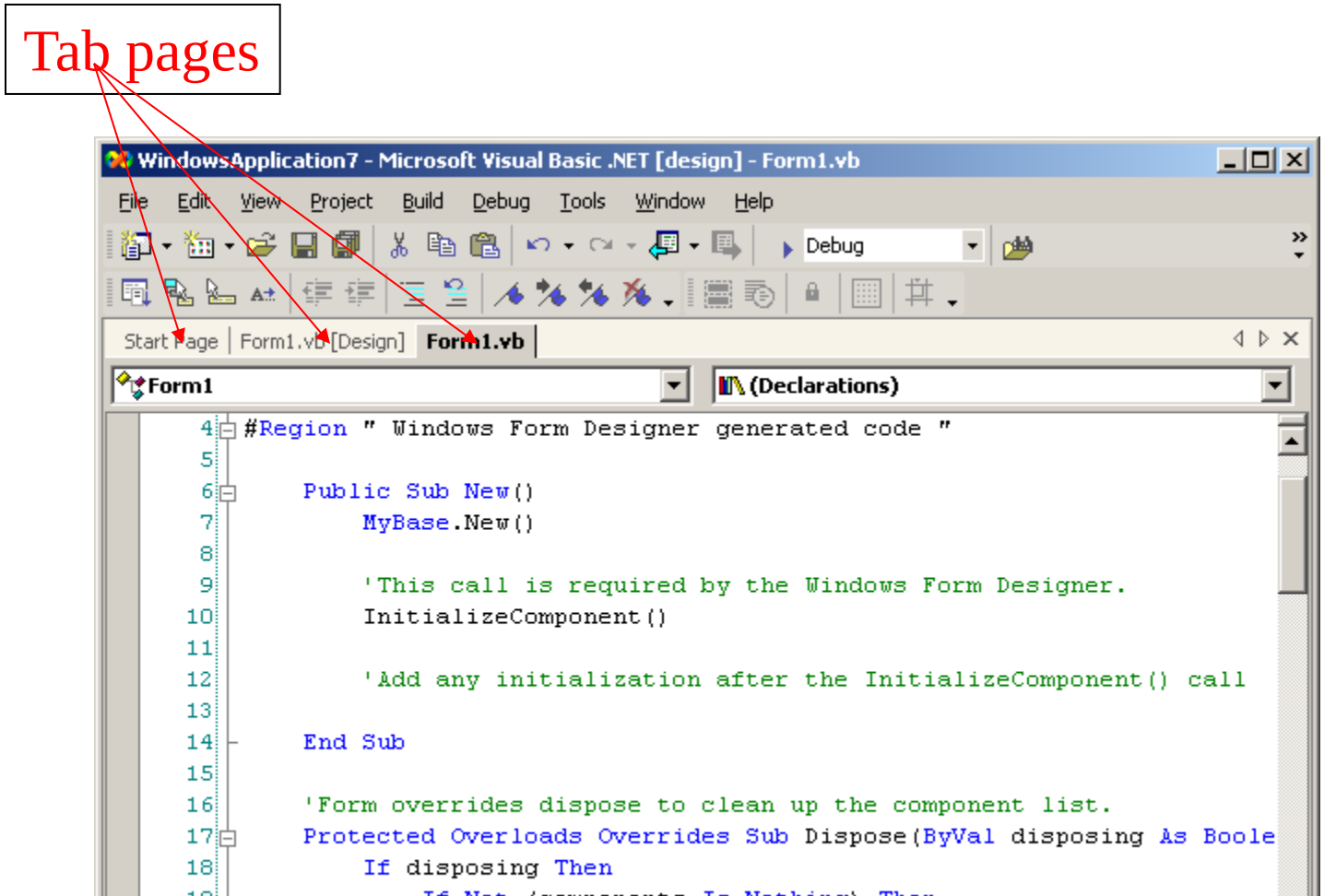




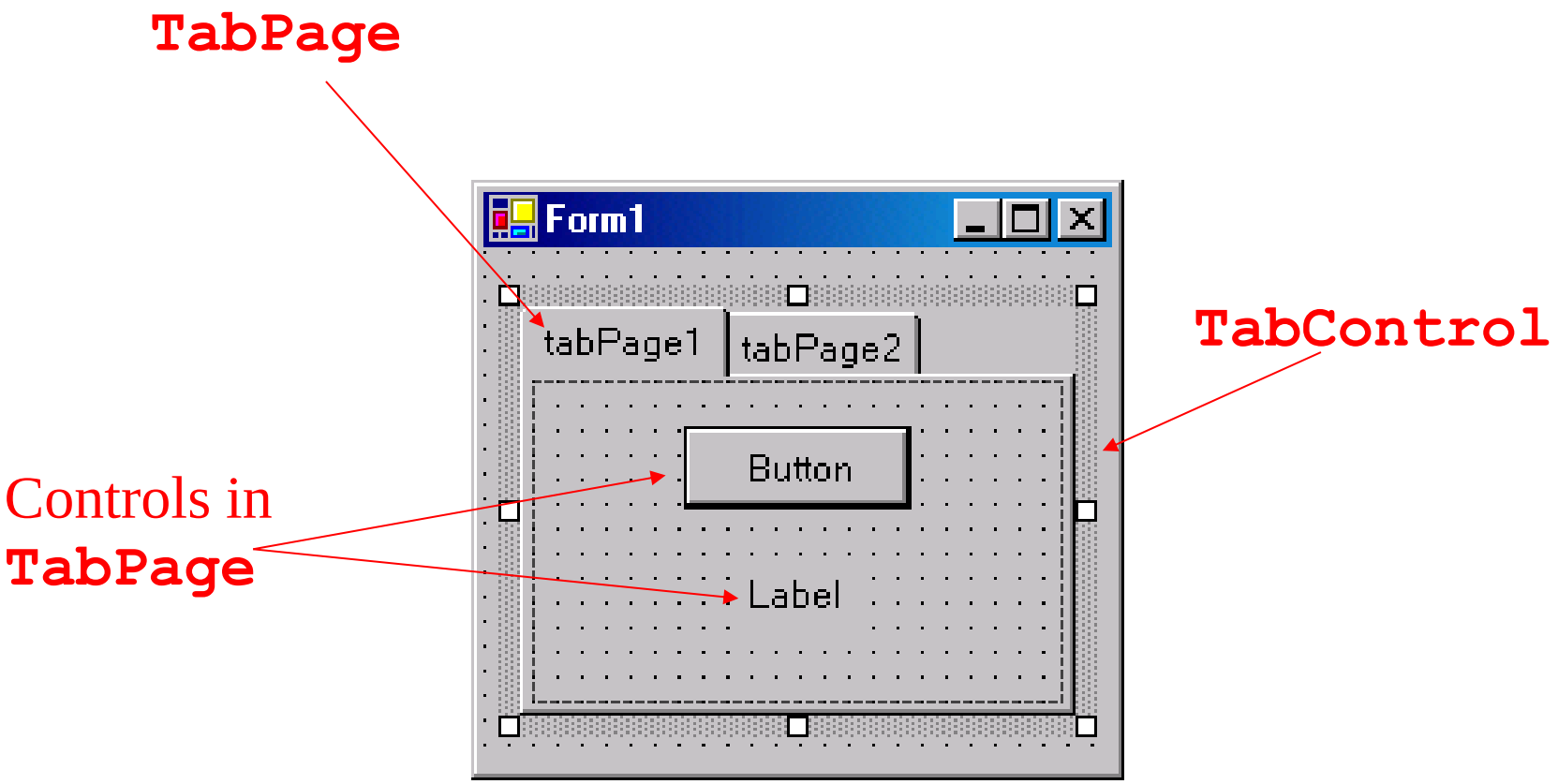


13.8 TabControl

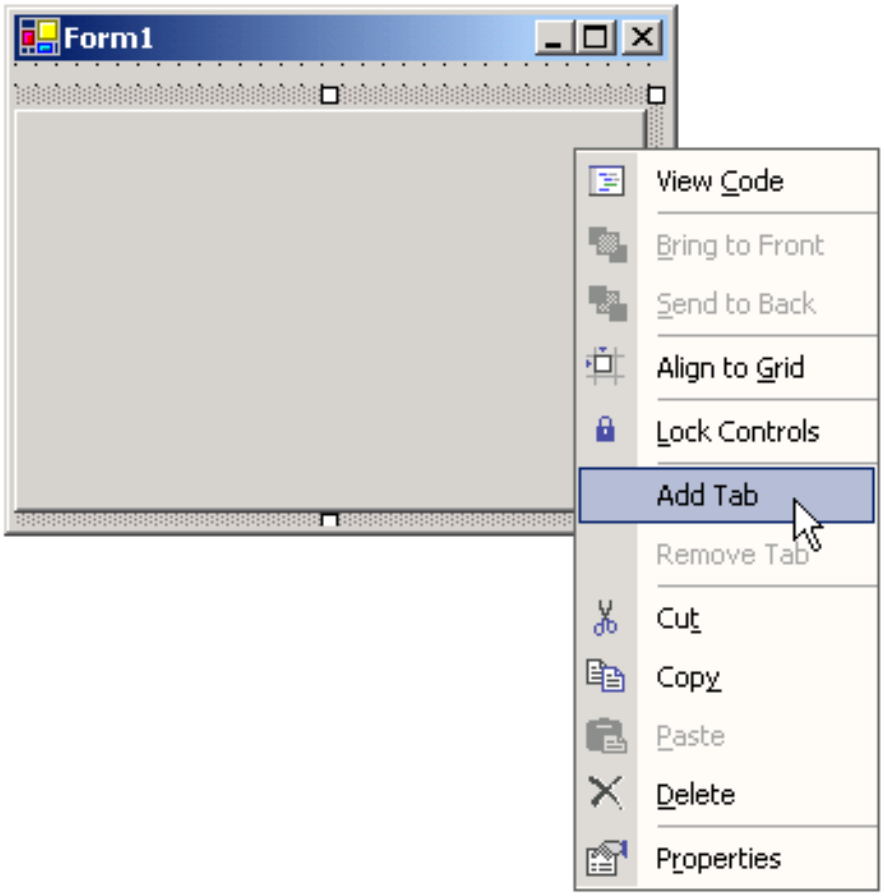
- Tabcontrol creates tabbed windows
- Windows called **TabPage** objects
  - **TabPages** can have controls
  - **Tabpages** have own **Click** event for when tab is clicked



13.8 Tab Controls



13.8 Tab Controls



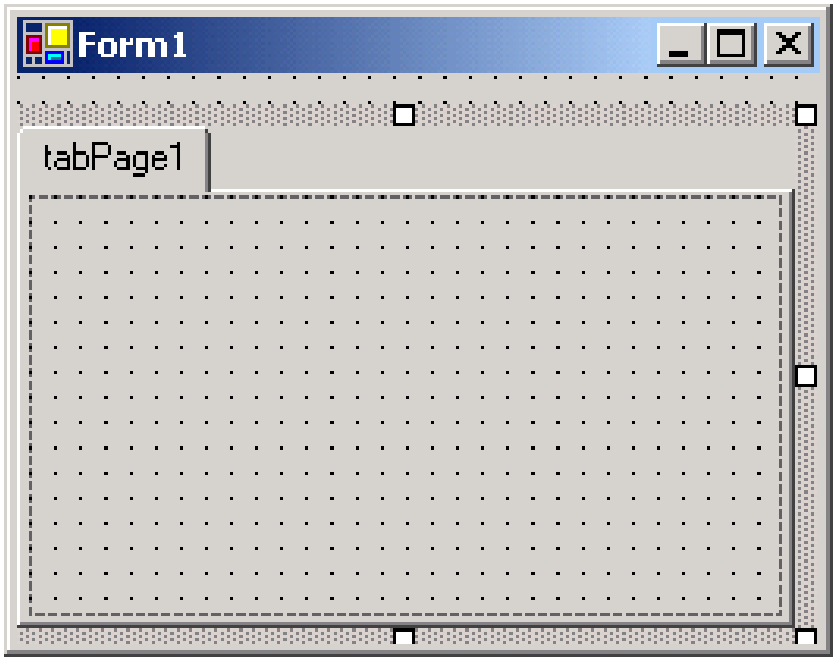


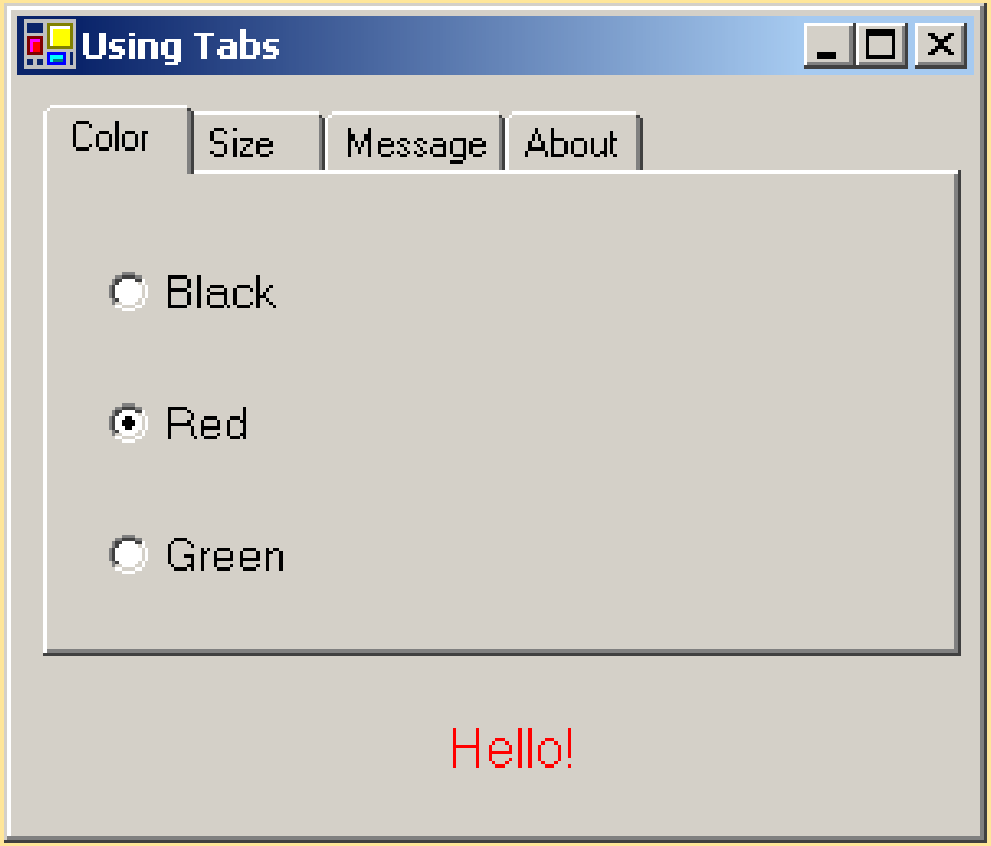
Fig. 13.27 Adding **`TabPage`**s to the **`TabControl`**.

13.8 Tab Controls

TabControl properties and events	Description / Delegate and Event Arguments
Common Properties	
ImageList	Specifies images to be displayed on a tab.
ItemSize	Specifies tab size.
MultiLine	Indicates whether multiple rows of tabs can be displayed.
SelectedIndex	Indicates index of **TabPage** that is currently selected.
SelectedTab	Indicates the **TabPage** that is currently selected.
TabCount	Returns the number of tabs.
TabPages	Gets the collection of **TabPage**s within our **TabControl**.
Common Event	*(Delegate **EventHandler**, event arguments **EventArgs**)*
SelectedIndexChanged	Generated when **SelectedIndex** changes (i.e., another **TabPage** is selected).

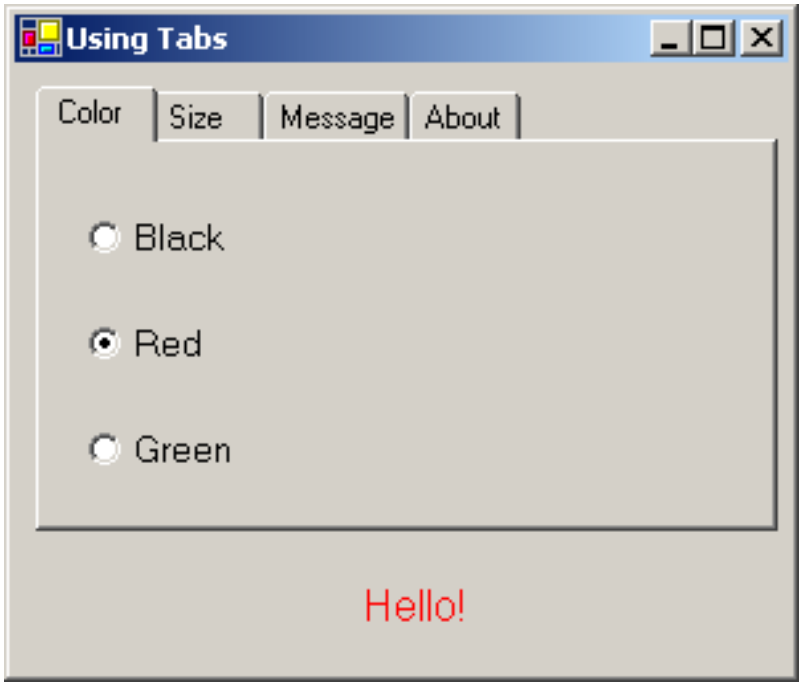
```
4     using System;
5     using System.Drawing;
6     using System.Collections;
7     using System.ComponentModel;
8     using System.Windows.Forms;
9     using System.Data;
11    public class UsingTabs : System.Windows.Forms.Form {
14      private System.Windows.Forms.Label displayLabel;
18      private System.Windows.Forms.TabControl optionsTabControl;
22      private System.Windows.Forms.TabPage colorTabPage;
23      private System.Windows.Forms.RadioButton greenRadioButton;
25      private System.Windows.Forms.RadioButton redRadioButton;
26      private System.Windows.Forms.RadioButton blackRadioButton;
30      private System.Windows.Forms.TabPage sizeTabPage;
31      private System.Windows.Forms.RadioButton size20RadioButton;
33      private System.Windows.Forms.RadioButton size16RadioButton;
35      private System.Windows.Forms.RadioButton size12RadioButton;
39      private System.Windows.Forms.TabPage messageTabPage;
40      private System.Windows.Forms.RadioButton
41          goodByeRadioButton;
42      private System.Windows.Forms.RadioButton
43          helloRadioButton;
```

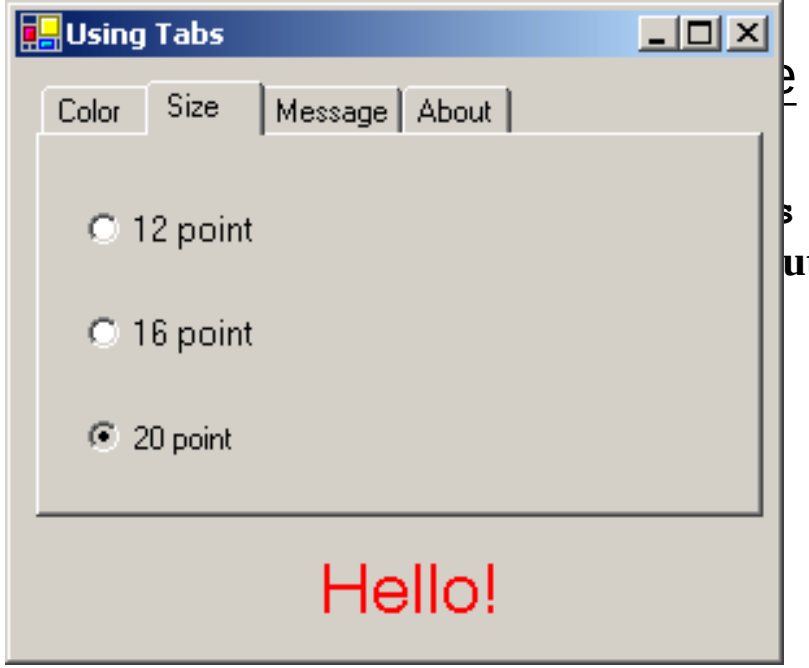
```
46      private System.Windows.Forms.TabPage aboutTabPage;
47      private System.Windows.Forms.Label messageLabel;
49      [STAThread]
50      static void Main() {
52         Application.Run( new UsingTabs() );
53      }
56      private void blackRadioButton_CheckedChanged(
57         object sender, System.EventArgs e ){
59         displayLabel.ForeColor = Color.Black;
60      }
61
```

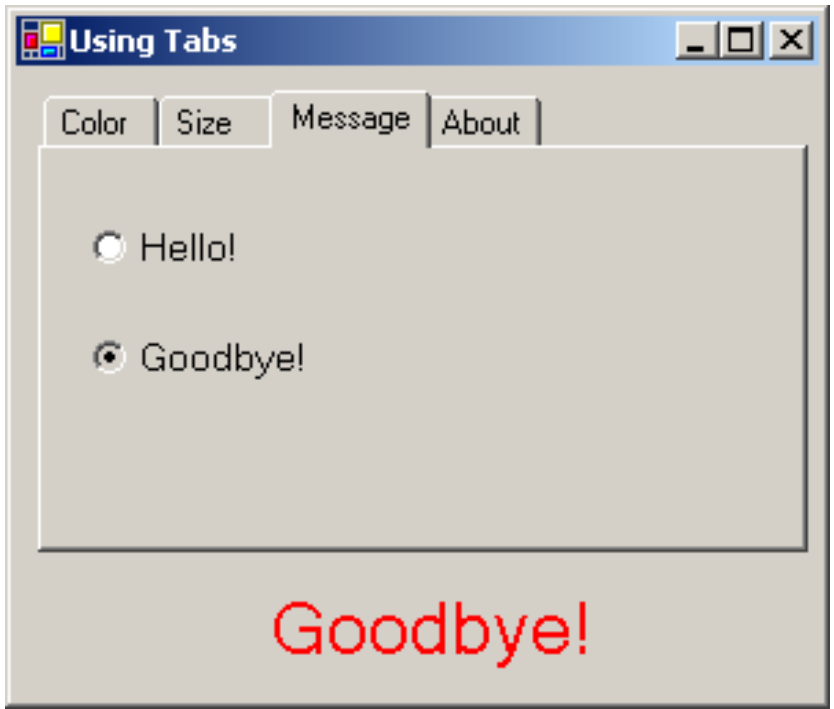


```
63      private void redRadioButton_CheckedChanged(
64         object sender, System.EventArgs e ){
66         displayLabel.ForeColor = Color.Red;
67      }
70      private void greenRadioButton_CheckedChanged(
71         object sender, System.EventArgs e ) {
73         displayLabel.ForeColor = Color.Green;
74      }
77      private void size12RadioButton_CheckedChanged(
78         object sender, System.EventArgs e ) {
80         displayLabel.Font =
81            new Font( displayLabel.Font.Name, 12 );
82      }
85      private void size16RadioButton_CheckedChanged(
86         object sender, System.EventArgs e ) {
88         displayLabel.Font =
89            new Font( displayLabel.Font.Name, 16 );
90      }
91
```

```
93      private void size20RadioButton_CheckedChanged(
94         object sender, System.EventArgs e ) {
96         displayLabel.Font =
97            new Font( displayLabel.Font.Name, 20 );
98      }
101     private void helloRadioButton_CheckedChanged(
102        object sender, System.EventArgs e ){
104        displayLabel.Text = "Hello!";
105     }
108     private void goodByeRadioButton_CheckedChanged(
109        object sender, System.EventArgs e ) {
111        displayLabel.Text = "Goodbye!";
112     }
114  }
```



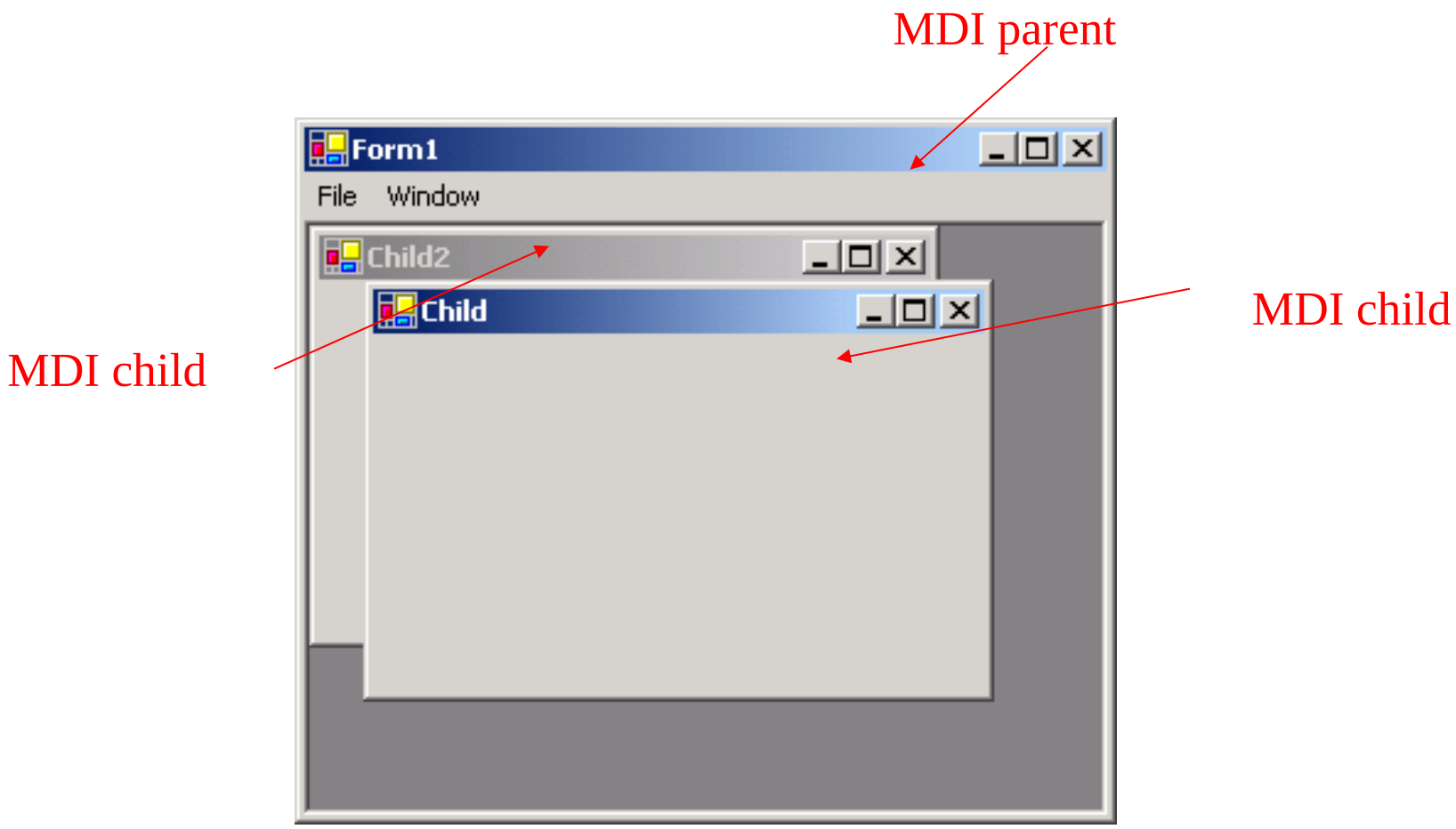




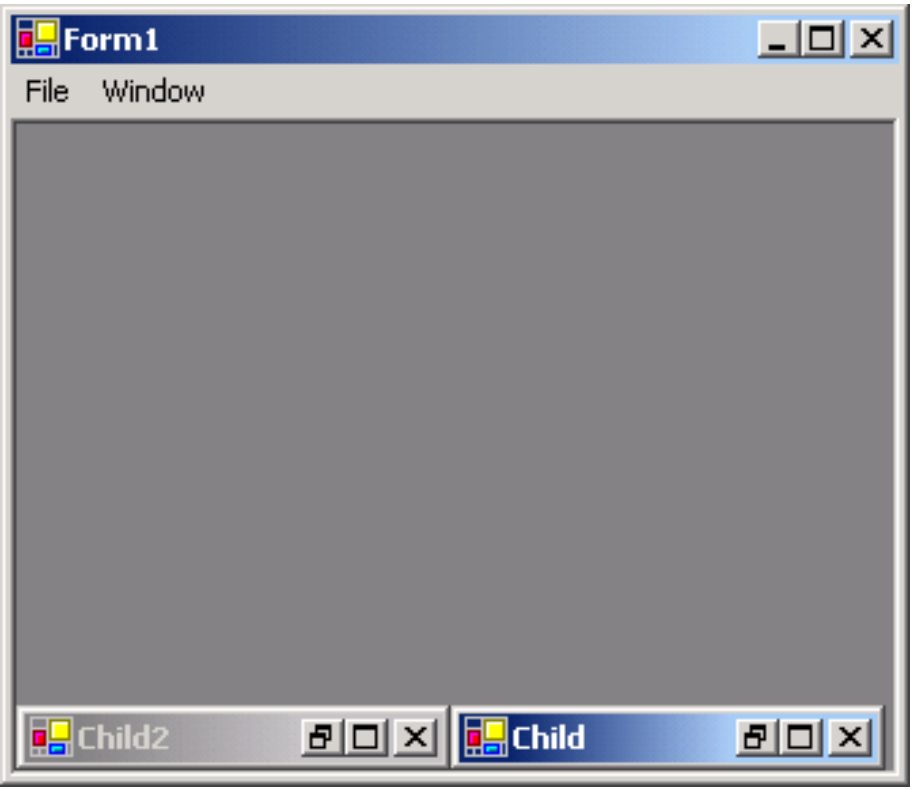
13.9 Multiple-Document Interface Windows

- *single-document-interface (SDI):*
  - support only one open window or document at a time.
  - Users can edit multiple documents at once
- *Multiple document interface (MDI):*
  - enable users to edit multiple documents at once.
  - Application window called parent, others child
- To create an MDI form,
  - create a new **Form** and set its ***IsMDIContainer*** property to **True**
- Parent and child menus can be merged
  - Based on **`MergeOrder`** property
- Child windows can be arranged in parent window:
  - Tiled windows:
  - Cascaded windows:
  - ArrangeIcons:

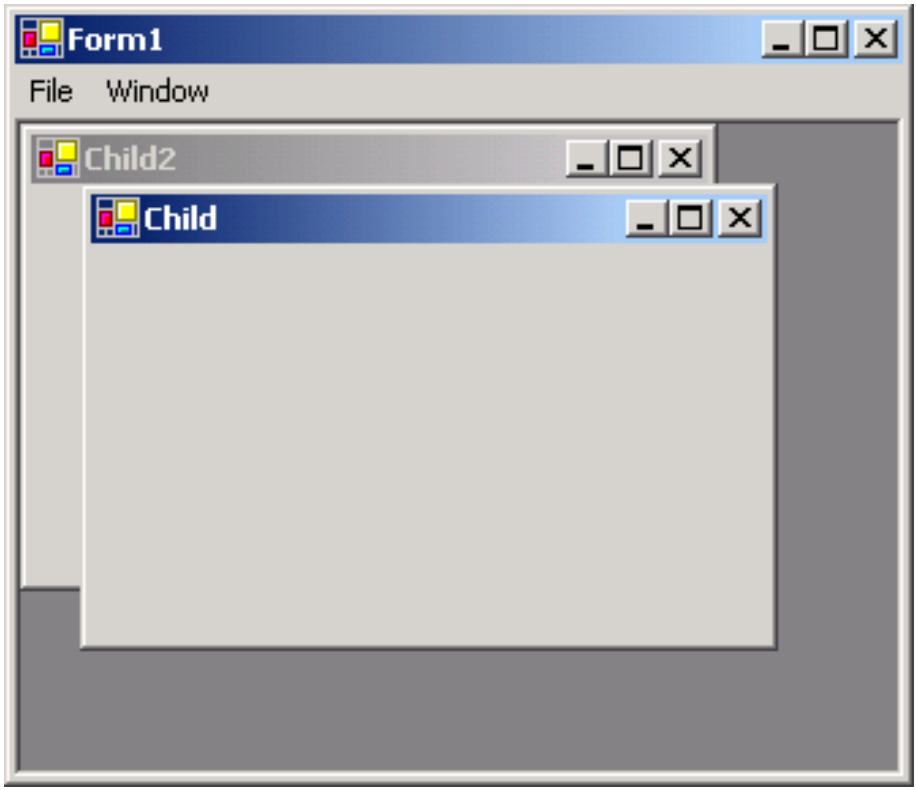
13.9 Multiple Document Interface (MDI) Windows



13.9 Multiple Document Interface (MDI) Windows



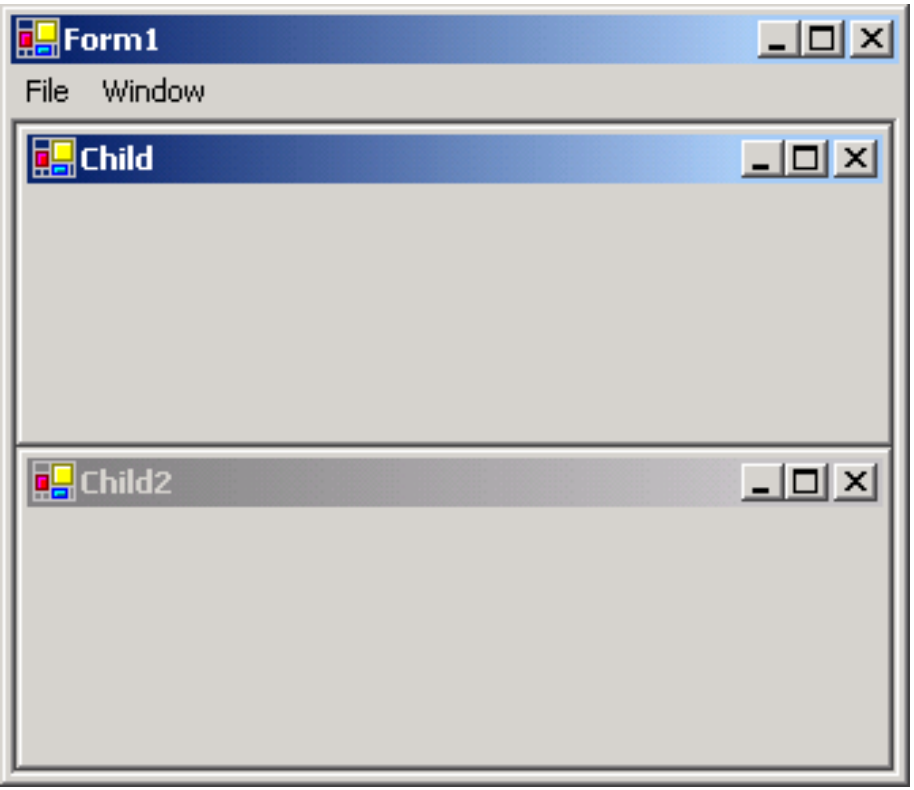
`ArrangeIcons`



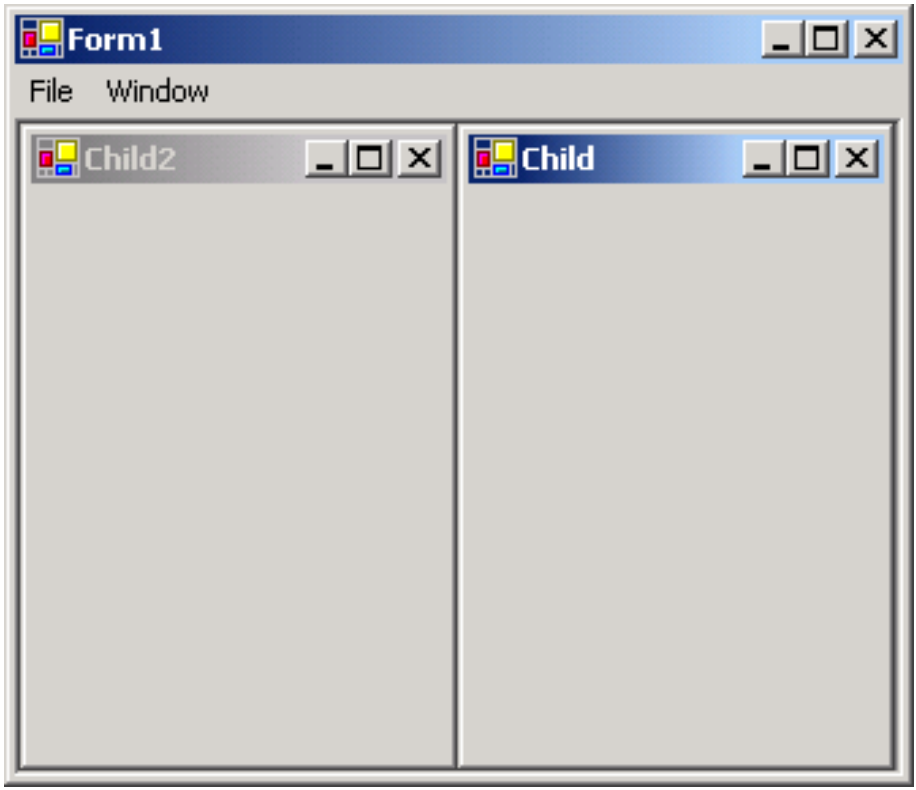
`Cascade`

Fig. 13.35 **`LayoutMdi`** enumeration values (Part 1).

13.9 Multiple Document Interface (MDI) Windows



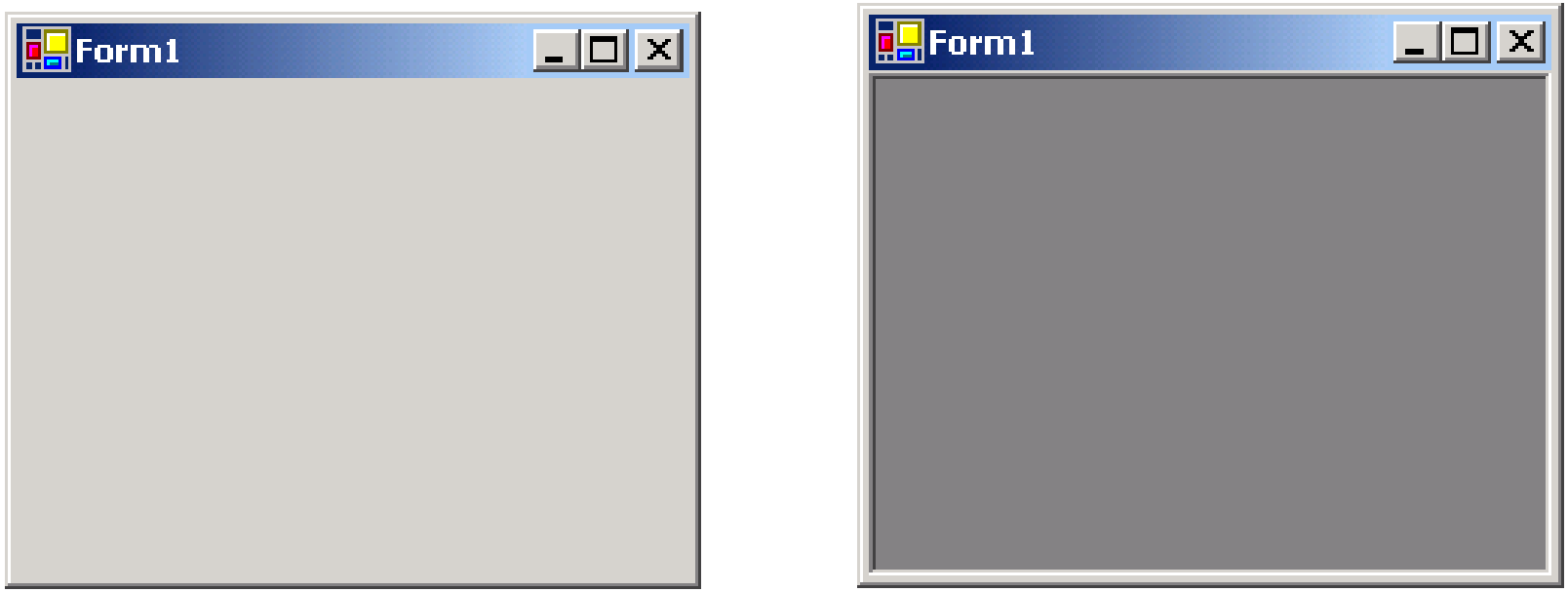
`TileHorizontal`



`TileVertical`

Fig. 13.35 **`LayoutMdi`** enumeration values (Part 2).

13.9 Multiple Document Interface (MDI) Windows



Single Document Interface (SDI)

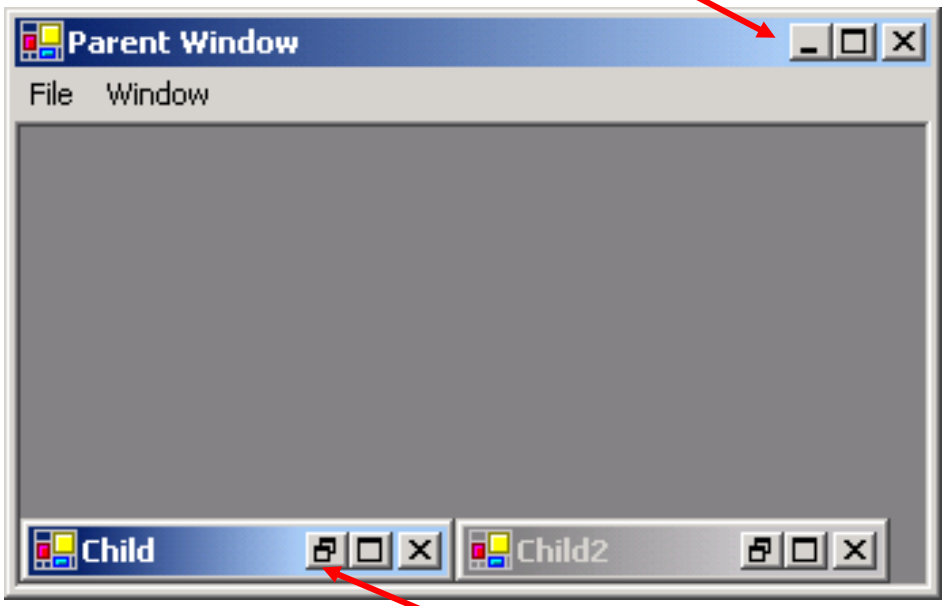
Multiple Document Interface (MDI)

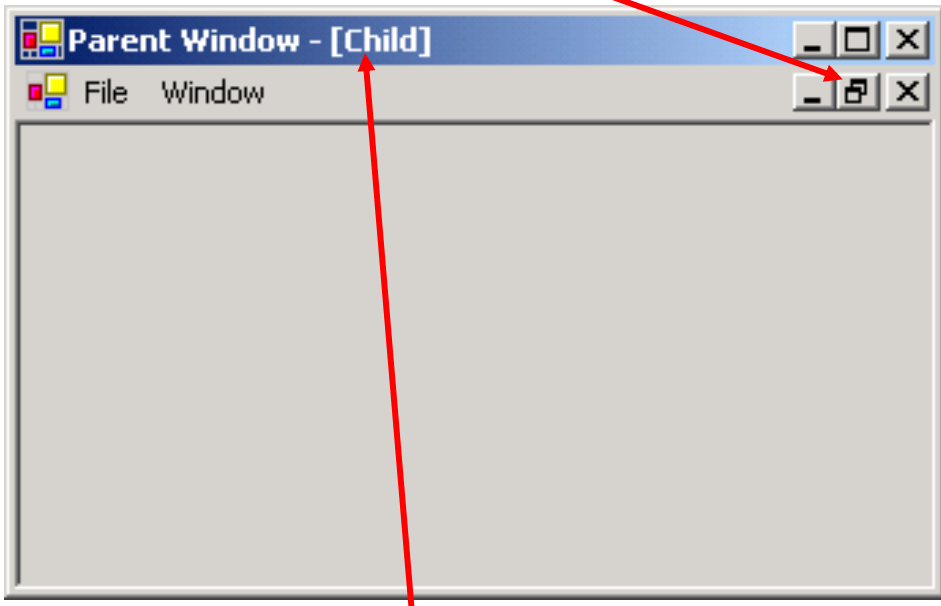
Fig. 13.31 SDI and MDI forms.

13.9 Multiple Document Interface (MDI) Windows

Parent's icons: minimize, maximize and close

Maximized child's icons: minimize, restore and close

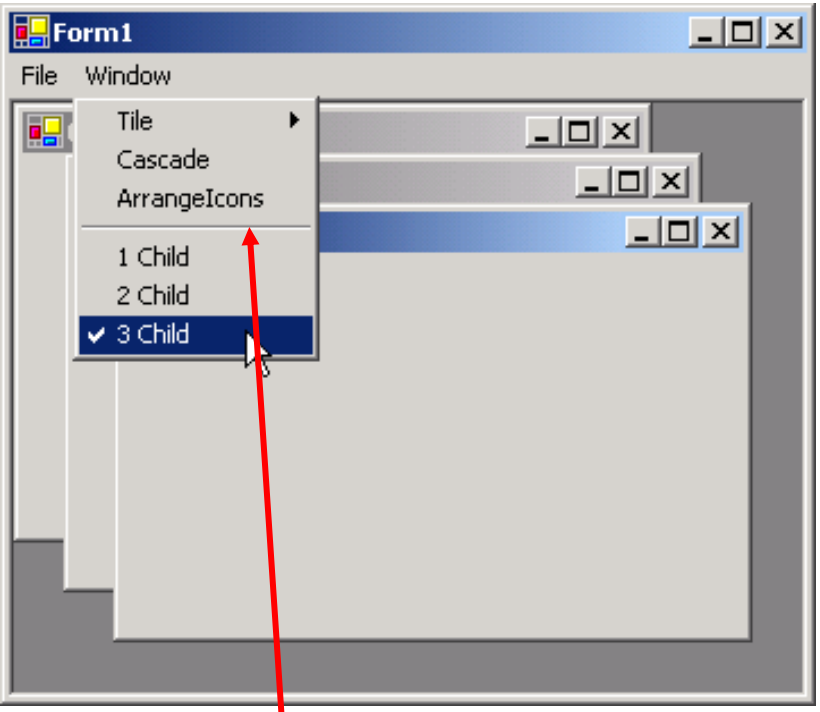


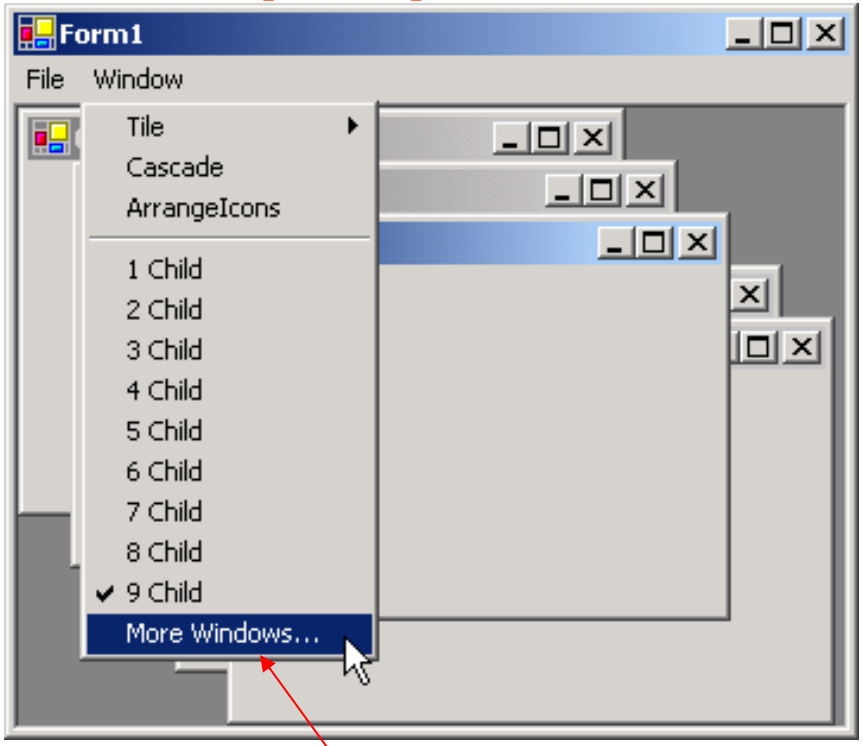


Minimized child's icons: restore, maximize and close

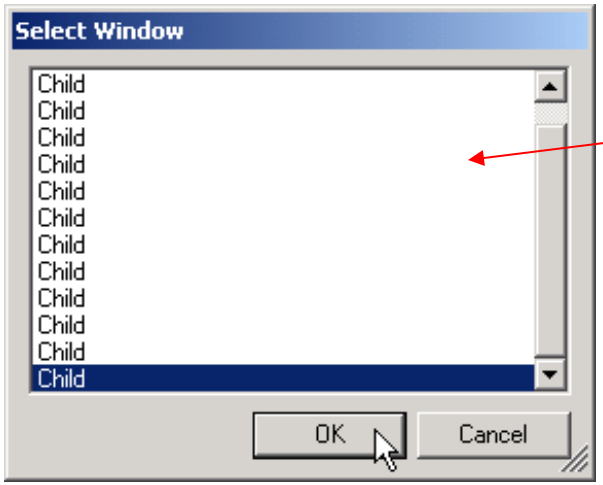
Parent's title bar displays maximized child

13.9 Multiple Document Interface (MDI) Windows





Separator bar and child windows



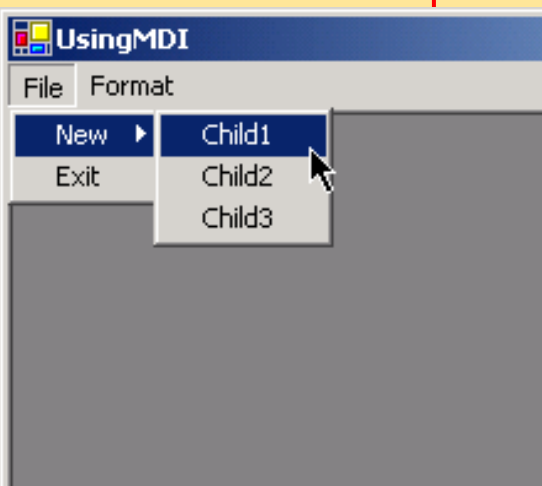
9 or more child windows enables the **More Windows...** option

Child windows list

MDI Form events and properties	Description / Delegate and Event Arguments
Common MDI Child Properties	
`IsMdiChild`	Indicates whether the `Form` is an MDI child. If `True`, `Form` is an MDI child (read-only property).
`MdiParent`	Specifies the MDI parent `Form` of the child.
Common MDI Parent Properties	
`ActiveMdiChild`	Returns the `Form` that is the currently active MDI child (returns `null` if no children are active).
`IsMdiContainer`	Indicates whether a `Form` can be an MDI. If `True`, the `Form` can be an MDI parent. Default is `False`.
`MdiChildren`	Returns the MDI children as an array of `Form`s.
Common Method	
`LayoutMdi`	Determines the display of child forms on an MDI parent. Takes as a parameter an `MdiLayout` enumeration with possible values `ArrangeIcons`, `Cascade`, `TileHorizontal` and `TileVertical`. Figure 13.35 depicts the effects of these values.
Common Event	*(Delegate `EventHandler`; event arguments `EventArgs`)*
`MdiChildActivate`	Generated when an MDI child is closed or activated.

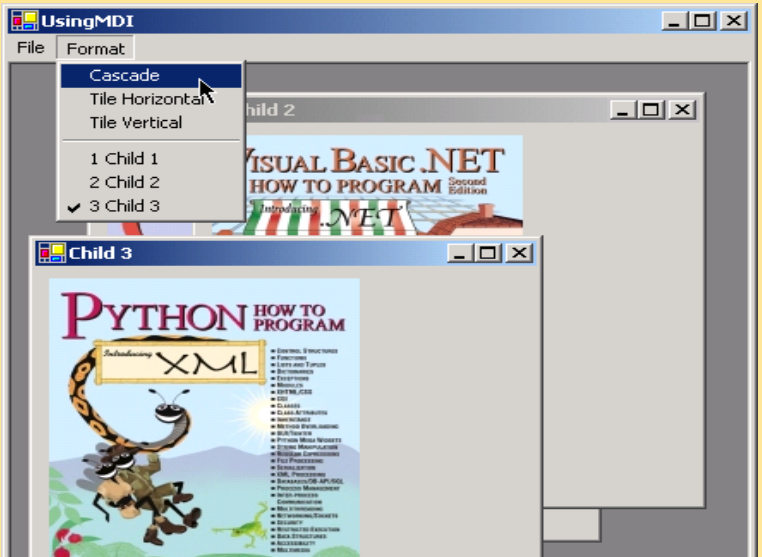
```
3     using System;
4     using System.Drawing;
5     using System.Collections;
6     using System.ComponentModel;
7     using System.Windows.Forms;
8     using System.IO;
10    public class Child : System.Windows.Forms.Form {
12       private System.Windows.Forms.PictureBox pictureBox;
14       public Child( string title, string fileName ) {
17          InitializeComponent();
19          Text = title;
22          pictureBox.Image = Image.FromFile(
23             Directory.GetCurrentDirectory() + fileName );
24       }
25    }
```

```
3     using System;
4     using System.Drawing;
5     using System.Collections;
6     using System.ComponentModel;
7     using System.Windows.Forms;
8     using System.Data;
10    public class UsingMDI : System.Windows.Forms.Form {
12       private System.Windows.Forms.MainMenu mainMenu1;
13       private System.Windows.Forms.MenuItem fileMenuItem;
14       private System.Windows.Forms.MenuItem newMenuItem;
15       private System.Windows.Forms.MenuItem child1MenuItem;
16       private System.Windows.Forms.MenuItem child2MenuItem;
17       private System.Windows.Forms.MenuItem child3MenuItem;
18       private System.Windows.Forms.MenuItem exitMenuItem;
19       private System.Windows.Forms.MenuItem formatMenuItem;
20       private System.Windows.Forms.MenuItem cascadeMenuItem;
21       private System.Windows.Forms.MenuItem
22          tileHorizontalMenuItem;
23       private System.Windows.Forms.MenuItem
24          tileVerticalMenuItem;
26       [STAThread]
27       static void Main() {
29          Application.Run( new UsingMDI() );
30       }
31
```



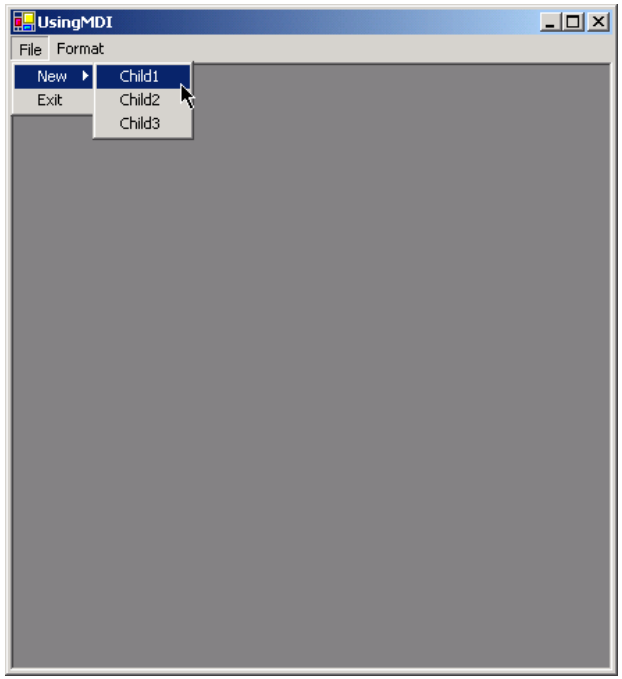
```
33      private void child1MenuItem_Click(
34         object sender, System.EventArgs e ) {
37         Child formChild = new Child( "Child 1",
38            "\\images\\csharphtp1.jpg" );
39         formChild.MdiParent = this;
40         formChild.Show();
41      }
44      private void child2MenuItem_Click(
45         object sender, System.EventArgs e ) {
48         Child formChild = new Child( "Child 2",
49            "\\images\\vbnethtp2.jpg" );
50         formChild.MdiParent = this;
51         formChild.Show();
52      }
55      private void child3MenuItem_Click(
56         object sender, System.EventArgs e ) {
59         Child formChild = new Child( "Child 3",
60            "\\images\\pythonhtp1.jpg" );
61         formChild.MdiParent = this;
62         formChild.Show();
63      }
64
```

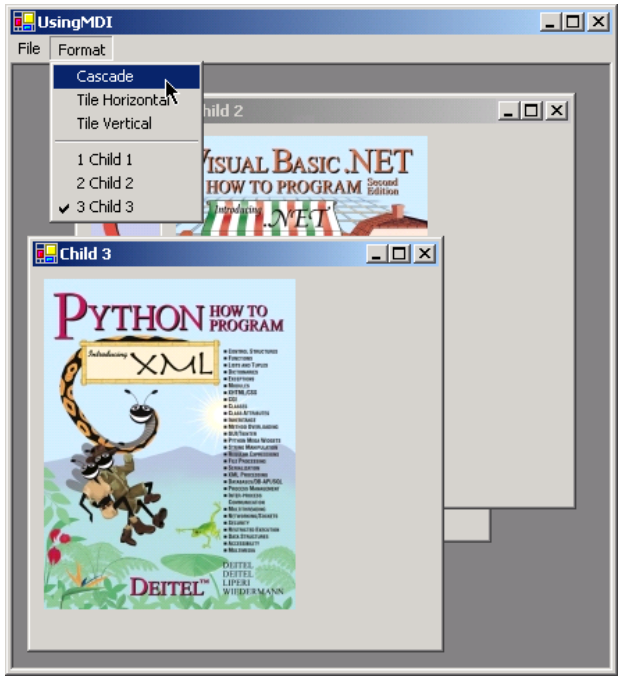
```
66      private void exitMenuItem_Click(
67         object sender, System.EventArgs e ) {
69         Application.Exit();
70      }
73      private void cascadeMenuItem_Click(
74         object sender, System.EventArgs e ) {
76         this.LayoutMdi( MdiLayout.Cascade );
77      }
80      private void tileHorizontalMenuItem_Click(
81         object sender, System.EventArgs e ) {
83         this.LayoutMdi( MdiLayout.TileHorizontal );
84      }
87      private void tileVerticalMenuItem_Click(
88         object sender, System.EventArgs e ){
90         this.LayoutMdi( MdiLayout.TileVertical );
91      }
93   }
```

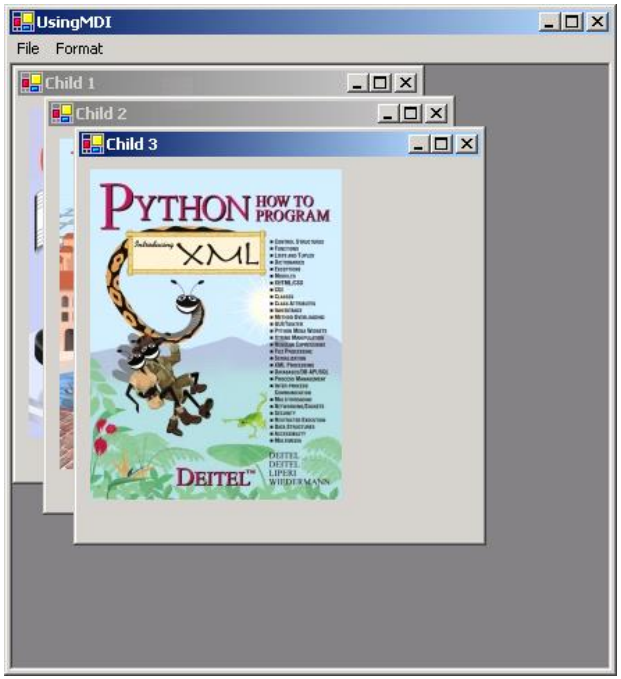


Outline

`UsingMDI.cs`
Program Output





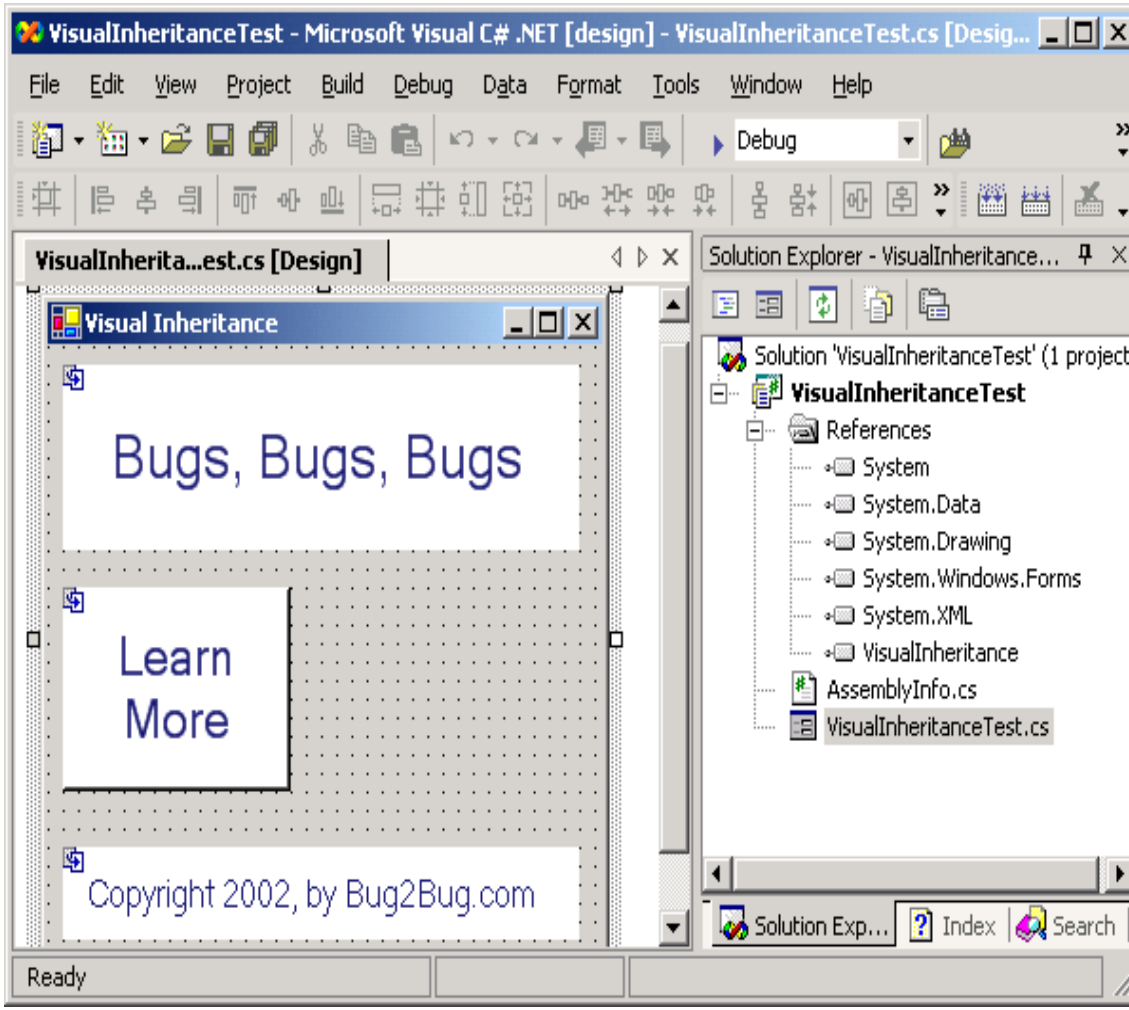


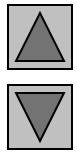
13.10 Visual Inheritance

- Visual inheritance
  - Allowed to create a new **Form** by inheriting from another **Form**
  - The derived **Form** class contains the functionality of its **Form** base class,
    - inherits all visual aspects—such as sizing, component layout, spacing between GUI components, colors and fonts
  - enables developers to achieve visual consistency across applications (principle of least astonishment)
- EX: a company could define a base form that contains a product's logo, a static background color, a predefined menu bar and other elements.
  - Like CSS in web instead

- After designing class **VisualInheritance**,
  - Need to package class **VisualInheritance** in a **.dll**.

1. in **Solution Explorer**, and select **Properties**.
2. In **Common Properties > General**, change the **Output Type** to **Class Library**
3. From the **Project** menu, select **Add Inherited Form...**.
4. Select **Inherited Form** from the templates window **Inheritance Picker**.

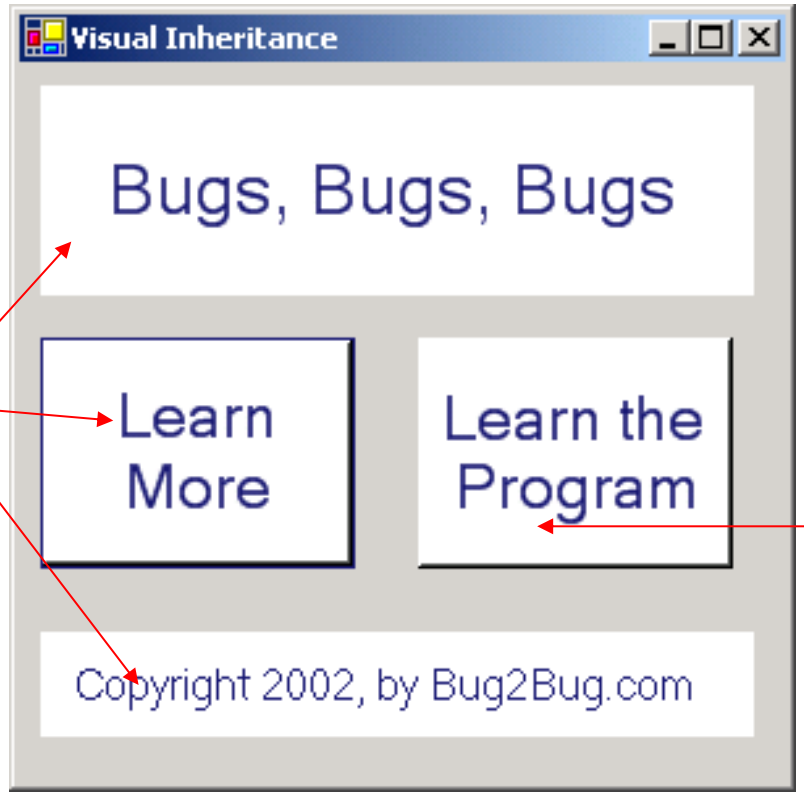




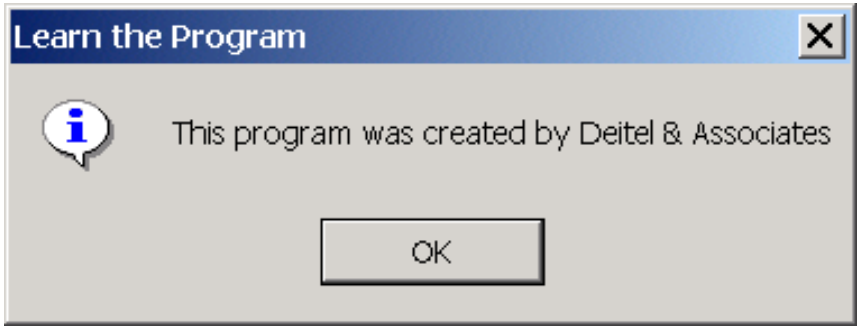
Outline

VisualInheritanceTest.cs
Program Output

Derived class cannot modify these controls



Derived class can modify this control



```
3     using System;
4     using System.Drawing;
5     using System.Collections;
6     using System.ComponentModel;
7     using System.Windows.Forms;
8     using System.Data;
10    public class VisualInheritance : System.Windows.Forms.Form {
12       private System.Windows.Forms.Label bugsLabel;
13       private System.Windows.Forms.Button learnMoreButton;
14       private System.Windows.Forms.Label label1;
16       [STAThread]
17       static void Main(){
19          Application.Run( new VisualInheritance() );
20       }
22       private void learnMoreButton_Click( object sender,
23          System.EventArgs e ) {
25          MessageBox.Show(
26             "Bugs, Bugs, Bugs is a product of Bug2Bug.com",
27             "Learn More", MessageBoxButtons.OK,
28             MessageBoxIcon.Information );
29       }
30    }
```

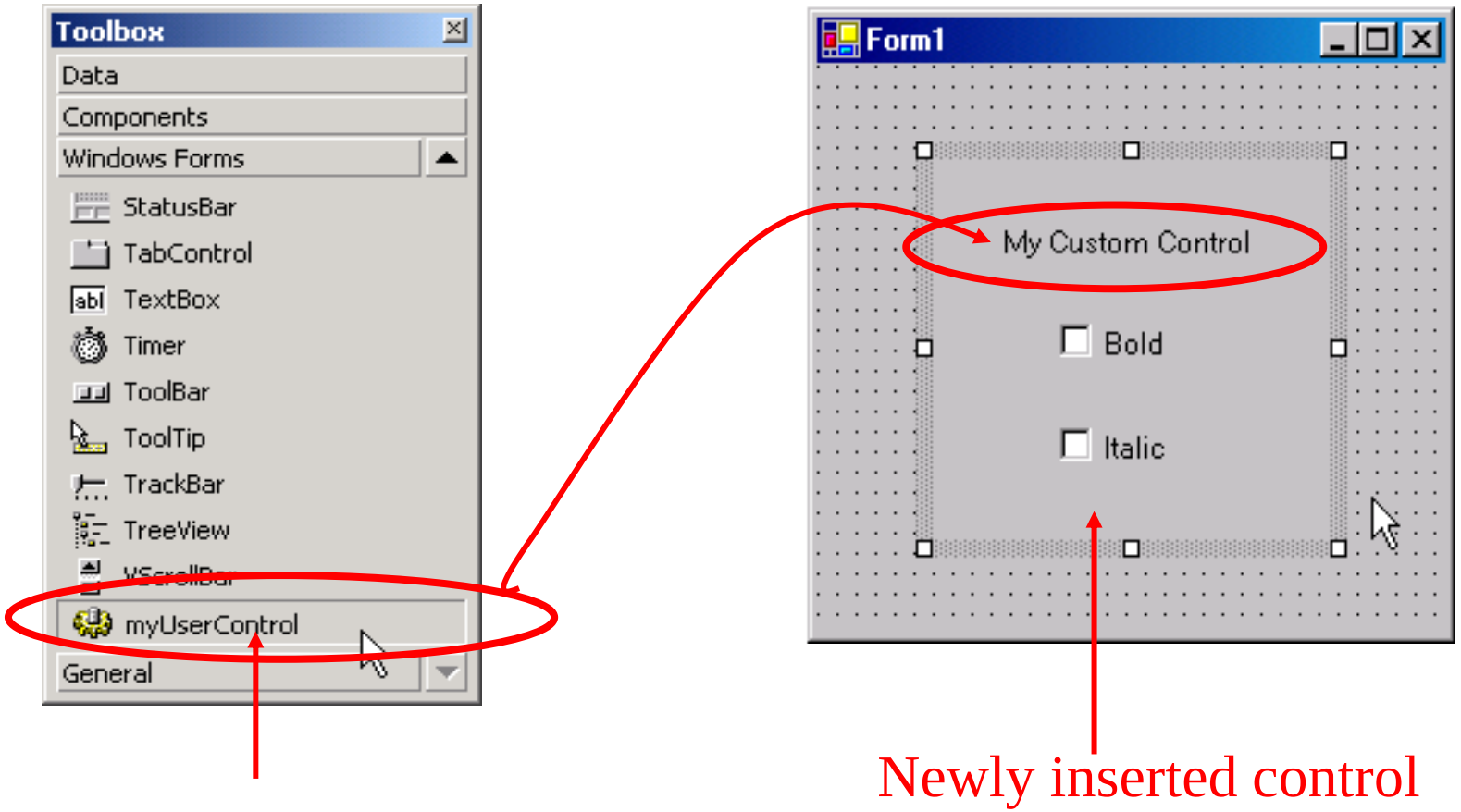
```
3  using System;
4  using System.Collections;
5  using System.ComponentModel;
6  using System.Drawing;
7  using System.Windows.Forms;
9  public class VisualInheritanceTest : VisualInheritance.VisualInheritance {
12    private System.Windows.Forms.Button learnProgramButton;
15    private void learnProgramButton_Click( object sender,  System.EventArgs e ) {
18       MessageBox.Show("This program was created by Deitel & Associates",
20          "Learn the Program", MessageBoxButtons.OK,  MessageBoxIcon.Information );
22    }
24    public static void Main( string[] args ) {
26       Application.Run( new VisualInheritanceTest() );
27    }
28 }
```

VisualInheritance class is a dll file defined by programmer

13.11 User-Defined Controls

- .NET Framework allows programmers to create *custom controls* that inherit from a variety of classes.
  - The controls can appear in the user's **Toolbox**
  - **it** can be added to **Form**s, **Panel**s or **GroupBox**es
- Ex: create a custom control is to derive a class from an existing Windows Forms control, **Label**.
  - can change appearance of a label

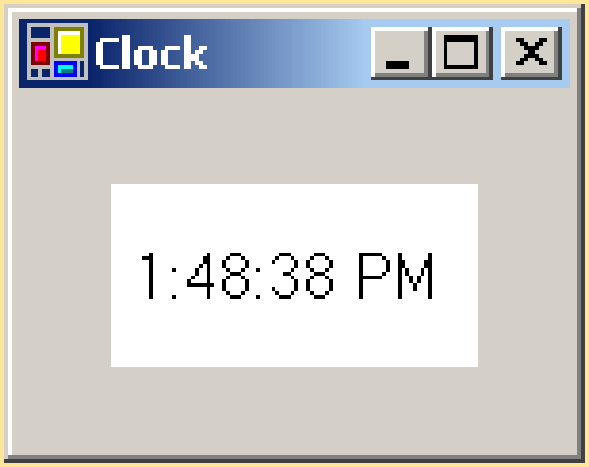
13.11 User-Defined Controls

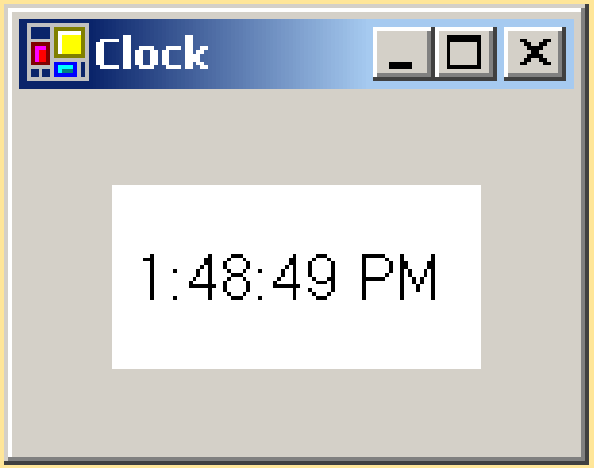


```
4     using System;
5     using System.Collections;
6     using System.ComponentModel;
7     using System.Drawing;
8     using System.Data;
9     using System.Windows.Forms;
11    public class ClockUserControl
12        : System.Windows.Forms.UserControl {
14        private System.Windows.Forms.Timer clockTimer;
15        private System.Windows.Forms.Label displayLabel;
18        private void clockTimer_Tick(
19            object sender, System.EventArgs e ) {
22            displayLabel.Text = DateTime.Now.ToLongTimeString();
24        }
26    }
```

composed of a

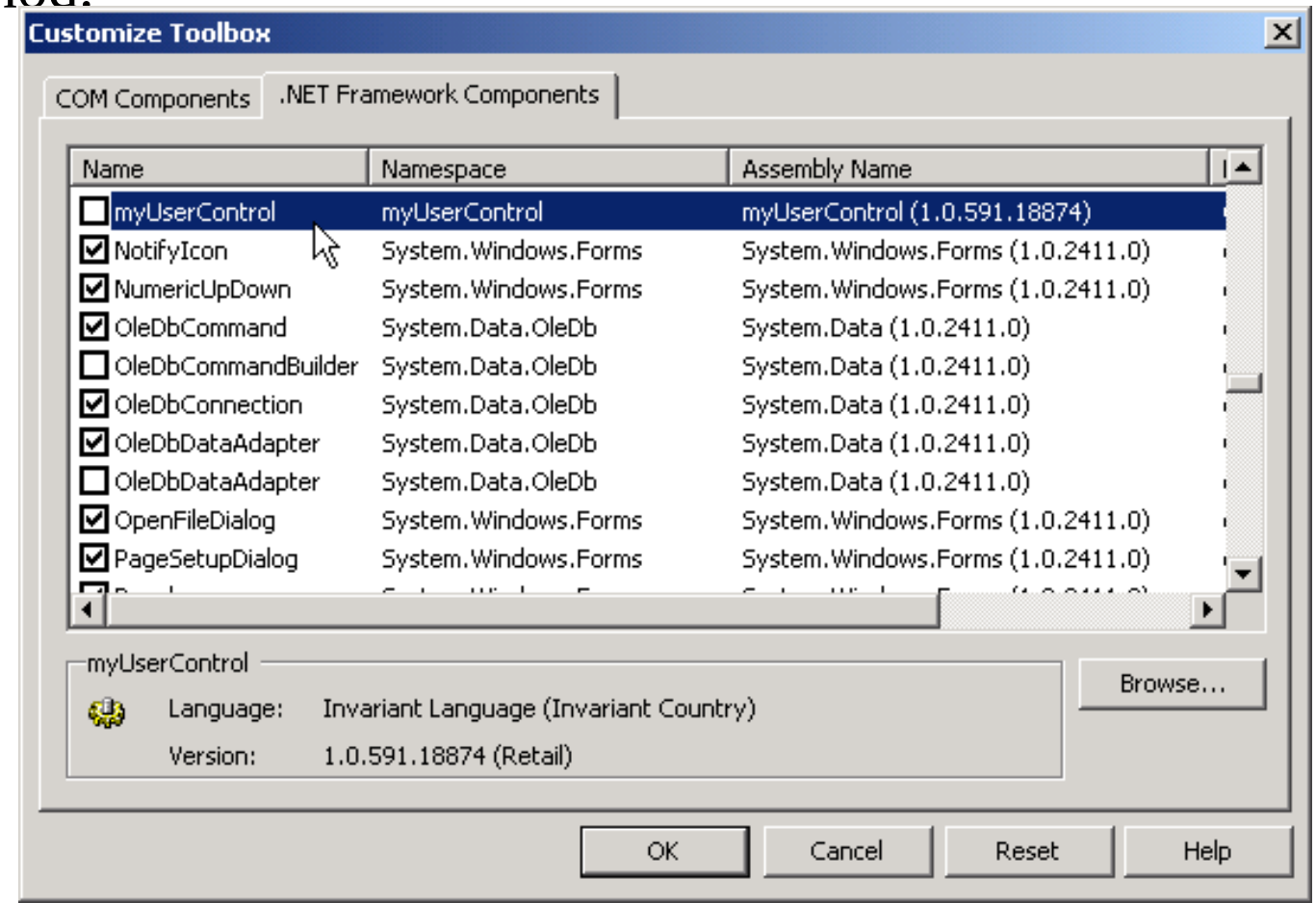
label and a timer





Steps to create a UserControl

1. Create a new **Windows Control Library** project.
2. add controls and functionality to the **UserControl**
3. Build the project to create a **.dll** file for the **UserControl**
   - The file is not executable: **Control** classes do not have a **Main** method.



Steps to use a UserControl

1. Create a new Windows application.
2. click the **ToolBox**, and select **Customize Toolbox...**.
3. Browse for the **.dll** file in output directory for the Windows control library project.
4. Click the checkbox next to the control …