Chapter 13 – Graphical User Interfaces Part 2

Outline

13.1 Introduction

- Continues study of Graphical User Interface
- Explores:
  - Menus
  - `LinkLabels`
  - `ListBox`
  - `CheckedListBox`
  - `ComboBoxes`
  - `TreeView` control
  - Tab controls
  - Multiple-document interface windows

13.2 Menus

- Group related commands together
- Contain:
  - Commands
  - Submenus
- **Exit** uses **`Application`** class to quit
- **Color** options mutually exclusive
- Every option has its own event handler
- Font style options use Xor operator

13.2 Menus

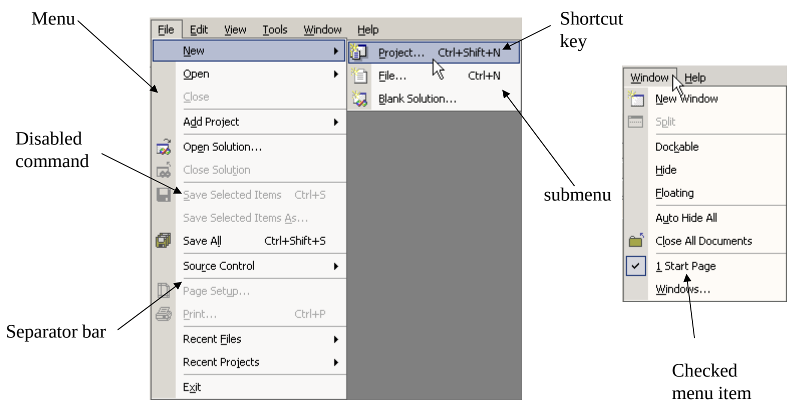


Fig. 13.1 Expanded and checked menus.

13.2 Menus

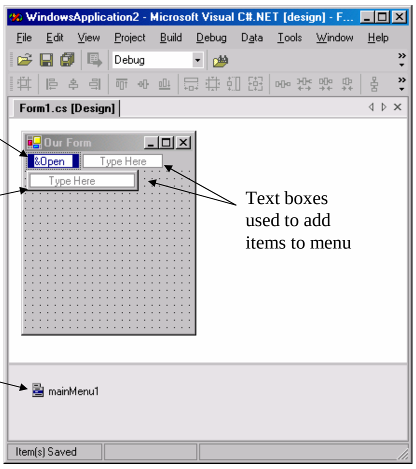

Place & character before the letter to be underlined
Menu Designer
MainMenu icon
Text boxes used to add items to menu
WindowsApplication2 - Microsoft Visual C#.NET [design] - F...
File Edit View Project Build Debug Data Tools Window Help
Debug
Form1.cs [Design]
Our Form
&Open
Type Here
Type Here
mainMenu1
Item(s) Saved

Fig. 13.2 Visual Studio .NET **Menu Designer**.

13.2 Menus

MainMenu and **MenuItem** events and properties	Description / Delegate and Event Arguments
MainMenu ***Properties***	
MenuItems	Collection of **MenuItem**s for the **MainMenu**.
RightToLeft	Used to display text from right to left. Useful for languages that are read from right to left.
MenuItem ***Properties***	
Checked	Whether menu item appears checked (according to property **RadioCheck**). Default **false**, meaning that the menu item is not checked.
Index	Item's position in parent menu.
MenuItems	Collection of submenu items for this menu item.

13.2 Menus

`MergeOrder`	This property sets the position of menu item when parent menu merged with another menu.
`MergeType`	This property takes a value of the **`MenuMerge`** enumeration. Specifies how parent menu merges with another menu. Possible values are **`Add`**, **`MergeItems`**, **`Remove`** and **`Replace`**.
`RadioCheck`	If **`true`**, menu item appears as radio button (black circle) when checked; if **`false`**, menu item displays checkmark. Default **`false`**.
`Shortcut`	Shortcut key for the menu item (i.e. *Ctrl* + *F9* can be equivalent to clicking a specific item).
`ShowShortcut`	If **`true`**, shortcut key shown beside menu item text. Default **`true`**.
Text	Text to appear on menu item. To make an *Alt* access shortcut, precede a character with **&** (i.e. **`&File`** for **F̲ile**).
Common Events	*(Delegate* ***`EventHandler`****, event arguments* ***`EventArgs`****)*
`Click`	Raised when item is clicked or shortcut key is used. Default when double-clicked in designer.

Fig. 13.3 `MainMenu` and `MenuItem` properties and events.

Outline

MenuTest.cs

```
1   // Fig 13.4: MenuTest.cs
2   // Using menus to change font colors and styles.
3
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
11  public class MenuTest : System.Windows.Forms.Form
12  {
13     // display label
14     private System.Windows.Forms.Label displayLabel;
15
16     // main menu (contains file and format menu)
17     private System.Windows.Forms.MainMenu mainMenu;
18
19     // file menu
20     private System.Windows.Forms.MenuItem fileMenuItem;
21     private System.Windows.Forms.MenuItem aboutMenuItem;
22     private System.Windows.Forms.MenuItem exitMenuItem;
23
24     // format menu
25     private System.Windows.Forms.MenuItem formatMenuItem;
26
27     // color submenu
28     private System.Windows.Forms.MenuItem colorMenuItem;
29     private System.Windows.Forms.MenuItem blackMenuItem;
30     private System.Windows.Forms.MenuItem blueMenuItem;
31     private System.Windows.Forms.MenuItem redMenuItem;
32     private System.Windows.Forms.MenuItem greenMenuItem;
33
```

About command
Exit command

Color options

Outline

MenuTest.cs

```
      // font submenu
      private System.Windows.Forms.MenuItem timesMenuItem;
      private System.Windows.Forms.MenuItem courierMenuItem;
      private System.Windows.Forms.MenuItem comicMenuItem;
      private System.Windows.Forms.MenuItem boldMenuItem;
      private System.Windows.Forms.MenuItem italicMenuItem;
      private System.Windows.Forms.MenuItem fontMenuItem;

      private System.Windows.Forms.MenuItem separatorMenuItem;

      [STAThread]
      static void Main()
      {
         Application.Run( new MenuTest() );
      }

      // display MessageBox
      private void aboutMenuItem_Click(
         object sender, System.EventArgs e )
      {
         MessageBox.Show(
            "This is an example\nof using menus.",
            "About", MessageBoxButtons.OK,
            MessageBoxIcon.Information );
      }

      // exit program
      private void exitMenuItem_Click(
         object sender, System.EventArgs e )
      {
         Application.Exit();
      }

```

Font options

Style options

About event handler

Exit event Handler

Outline

MenuTest.cs

```
      // reset color
      private void ClearColor()
      {
         // clear all checkmarks
         blackMenuItem.Checked = false;
         blueMenuItem.Checked = false;
         redMenuItem.Checked = false;
         greenMenuItem.Checked = false;
      }

      // update menu state and color display black
      private void blackMenuItem_Click(
         object sender, System.EventArgs e )
      {
         // reset checkmarks for color menu items
         ClearColor();

         // set color to black
         displayLabel.ForeColor = Color.Black;
         blackMenuItem.Checked = true;
      }

      // update menu state and color display blue
      private void blueMenuItem_Click(
         object sender, System.EventArgs e )
      {
         // reset checkmarks for color menu items
         ClearColor();

         // set color to blue
         displayLabel.ForeColor = Color.Blue;
         blueMenuItem.Checked = true;
      }

```

Black event handler

Blue event Handler

Outline

MenuTest.cs

```
      // update menu state and color display red
      private void redMenuItem_Click(
         object sender, System.EventArgs e )
      {
         // reset checkmarks for color menu items
         ClearColor();

         // set color to red
         displayLabel.ForeColor = Color.Red;
         redMenuItem.Checked = true;
      }

      // update menu state and color display green
      private void greenMenuItem_Click(
         object sender, System.EventArgs e )
      {
         // reset checkmarks for color menu items
         ClearColor();

         // set color to green
         displayLabel.ForeColor = Color.Green;
         greenMenuItem.Checked = true;
      }

      // reset font types
      private void ClearFont()
      {
         // clear all checkmarks
         timesMenuItem.Checked = false;
         courierMenuItem.Checked = false;
         comicMenuItem.Checked = false;
      }

```

Red event handler

Green event handler

Outline

nuTest.cs

```
      // update menu state and set font to Times
      private void timesMenuItem_Click(
         object sender, System.EventArgs e )
      {
         // reset checkmarks for font menu items
         ClearFont();

         // set Times New Roman font
         timesMenuItem.Checked = true;
         displayLabel.Font = new Font(
            "Times New Roman", 14, displayLabel.Font.Style );
      }

      // update menu state and set font to Courier
      private void courierMenuItem_Click(
         object sender, System.EventArgs e )
      {
         // reset checkmarks for font menu items
         ClearFont();

         // set Courier font
         courierMenuItem.Checked = true;
         displayLabel.Font = new Font(
            "Courier New", 14, displayLabel.Font.Style );
      }

      // update menu state and set font to Comic Sans MS
      private void comicMenuItem_Click(
         object sender, System.EventArgs e )
      {
         // reset checkmarks for font menu items
         ClearFont();

```

Times New Roman event handler

Courier New event handler

Comic Sans event handler

Outline

MenuTest.cs

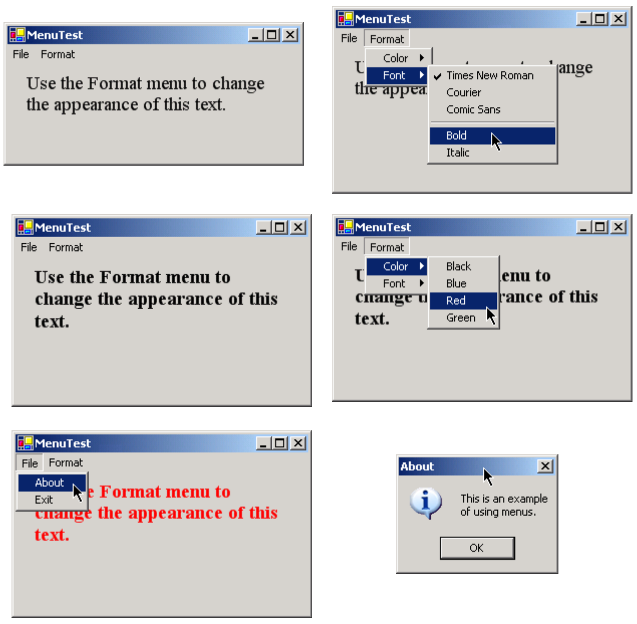
```
      // set Comic Sans font
      comicMenuItem.Checked = true;
      displayLabel.Font = new Font(
         "Comic Sans MS", 14, displayLabel.Font.Style );
   }

   // toggle checkmark and toggle bold style
   private void boldMenuItem_Click(
      object sender, System.EventArgs e )
   {
      // toggle checkmark
      boldMenuItem.Checked = !boldMenuItem.Checked;

      // use Xor to toggle bold, keep all other styles
      displayLabel.Font = new Font(
         displayLabel.Font.FontFamily, 14,
         displayLabel.Font.Style ^ FontStyle.Bold );
   }

   // toggle checkmark and toggle italic style
   private void italicMenuItem_Click(
      object sender, System.EventArgs e)
   {
      // toggle checkmark
      italicMenuItem.Checked = !italicMenuItem.Checked;

      // use Xor to toggle bold, keep all other styles
      displayLabel.Font = new Font(
         displayLabel.Font.FontFamily, 14,
         displayLabel.Font.Style ^ FontStyle.Italic );
   }

} // end class MenuTest
```

Bold event handler

Italic event handler

Outline

MenuTest.cs
Program Output



```
Use the Format menu to change the appearance of this text.
```

```
About
This is an example of using menus.
```

13.3 LinkLabels

- Displays links to other objects
  - Uses **event handlers** to link to right file or program
  - **Start** method of **Process** class opens other programs
- Derived from class **Label**, inherits functionality

13.3 LinkLabels

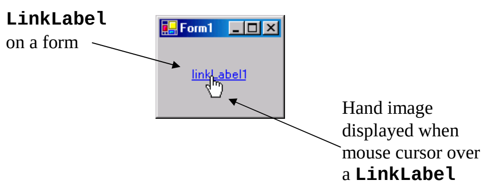


Fig. 13.5 **LinkLabel** control in the design phase and in running program.

13.3 LinkLabels

LinkLabel properties and events	Description / Delegate and Event Arguments
Common Properties	
ActiveLinkColor	Specifies the color of the active link when clicked. Default is red.
LinkArea	Specifies which portion of text in the **LinkLabel** is treated as part of the link.
LinkBehavior	Specifies the link's behavior, such as how the link appears when the mouse is placed over it.
LinkColor	Specifies the original color of all links before they have been visited. Default is blue.
Links	Lists the **LinkLabel.Link** objects, which are the links contained in the **LinkLabel**.
LinkVisited	If **True**, link appears as if it were visited (its color is changed to that specified by property **VisitedLinkColor**). Default **False**.
Text	Specifies the text to appear on the control.
UseMnemonic	If **True**, **&** character in **Text** property acts as a shortcut (similar to the *Alt* shortcut in menus).
VisitedLinkColor	Specifies the color of visited links. Default is **Color.Purple**.
Common Event	*(Delegate* ***LinkLabelLinkClickedEventHandler****, event arguments* ***LinkLabelLinkClickedEventArgs****)*
LinkClicked	Generated when link is clicked. Default when control is double-clicked in designer.

Outline

LinkLabelTest.cs

```
// Fig. 13.7: LinkLabelTest.cs
// Using LinkLabels to create hyperlinks.

using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;

public class LinkLabelTest : System.Windows.Forms.Form
{
   // linklabels to C: drive, www.deitel.com and Notepad
   private System.Windows.Forms.LinkLabel driveLinkLabel;
   private System.Windows.Forms.LinkLabel deitelLinkLabel;
   private System.Windows.Forms.LinkLabel notepadLinkLabel;

   [STAThread]
   static void Main()
   {
      Application.Run( new LinkLabelTest() );
   }

   // browse C:\ drive
   private void driveLinkLabel_LinkClicked( object sender,
      System.Windows.Forms.LinkLabelLinkClickedEventArgs e )
   {
      driveLinkLabel.LinkVisited = true;
      System.Diagnostics.Process.Start( "C:\\" );
   }

```

C drive link

Deitel website link

Note

C drive event handler

Start method to open other programs

Outline

belTest.cs

```
   // load www.deitel.com in Web broswer
   private void deitelLinkLabel_LinkClicked( object sender,
      System.Windows.Forms.LinkLabelLinkClickedEventArgs e )
   {
      deitelLinkLabel.LinkVisited = true;
      System.Diagnostics.Process.Start(
         "IExplore", "http://www.deitel.com" );
   }

   // run application Notepad
   private void notepadLinkLabel_LinkClicked(
      object sender,
      System.Windows.Forms.LinkLabelLinkClickedEventArgs e )
   {
      notepadLinkLabel.LinkVisited = true;

      // program called as if in run
      // menu and full path not needed
      System.Diagnostics.Process.Start( "notepad" );
   }

} // end class LinkLabelTest
```

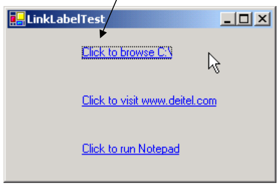
Deitel website event handler

Notepad event handler

Outline

LinkLabelTest.cs
Program Output

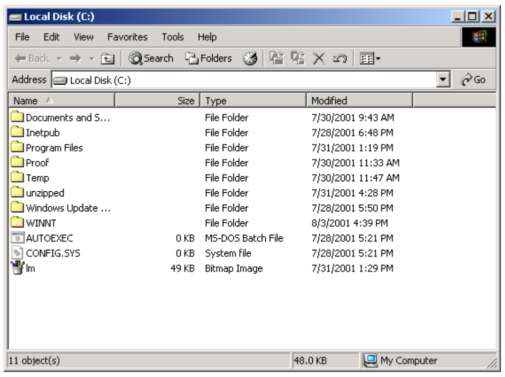
Click on first **LinkLabel** to look at contents of C drive


LinkLabelTest

Click to browse C:\

Click to visit www.deitel.com

Click to run Notepad


Local Disk (C:)

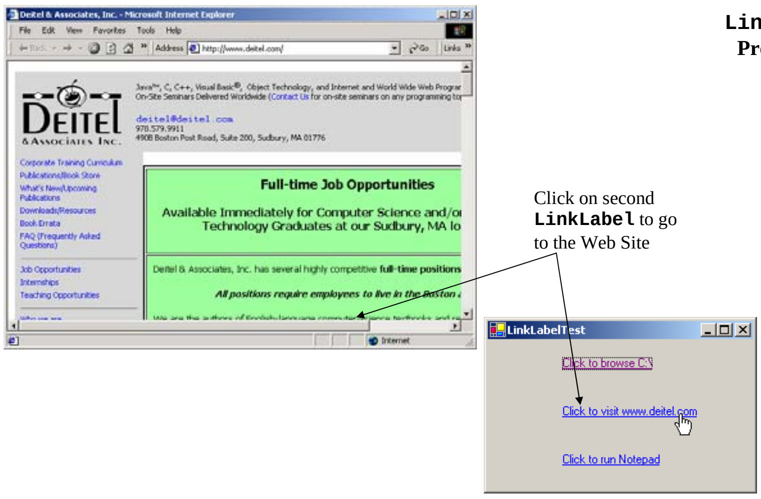
Address: Local Disk (C:)

Name	Size	Type	Modified
Documents and S...		File Folder	7/30/2001 9:43 AM
Inetpub		File Folder	7/28/2001 6:48 PM
Program Files		File Folder	7/31/2001 1:19 PM
Proof		File Folder	7/30/2001 11:33 AM
Temp		File Folder	7/30/2001 11:47 AM
unzipped		File Folder	7/31/2001 4:28 PM
Windows Update ...		File Folder	7/28/2001 5:50 PM
WINNT		File Folder	8/3/2001 4:39 PM
AUTOEXEC	0 KB	MS-DOS Batch File	7/28/2001 5:21 PM
CONFIG.SYS	0 KB	System file	7/28/2001 5:21 PM
lm	49 KB	Bitmap Image	7/31/2001 1:29 PM

11 object(s) | 48.0 KB | My Computer

Outline

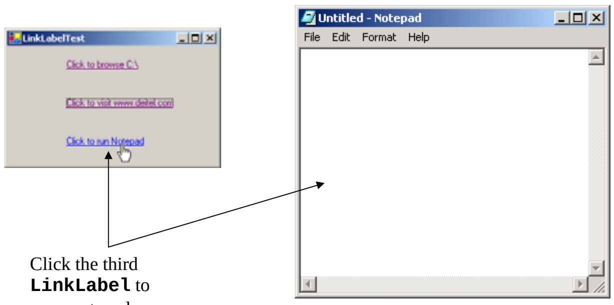
`LinkLabelTest.cs`
Program Output


Deitel & Associates, Inc. - Microsoft Internet Explorer
http://www.deitel.com/
Full-time Job Opportunities
LinkLabelTest
Click to browse C:\
Click to visit www.deitel.com
Click to run Notepad

Click on second **`LinkLabel`** to go to the Web Site

Outline

`LinkLabelTest.cs`
Program Output



Click the third **`LinkLabel`** to open notepad

13.4 ListBoxes and CheckedListBoxes

- **ListBoxes**
  - Allow users to view and select from items on a list
  - Static objects
  - **SelectionMode** property determines number of items that can be selected
  - Property **Items** returns all objects in list
  - Property **SelectedItem** returns current selected item
  - Property **SelectedIndex** returns index of selected item
  - Property **GetSelected** returns true if property at given index is selected
  - Use **Add** method to add to **Items** collection
    - myListBox.**Items.Add**(“myListItem”)

13.4 ListBoxes and CheckedListBoxes

- **CheckedListBoxes**
  - Extends **ListBox** by placing check boxes next to items
  - Can select more than one object at one time

13.4.1 `ListBoxes`

- Class **`ListBoxTest`**
  - Allows users to add and remove items from **`ListBox`**
  - Uses event handlers to add to, remove from and clear list

13.4.2 `CheckedListBoxes`

- **`CheckedListBox`** derives from class **`ListBox`**
  - Can add to, remove from or clear list
  - Can select multiple items from the list
  - Properties **`CurrentValue`** and **`NewValue`** return state of object selected
  - Properties **`CheckedItems`** and **`CheckedIndices`** return the objects and indices of selected items respectively

13.4 ListBoxes and CheckListBoxes

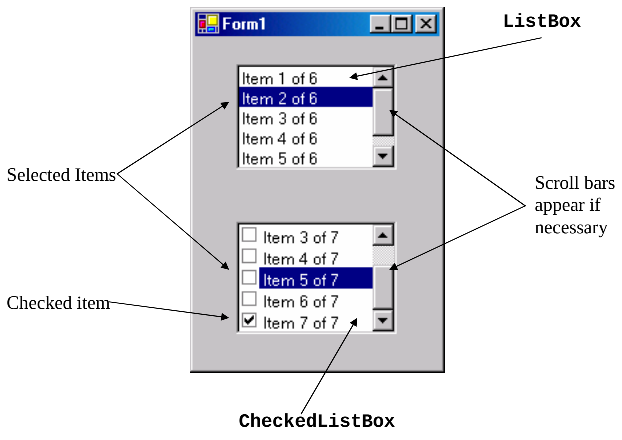
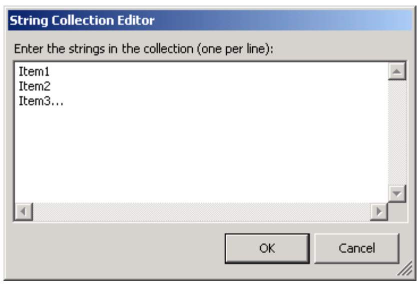


Fig. 13.8 `ListBox` and `CheckedListBox` on a form.

13.4 ListBoxes and CheckListBoxes

`ListBox` properties, methods and events	Description / Delegate and Event Arguments
Common Properties	
`Items`	Lists the collection of items within the **`ListBox`**.
`MultiColumn`	Indicates whether the **`ListBox`** can break a list into multiple columns. Multiple columns are used to make vertical scroll bars unnecessary.
`SelectedIndex`	Returns the index of the currently selected item. If the user selects multiple items, this method arbitrarily returns one of the selected indices; if no items have been selected, the method returns **`-1`**.
`SelectedIndices`	Returns a collection of the indices of all currently selected items.
`SelectedItem`	Returns a reference to the currently selected item (if multiple items are selected, it returns the item with the lowest index number).
`SelectedItems`	Returns a collection of the currently selected item(s).
`SelectionMode`	Determines the number of items that can be selected and the means through which multiple items can be selected. Values **`None`**, **`One`**, **`MultiSimple`** (multiple selection allowed) and **`MultiExtended`** (multiple selection allowed via a combination of arrow keys, mouse clicks and *Shift* and *Control* buttons).
`Sorted`	Indicates whether items appear in alphabetical order. **`True`** causes alphabetization; default is **`False`**.
Common Method	
`GetSelected`	Takes an index, and returns **`True`** if the corresponding item is selected.

13.4 ListBoxes and CheckListBoxes



String Collection Editor

Enter the strings in the collection (one per line):

```
Item1
Item2
Item3...
```

Fig. 13.10 **String Collection Editor**.

Outline

ListBoxTest.cs

```
// Fig 13.11: ListBoxTest.cs
// Program to add, remove and clear list box items.

using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;

public class ListBoxTest : System.Windows.Forms.Form
{
   // contains user-input list of elements
   private System.Windows.Forms.ListBox displayListBox;

   // user input textbox
   private System.Windows.Forms.TextBox inputTextBox;

   // add, remove, clear and exit command buttons
   private System.Windows.Forms.Button addButton;
   private System.Windows.Forms.Button removeButton;
   private System.Windows.Forms.Button clearButton;
   private System.Windows.Forms.Button exitButton;

   [STAThread]
   static void Main()
   {
      Application.Run( new ListBoxTest() );
   }

```

Display ListBox

Text field for input

Add button

Clear button

n

Exit button

Outline

ListBoxTest.cs

```
      // add new item (text from input box)
      // and clear input box
      private void addButton_Click(
         object sender, System.EventArgs e )
      {
         displayListBox.Items.Add( inputTextBox.Text );
         inputTextBox.Clear();
      }

      // remove item if one selected
      private void removeButton_Click(
         object sender, System.EventArgs e )
      {
         // remove only if item selected
         if ( displayListBox.SelectedIndex != -1 )
            displayListBox.Items.RemoveAt(
               displayListBox.SelectedIndex );
      }

      // clear all items
      private void clearButton_Click(
         object sender, System.EventArgs e )
      {
         displayListBox.Items.Clear();
      }

      // exit application
      private void exitButton_Click(
         object sender, System.EventArgs e )
      {
         Application.Exit();
      }

   } // end class ListBoxTest
```

Add event handler

Add method

Test if item is selected

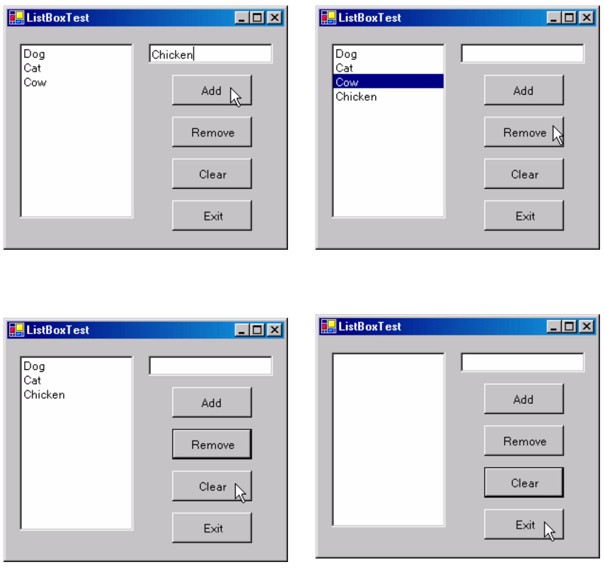
Remove method

Clear method

Exit

Outline

ListBoxTest.cs
Program Output



13.4 ListBoxes and CheckListBoxes

CheckedListBox properties, methods and events	Description / Delegate and Event Arguments
Common Properties	*(All the **ListBox** properties and events are inherited by **CheckedListBox**.)*
CheckedItems	The collection of items that are checked. Not the same as the selected items, which are highlighted (but not necessarily checked).
CheckedIndices	Returns indices for the items that are checked. Not the same as the selected indices.
SelectionMode	Can only have values **One** (allows multiple selection) or **None** (does not allow multiple selection).
Common Methods	
GetItemChecked	Takes an index and returns **true** if corresponding item checked.
Common Events	*(Delegate **ItemCheckEventHandler**, event arguments **ItemCheckEventArgs**)*
ItemCheck	Raised when an item is checked or unchecked.
ItemCheckEventArgs *Properties*	
CurrentValue	Whether current item is checked or unchecked. Values **Checked**, **Unchecked** or **Indeterminate**.
Index	Index of item that changed.
NewValue	New state of item.

Fig. 13.12 **CheckedListBox** properties, methods and events.

Outline

CheckedListBoxTest.cs

```
// Fig. 13.13: CheckedListBoxTest.cs
// Using the checked list boxes to add items to a list box

using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;

public class CheckedListBoxTest : System.Windows.Forms.Form
{
   // list of available book titles
   private System.Windows.Forms.CheckedListBox
      inputCheckedListBox;

   // user selection list
   private System.Windows.Forms.ListBox displayListBox;

   [STAThread]
   static void Main()
   {
      Application.Run( new CheckedListBoxTest() );
   }

   // item about to change,
   // add or remove from displayListBox
   private void inputCheckedListBox_ItemCheck(
      object sender,
      System.Windows.Forms.ItemCheckEventArgs e )
   {
      // obtain reference of selected item
      string item =
         inputCheckedListBox.SelectedItem.ToString();

```

CheckedListBox

ListBox

ItemCheck event handler

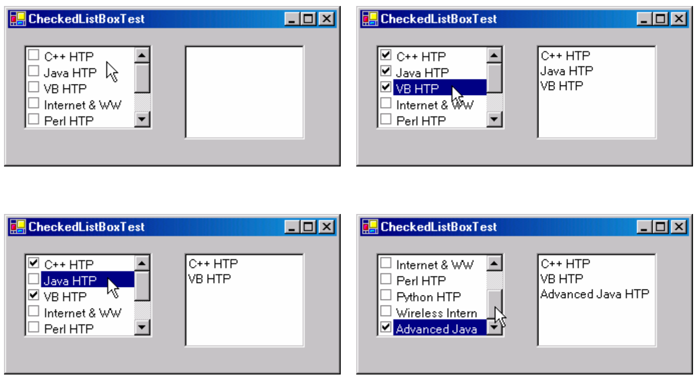
CheckedListBoxTest.cs

```
         // if item checked add to listbox
         // otherwise remove from listbox
         if ( e.NewValue == CheckState.Checked )
            displayListBox.Items.Add( item );
         else
            displayListBox.Items.Remove( item );

      } // end method inputCheckedListBox_Click

   } // end class CheckedListBox
```

Add Item

Remove Item

Program Output



13.5 ComboBoxes

- Combine **`TextBox`** and drop-down list
- **`Add`** method adds object to collection
- Properties:
  - **`DropDownStyle:`** determines type of **`ComboBox`**
  - **`Items:`** returns objects in the list
  - **`SelectedItem`**: returns object selected
  - **`SelectedIndex`**: returns index of selected item

13.5 ComboBoxes

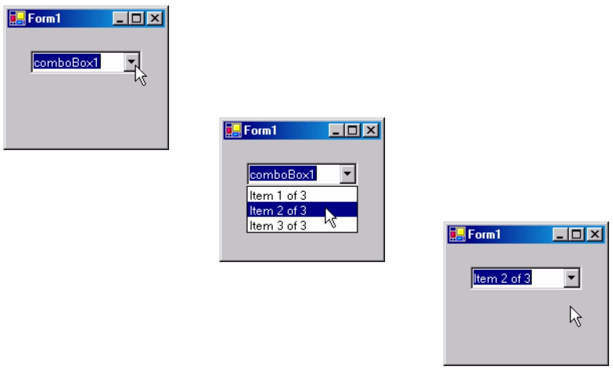
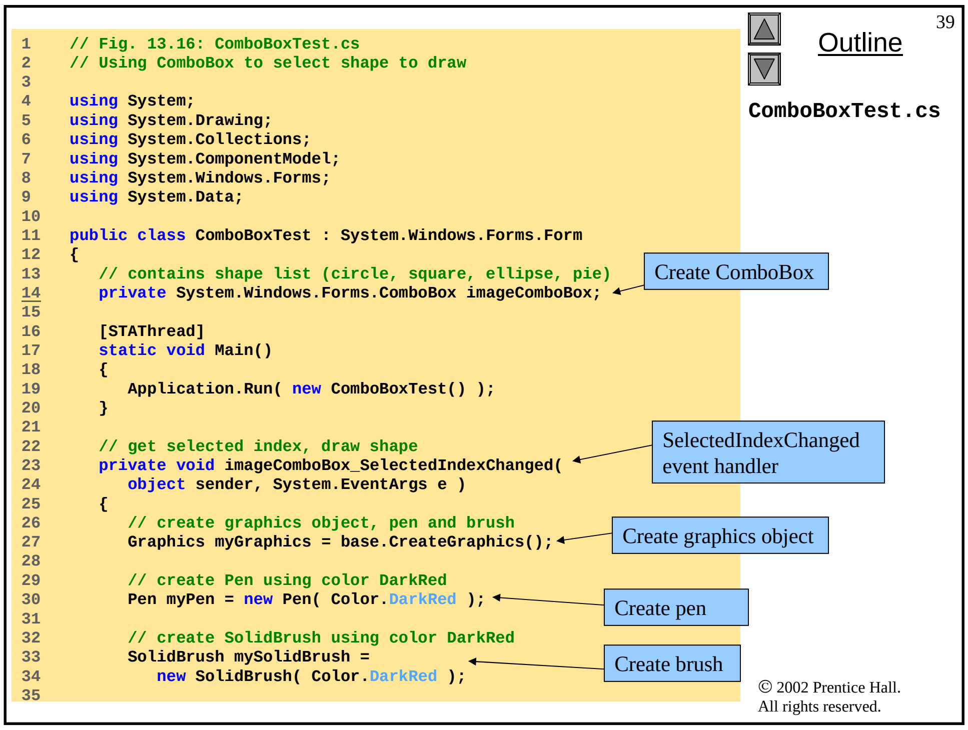


Fig. 13.14 Demonstrating a **ComboBox**.

13.5 ComboBoxes

ComboBox events and properties	Description / Delegate and Event Arguments
Common *Properties*	
DropDownStyle	Determines the type of combo box. Value **Simple** means that the text portion is editable and the list portion is always visible. Value **DropDown** (the default) means that the text portion is editable but an arrow button must be clicked to see the list portion. Value **DropDownList** means that the text portion is not editable and the arrow button must be clicked to see the list portion.
Items	Collection of items in the **ComboBox** control.
MaxDropDownItems	Maximum number of items to display in the drop-down list (between **1** and **100**). If value is exceeded, a scroll bar appears.
SelectedIndex	Returns index of currently selected item. If there is no currently selected item, **-1** is returned.
SelectedItem	Returns reference to currently selected item.
Sorted	If **true**, items appear in alphabetical order. Default **false**.
Common Events	*(Delegate **EventHandler**, event arguments **EventArgs**)*
SelectedIndexChanged	Raised when selected index changes (i.e., a check box has been checked or unchecked). Default when control double-clicked in designer.

Fig. 13.15 **ComboBox** properties and events.



Outline

ComboBoxTest.cs

```
// Fig. 13.16: ComboBoxTest.cs
// Using ComboBox to select shape to draw

using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;

public class ComboBoxTest : System.Windows.Forms.Form
{
   // contains shape list (circle, square, ellipse, pie)
   private System.Windows.Forms.ComboBox imageComboBox;

   [STAThread]
   static void Main()
   {
      Application.Run( new ComboBoxTest() );
   }

   // get selected index, draw shape
   private void imageComboBox_SelectedIndexChanged(
      object sender, System.EventArgs e )
   {
      // create graphics object, pen and brush
      Graphics myGraphics = base.CreateGraphics();

      // create Pen using color DarkRed
      Pen myPen = new Pen( Color.DarkRed );

      // create SolidBrush using color DarkRed
      SolidBrush mySolidBrush =
         new SolidBrush( Color.DarkRed );

```

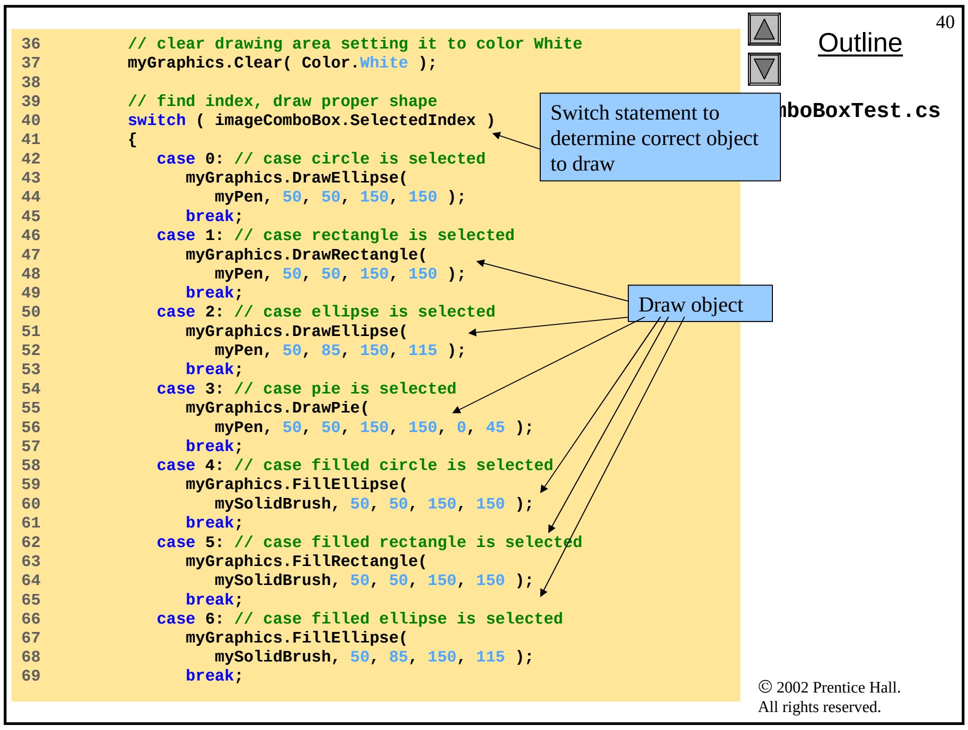
Create ComboBox

SelectedIndexChanged event handler

Create graphics object

Create pen

Create brush



Outline

boBoxTest.cs

```
      // clear drawing area setting it to color White
      myGraphics.Clear( Color.White );

      // find index, draw proper shape
      switch ( imageComboBox.SelectedIndex )
      {
         case 0: // case circle is selected
            myGraphics.DrawEllipse(
               myPen, 50, 50, 150, 150 );
            break;
         case 1: // case rectangle is selected
            myGraphics.DrawRectangle(
               myPen, 50, 50, 150, 150 );
            break;
         case 2: // case ellipse is selected
            myGraphics.DrawEllipse(
               myPen, 50, 85, 150, 115 );
            break;
         case 3: // case pie is selected
            myGraphics.DrawPie(
               myPen, 50, 50, 150, 150, 0, 45 );
            break;
         case 4: // case filled circle is selected
            myGraphics.FillEllipse(
               mySolidBrush, 50, 50, 150, 150 );
            break;
         case 5: // case filled rectangle is selected
            myGraphics.FillRectangle(
               mySolidBrush, 50, 50, 150, 150 );
            break;
         case 6: // case filled ellipse is selected
            myGraphics.FillEllipse(
               mySolidBrush, 50, 85, 150, 115 );
            break;
```

Switch statement to determine correct object to draw

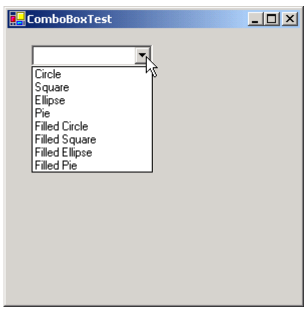
Draw object

Outline

ComboBoxTest.cs

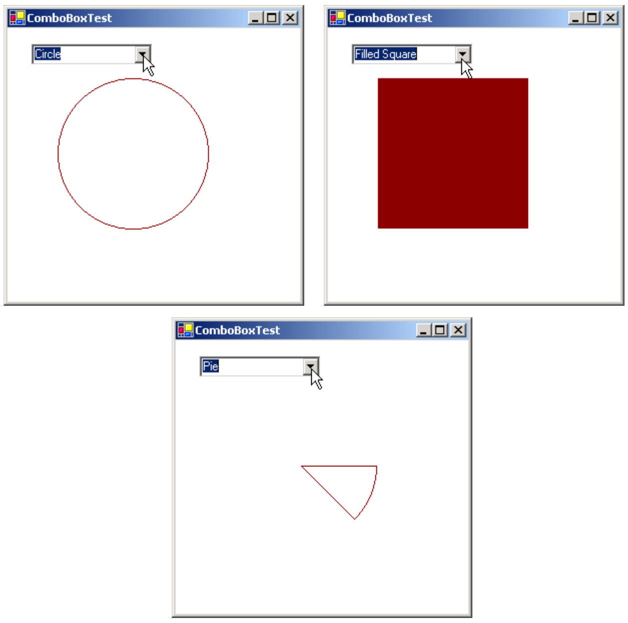
```
            case 7: // case filled pie is selected
               myGraphics.FillPie(
                  mySolidBrush, 50, 50, 150, 150, 0, 45 );
               break;

         } // end switch

      } // end method imageComboBox_SelectedIndexChanged

   } // end class ComboBoxTest
```

Program Output



Outline

ComboBoxTest.cs
Program Output



13.6 TreeViews

- Displays nodes hierarchically
- Parent nodes have children
- The first parent node is called the root
- Use Add method to add nodes

13.6 TreeView

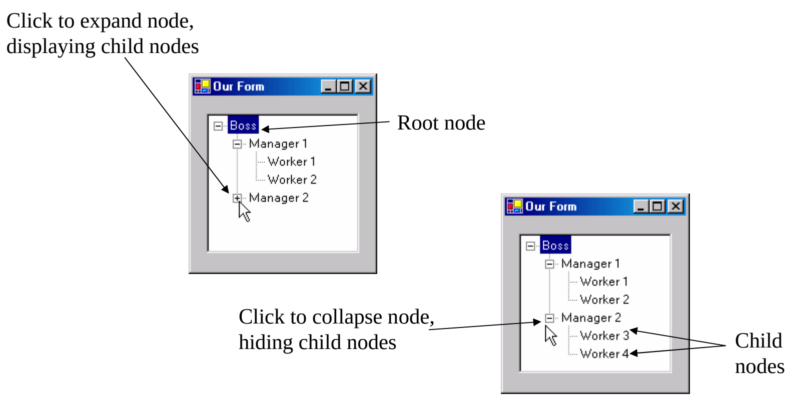


Fig. 13.17 Displaying a sample tree in a `TreeView`.

13.6 TreeView

TreeView properties and events	Description / Delegate and Event Arguments
Common Properties	
CheckBoxes	Indicates whether checkboxes appear next to nodes. **True** displays checkboxes. Default is **False**.
ImageList	Indicates the **ImageList** used to display icons by the nodes. An ***ImageList*** is a collection that contains a number of **Image** objects.
Nodes	Lists the collection of **TreeNode**s in the control. Contains methods **Add** (adds a **TreeNode** object), **Clear** (deletes the entire collection) and **Remove** (deletes a specific node). Removing a parent node deletes all its children.
SelectedNode	Currently selected node.
Common Event	*(Delegate **TreeViewEventHandler**, event arguments **TreeViewEventArgs**)*
AfterSelect	Generated after selected node changes. Default when double-clicked in designer.

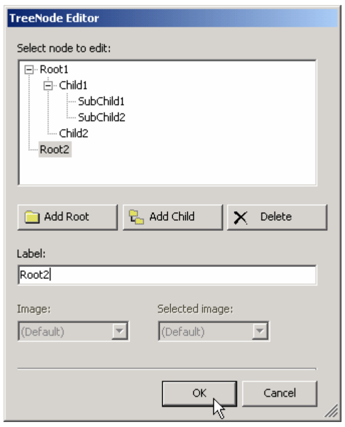
Fig. 13.18 **TreeView** properties and events.

13.6 TreeView

TreeNode properties and methods	Description / Delegate and Event Arguments
Common Properties	
Checked	Indicates whether the **TreeNode** is checked. (**CheckBoxes** property must be set to **True** in parent **TreeView**.)
FirstNode	Specifies the first node in the **Nodes** collection (i.e., first child in tree).
FullPath	Indicates the path of the node, starting at the root of the tree.
ImageIndex	Specifies the index of the image to be shown when the node is deselected.
LastNode	Specifies the last node in the **Nodes** collection (i.e., last child in tree).
NextNode	Next sibling node.
Nodes	The collection of **TreeNode**s contained in the current node (i.e., all the children of the current node). Contains methods **Add** (adds a **TreeNode** object), **Clear** (deletes the entire collection) and **Remove** (deletes a specific node). Removing a parent node deletes all its children.
PrevNode	Indicates the previous sibling node.
SelectedImageIndex	Specifies the index of the image to use when the node is selected.
Text	Specifies the text to display in the **TreeView**.
Common Methods	
Collapse	Collapses a node.
Expand	Expands a node.
ExpandAll	Expands all the children of a node.
GetNodeCount	Returns the number of child nodes.

Fig. 13.19 **TreeNode** properties and methods.

13.6 TreeView



```
Root1
  Child1
    SubChild1
    SubChild2
  Child2
Root2
```

Label: Root2

Fig. 13.20 **TreeNode Editor**.

Outline

TreeViewDirectoryStructureTest.cs

```
// Fig. 13.21: TreeViewDirectoryStructureTest.cs
// Using TreeView to display directory structure

using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.IO;

public class TreeViewDirectoryStructureTest
   : System.Windows.Forms.Form
{
   // contains view of c: drive directory structure
   private System.Windows.Forms.TreeView directoryTreeView;

   [STAThread]
   static void Main()
   {
      Application.Run(
         new TreeViewDirectoryStructureTest() );
   }

   public void PopulateTreeView(
      string directoryValue, TreeNode parentNode )
   {
      // populate current node with subdirectories
      string[] directoryArray =
         Directory.GetDirectories( directoryValue );

```

Class that creates children of root

Get subdirectories of root

Outline

TreeViewDirectoryStructureTest.cs

```
      // populate current node with subdirectories
      try
      {
         if ( directoryArray.Length != 0 )
         {
            // for every subdirectory, create new TreeNode,
            // add as child of current node and recursively
            // populate child nodes with subdirectories
            foreach ( string directory in directoryArray )
            {
               // create TreeNode for current directory
               TreeNode myNode = new TreeNode( directory );

               // add current directory node to parent node
               parentNode.Nodes.Add( myNode );

               // recursively populate every subdirectory
               PopulateTreeView( directory, myNode );
            }

         } // end if
      }

      // catch exception
      catch ( UnauthorizedAccessException )
      {
         parentNode.Nodes.Add( "Access denied" );
      }

   } // end PopulateTreeView

```

Create new node

Recursive call to finish tree

Catches security exception

Outline

TreeViewDirectoryStructureTest.cs

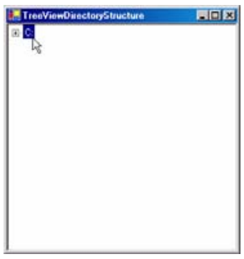
```
   // called by system when form loads
   private void TreeViewDirectoryStructureTest_Load(
      object sender, System.EventArgs e)
   {
      // add c:\ drive to directoryTreeView and
      // insert its subfolders
      directoryTreeView.Nodes.Add( "C:\\" );
      PopulateTreeView(
         "C:\\", directoryTreeView.Nodes[ 0 ] );
   }

} // end class TreeViewDirectoryStructure
```

Create root

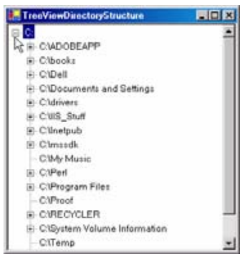
Outline

`TreeViewDirectoryStructureTest.cs`

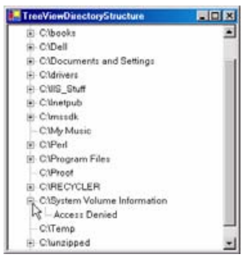
Program Output



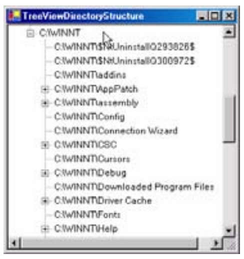
```
C:
```



```
C:
  C:\ADOBEAPP
  C:\books
  C:\Dell
  C:\Documents and Settings
  C:\drivers
  C:\IIS_Stuff
  C:\Inetpub
  C:\mssdk
  C:\My Music
  C:\Perl
  C:\Program Files
  C:\Proof
  C:\RECYCLER
  C:\System Volume Information
  C:\Temp
```



```
  C:\books
  C:\Dell
  C:\Documents and Settings
  C:\drivers
  C:\IIS_Stuff
  C:\Inetpub
  C:\mssdk
  C:\My Music
  C:\Perl
  C:\Program Files
  C:\Proof
  C:\RECYCLER
  C:\System Volume Information
    Access Denied
  C:\Temp
  C:\unzipped
```



```
  C:\WINNT
    C:\WINNT\$NtUninstallQ293826$
    C:\WINNT\$NtUninstallQ300972$
    C:\WINNT\addins
    C:\WINNT\AppPatch
    C:\WINNT\assembly
    C:\WINNT\Config
    C:\WINNT\Connection Wizard
    C:\WINNT\CSC
    C:\WINNT\Cursors
    C:\WINNT\Debug
    C:\WINNT\Downloaded Program Files
    C:\WINNT\Driver Cache
    C:\WINNT\Fonts
    C:\WINNT\Help
```

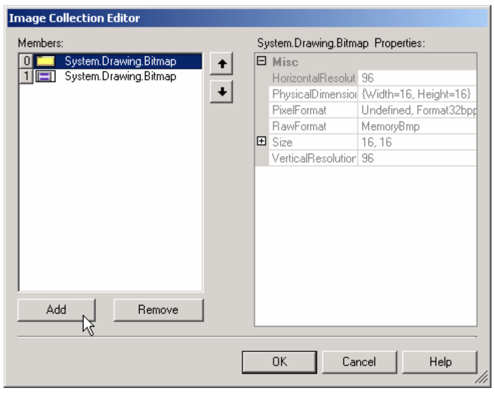
13.7 `ListViews`

- Displays list of items
  - Can select one or more items from list
  - Displays icons to go along with items

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.
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.

13.7 ListViews



System.Drawing.Bitmap Properties:

Misc	
HorizontalResolut	96
PhysicalDimensio	{Width=16, Height=16}
PixelFormat	Undefined, Format32bpp
RawFormat	MemoryBmp
Size	16, 16
VerticalResolution	96

Fig. 13.23 **Image Collection Editor** window for an **ImageList** component.

Outline

`ListViewTest.cs`

```
// Fig. 13.24: ListViewTest.cs
// Displaying directories and their contents in ListView.

using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.IO;

public class ListViewTest : System.Windows.Forms.Form
{
   // display labels for current location
   // in directory tree
   private System.Windows.Forms.Label currentLabel;
   private System.Windows.Forms.Label displayLabel;

   // display contents of current directory
   private System.Windows.Forms.ListView browserListView;

   // specifies images for file icons and folder icons
   private System.Windows.Forms.ImageList fileFolder;

   // get current directory
   string currentDirectory =
      Directory.GetCurrentDirectory();

   [STAThread]
   static void Main()
   {
      Application.Run( new ListViewTest() );
   }

```

Create Image List

Load the current directory

Outline

`ListViewTest.cs`

```
   // browse directory user clicked or go up one level
   private void browserListView_Click(
      object sender, System.EventArgs e )
   {
      // ensure item selected
      if ( browserListView.SelectedItems.Count != 0 )
      {
         // if first item selected, go up one level
         if ( browserListView.Items[ 0 ].Selected )
         {
            // create DirectoryInfo object for directory
            DirectoryInfo directoryObject =
               new DirectoryInfo( currentDirectory );

            // if directory has parent, load it
            if ( directoryObject.Parent != null )
               LoadFilesInDirectory(
                  directoryObject.Parent.FullName );
         }

         // selected directory or file
         else
         {
            // directory or file chosen
            string chosen =
               browserListView.SelectedItems[ 0 ].Text;

            // if item selected is directory
            if ( Directory.Exists( currentDirectory +
               "\\" + chosen ) )
            {
```

Test if item is selected

If first item selected go up one level

Make directory information

Test to see if at root

Return parent of current directory

Check if selected item is directory

Outline

ListViewTest.cs

```
               // load subdirectory
               // if in c:\, do not need '\',
               // otherwise we do
               if ( currentDirectory == "C:\\" )
                  LoadFilesInDirectory(
                     currentDirectory + chosen );
               else
                  LoadFilesInDirectory(
                     currentDirectory + "\\" + chosen );
            } //end if

         } // end else

         // update displayLabel
         displayLabel.Text = currentDirectory;

      } // end if

   } // end method browserListView_Click

   // display files/subdirectories of current directory
   public void LoadFilesInDirectory(
      string currentDirectoryValue )
   {
      // load directory information and display
      try
      {
         // clear ListView and set first item
         browserListView.Items.Clear();
         browserListView.Items.Add( "Go Up One Level" );

```

Load subdirectory

Update to display current directory

Class to load files in current directory

Outline

ListViewTest.cs

```
         // update current directory
         currentDirectory = currentDirectoryValue;
         DirectoryInfo newCurrentDirectory =
            new DirectoryInfo( currentDirectory );

         // put files and directories into arrays
         DirectoryInfo[] directoryArray =
            newCurrentDirectory.GetDirectories();

         FileInfo[] fileArray =
            newCurrentDirectory.GetFiles();

         // add directory names to ListView
         foreach ( DirectoryInfo dir in directoryArray )
         {
            // add directory to ListView
            ListViewItem newDirectoryItem =
               browserListView.Items.Add( dir.Name );

            // set directory image
            newDirectoryItem.ImageIndex = 0;
         }

         // add file names to ListView
         foreach ( FileInfo file in fileArray )
         {
            // add file to ListView
            ListViewItem newFileItem =
               browserListView.Items.Add( file.Name );

            newFileItem.ImageIndex = 1;  // set file image
         }
      } // end try

```

Get subdirectories of current directory

Get files of current directory

Add directory to list

Add file to list

Outline

ewTest.cs

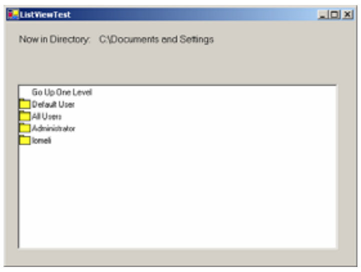
```
      // access denied
      catch ( UnauthorizedAccessException exception )
      {
         MessageBox.Show(
            "Warning: Some fields may not be " +
            "visible due to permission settings",
            "Attention", 0, MessageBoxIcon.Warning );
      }

   } // end method LoadFilesInDirectory

   // handle load event when Form displayed for first time
   private void ListViewTest_Load(
      object sender, System.EventArgs e )
   {
      // set image list
      Image folderImage = Image.FromFile(
         currentDirectory + "\\images\\folder.bmp" );

      Image fileImage = Image.FromFile( currentDirectory +
         "\\images\\file.bmp" );

      fileFolder.Images.Add( folderImage );
      fileFolder.Images.Add( fileImage );

      // load current directory into browserListView
      LoadFilesInDirectory( currentDirectory );
      displayLabel.Text = currentDirectory;

   }  // end method ListViewTest_Load

} // end class ListViewTest
```

Security exception handler

Load Images

Outline

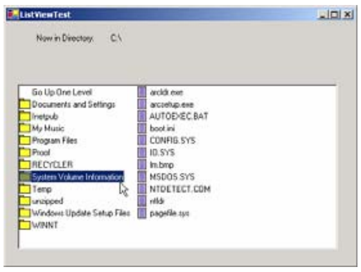
ListViewTest.cs
Program Output



ListViewTest

Now in Directory: C:\Documents and Settings

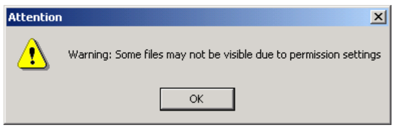
```
Go Up One Level
Default User
All Users
Administrator
lomeli
```



ListViewTest

Now in Directory: C:\

```
Go Up One Level                 arcldr.exe
Documents and Settings          arcsetup.exe
Inetpub                         AUTOEXEC.BAT
My Music                        boot.ini
Program Files                   CONFIG.SYS
Proof                           IO.SYS
RECYCLER                        lm.bmp
System Volume Information       MSDOS.SYS
Temp                            NTDETECT.COM
unzipped                        ntldr
Windows Update Setup Files      pagefile.sys
WINNT
```



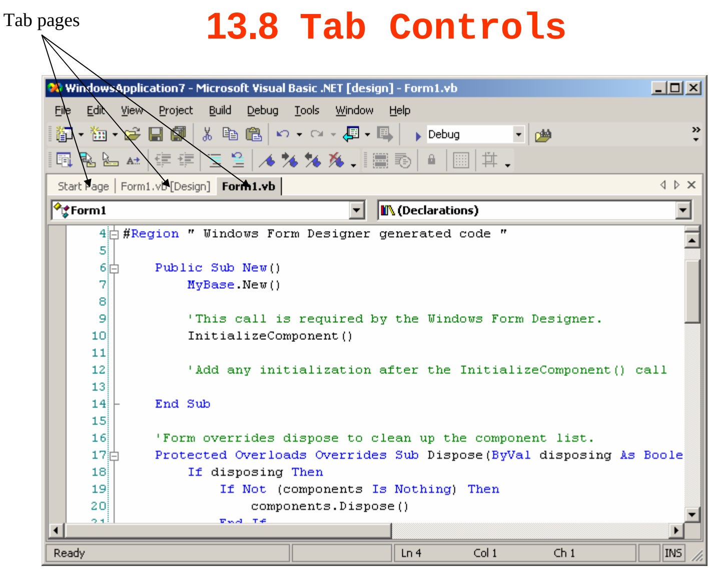
Attention

Warning: Some files may not be visible due to permission settings

13.8 `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



```
#Region " Windows Form Designer generated code "

    Public Sub New()
        MyBase.New()

        'This call is required by the Windows Form Designer.
        InitializeComponent()

        'Add any initialization after the InitializeComponent() call

    End Sub

    'Form overrides dispose to clean up the component list.
    Protected Overloads Overrides Sub Dispose(ByVal disposing As Boole
        If disposing Then
            If Not (components Is Nothing) Then
                components.Dispose()
```

Fig. 13.25 Tabbed pages in Visual Studio .NET.

13.8 Tab Controls

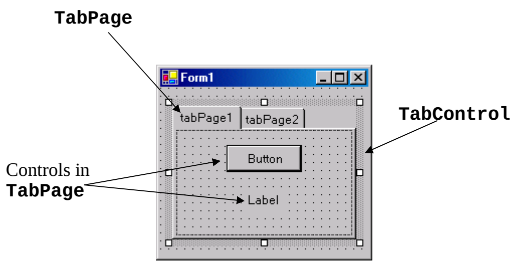


Fig. 13.26 Example **`TabControl`** with **`TabPage`**s.

13.8 Tab Controls

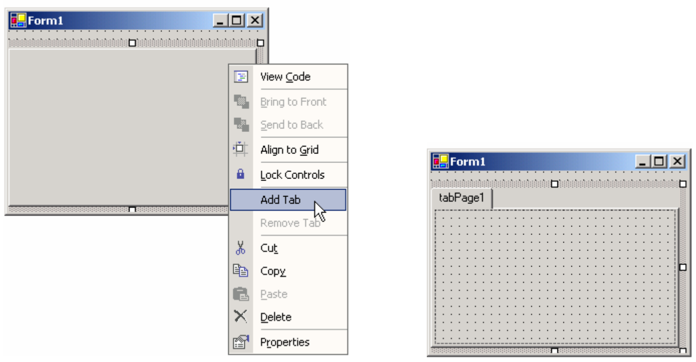


Fig. 13.27 Adding **`TabPage`**s to the **`TabControl`**.

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).

Fig. 13.28 **TabControl** properties and events.

Outline

UsingTabs.cs

```
// Fig. 13.29: UsingTabs.cs
// Using TabControl to display various font settings.

using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;

public class UsingTabs : System.Windows.Forms.Form
{
   // output label reflects text changes
   private System.Windows.Forms.Label displayLabel;

   // table control containing table pages colorTabPage,
   // sizeTabPage, messageTabPage and aboutTabPage
   private System.Windows.Forms.TabControl
      optionsTabControl;

   // table page containing color options
   private System.Windows.Forms.TabPage colorTabPage;
   private System.Windows.Forms.RadioButton
      greenRadioButton;
   private System.Windows.Forms.RadioButton redRadioButton;
   private System.Windows.Forms.RadioButton
      blackRadioButton;

```

Color tab

Color buttons for color tab

Outline

UsingTabs.cs

```
      // table page containing font size options
      private System.Windows.Forms.TabPage sizeTabPage;
      private System.Windows.Forms.RadioButton
         size20RadioButton;
      private System.Windows.Forms.RadioButton
         size16RadioButton;
      private System.Windows.Forms.RadioButton
         size12RadioButton;

      // table page containing text display options
      private System.Windows.Forms.TabPage messageTabPage;
      private System.Windows.Forms.RadioButton
         goodByeRadioButton;
      private System.Windows.Forms.RadioButton
         helloRadioButton;

      // table page containing about message
      private System.Windows.Forms.TabPage aboutTabPage;
      private System.Windows.Forms.Label messageLabel;

      [STAThread]
      static void Main()
      {
         Application.Run( new UsingTabs() );
      }

      // event handler for black color radio button
      private void blackRadioButton_CheckedChanged(
         object sender, System.EventArgs e )
      {
         displayLabel.ForeColor = Color.Black;
      }

```

Size tab

Size buttons

Message tab

About tab

Event handler

Outline

UsingTabs.cs

```
      // event handler for red color radio button
      private void redRadioButton_CheckedChanged(
         object sender, System.EventArgs e )
      {
         displayLabel.ForeColor = Color.Red;
      }

      // event handler for green color radio button
      private void greenRadioButton_CheckedChanged(
         object sender, System.EventArgs e )
      {
         displayLabel.ForeColor = Color.Green;
      }

      // event handler for size 12 radio button
      private void size12RadioButton_CheckedChanged(
         object sender, System.EventArgs e )
      {
         displayLabel.Font =
            new Font( displayLabel.Font.Name, 12 );
      }

      // event handler for size 16 radio button
      private void size16RadioButton_CheckedChanged(
         object sender, System.EventArgs e )
      {
         displayLabel.Font =
            new Font( displayLabel.Font.Name, 16 );
      }

```

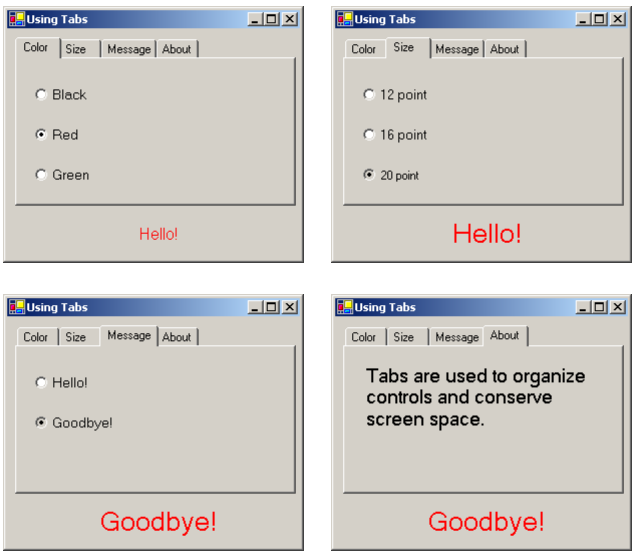
Event handlers

UsingTabs.cs

```
   // event handler for size 20 radio button
   private void size20RadioButton_CheckedChanged(
      object sender, System.EventArgs e )
   {
      displayLabel.Font =
         new Font( displayLabel.Font.Name, 20 );
   }

   // event handler for message "Hello!" radio button
   private void helloRadioButton_CheckedChanged(
      object sender, System.EventArgs e )
   {
      displayLabel.Text = "Hello!";
   }

   // event handler for message "Goodbye!" radio button
   private void goodByeRadioButton_CheckedChanged(
      object sender, System.EventArgs e )
   {
      displayLabel.Text = "Goodbye!";
   }

} // end class UsingTabs
```

Event handlers

UsingTabs.cs
Program Output



```
Color tab:    ○ Black   ● Red   ○ Green                Hello!
Size tab:     ○ 12 point   ○ 16 point   ● 20 point      Hello!
Message tab:  ○ Hello!   ● Goodbye!                     Goodbye!
About tab:    Tabs are used to organize controls and conserve screen space.   Goodbye!
```

13.9 Multiple-Document Interface Windows

- Users can edit multiple documents at once
- Usually more complex then single-document-interface applications
- Application window called parent, others child
- Parent and child menus can be merged
  - Based on **`MergeOrder`** property
- Child windows can be arranged in parent window:
  - Tiled windows: completely fill parent, no overlap
    - Either horizontal or vertical
  - Cascaded windows: overlap, same size, display title bar
  - ArrangeIcons: arranges icons for minimized windows

13.9 Multiple Document Interface (MDI) Windows

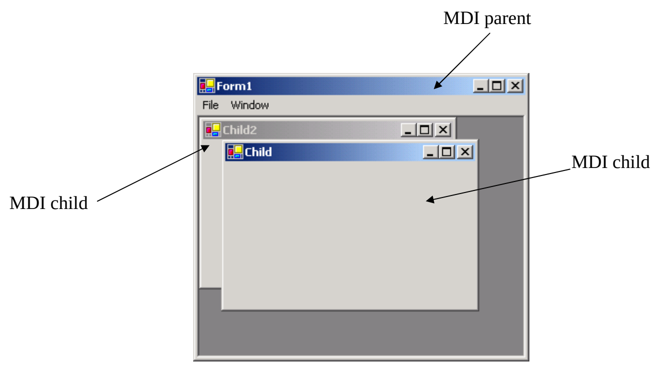
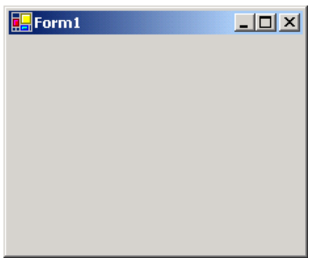
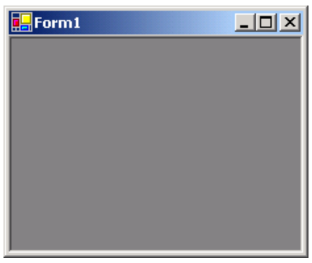


Fig. 13.30 MDI parent and MDI child.

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

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.

Fig. 13.32 MDI parent and MDI child events and properties.

13.9 Multiple Document Interface (MDI) Windows

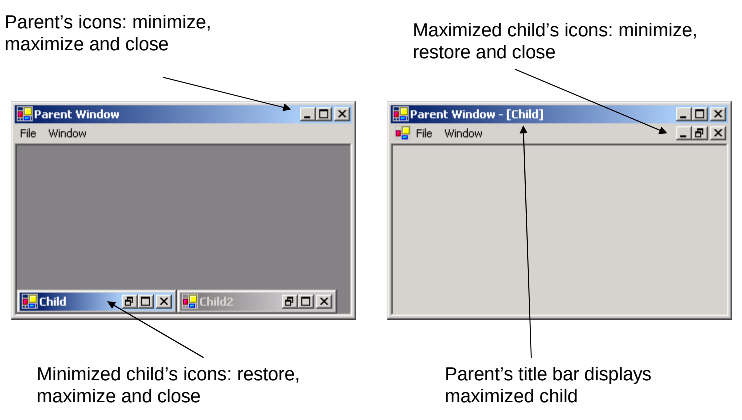

Parent's icons: minimize, maximize and close
Maximized child's icons: minimize, restore and close
Parent Window
File Window
Child
Child2
Parent Window - [Child]
Minimized child's icons: restore, maximize and close
Parent's title bar displays maximized child

Fig. 13.33 Minimized and maximized child windows.

13.9 Multiple Document Interface (MDI) Windows

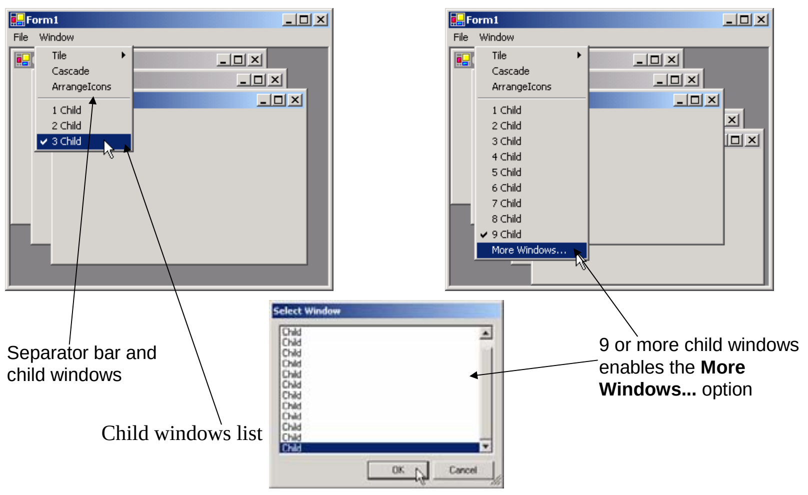
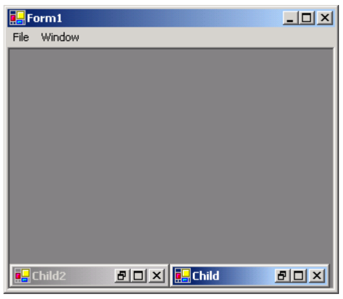
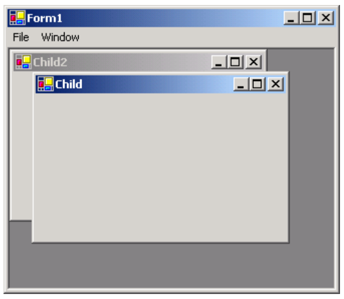


Fig. 13.34 Using `MenuItem` property `MdiList`.

13.9 Multiple Document Interface (MDI) Windows



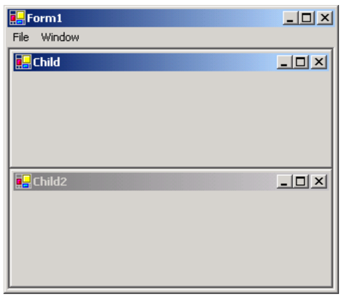
ArrangeIcons



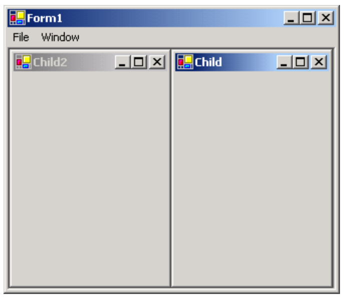
Cascade

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).

79

Outline

UsingMDI.cs

```
// Fig. 13.36: UsingMDI.cs
// Demonstrating use of MDI parent and child windows.
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;

public class UsingMDI : System.Windows.Forms.Form
{
   private System.Windows.Forms.MainMenu mainMenu1;
   private System.Windows.Forms.MenuItem fileMenuItem;
   private System.Windows.Forms.MenuItem newMenuItem;
   private System.Windows.Forms.MenuItem child1MenuItem;
   private System.Windows.Forms.MenuItem child2MenuItem;
   private System.Windows.Forms.MenuItem child3MenuItem;
   private System.Windows.Forms.MenuItem exitMenuItem;
   private System.Windows.Forms.MenuItem formatMenuItem;
   private System.Windows.Forms.MenuItem cascadeMenuItem;
   private System.Windows.Forms.MenuItem
      tileHorizontalMenuItem;
   private System.Windows.Forms.MenuItem
      tileVerticalMenuItem;

   [STAThread]
   static void Main()
   {
      Application.Run( new UsingMDI() );
   }

```

File menu

New submenu

Exit submenu

Formant menu

Cascade option

Tiling options

80

Outline

UsingMDI.cs

```
   // create Child 1 when menu clicked
   private void child1MenuItem_Click(
      object sender, System.EventArgs e )
   {
      // create new child
      Child formChild = new Child( "Child 1",
         "\\images\\csharphtp1.jpg" );
      formChild.MdiParent = this;    // set parent
      formChild.Show();              // display child
   }

   // create Child 2 when menu clicked
   private void child2MenuItem_Click(
      object sender, System.EventArgs e )
   {
      // create new child
      Child formChild = new Child( "Child 2",
         "\\images\\vbnethtp2.jpg" );
      formChild.MdiParent = this;    // set parent
      formChild.Show();              // display child
   }

   // create Child 3 when menu clicked
   private void child3MenuItem_Click(
      object sender, System.EventArgs e )
   {
      // create new child
      Child formChild = new Child( "Child 3",
         "\\images\\pythonhtp1.jpg" );
      formChild.MdiParent = this;    // set parent
      formChild.Show();              // display child
   }

```

Create child windows

Outline

UsingMDI.cs

```
      // exit application
      private void exitMenuItem_Click(
         object sender, System.EventArgs e )
      {
         Application.Exit();
      }

      // set cascade layout
      private void cascadeMenuItem_Click(
         object sender, System.EventArgs e )
      {
         this.LayoutMdi( MdiLayout.Cascade );
      }

      // set TileHorizontal layout
      private void tileHorizontalMenuItem_Click(
         object sender, System.EventArgs e )
      {
         this.LayoutMdi( MdiLayout.TileHorizontal );
      }

      // set TileVertical layout
      private void tileVerticalMenuItem_Click(
         object sender, System.EventArgs e )
      {
         this.LayoutMdi( MdiLayout.TileVertical );
      }

   } // end class UsingMDI
```

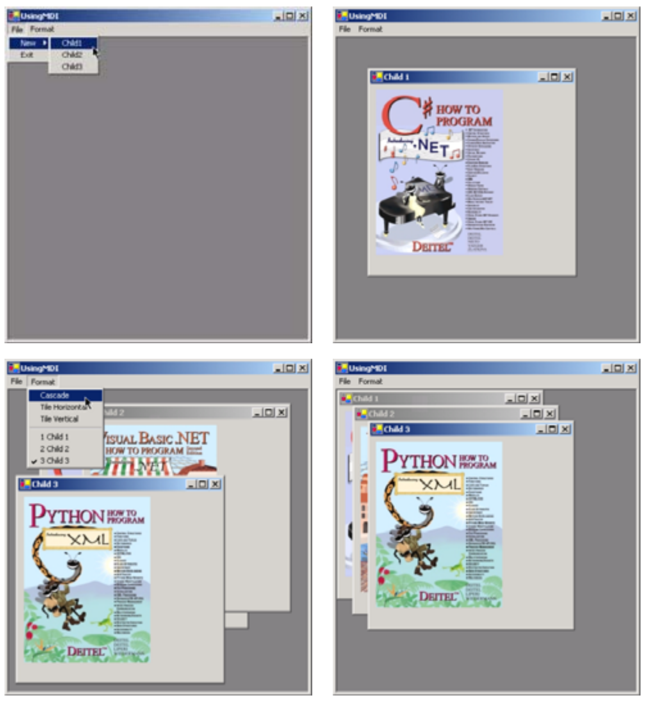
Cascade

Tile horizontally

Tile vertically

Outline

UsingMDI.cs
Program Output



Outline

Child.cs

```
// Fig. 13.37: Child.cs
// Child window of MDI parent.
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.IO;

public class Child : System.Windows.Forms.Form
{
   private System.Windows.Forms.PictureBox pictureBox;

   public Child( string title, string fileName )
   {
      // Required for Windows Form Designer support
      InitializeComponent();

      Text = title; // set title text

      // set image to display in pictureBox
      pictureBox.Image = Image.FromFile(
         Directory.GetCurrentDirectory() + fileName );
   }
}
```

Child class

Create picture box

Display title

Display picture

13.10 Visual Inheritance

- Create **`Form`** by inheriting from another **Form**
  - Derived **`Form`** inherits functionality of base **`Form`**
  - Derived **`Form`** inherits visual aspects of base **`Form`**

Outline

VisualInheritance.cs

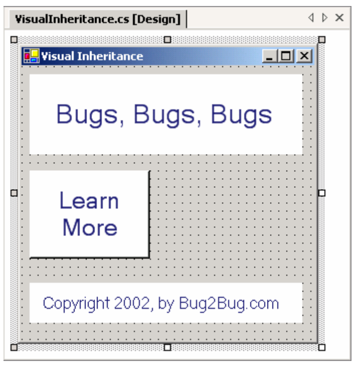
```
// Fig. 13.38: VisualInheritance.cs
// Base Form for use with visual inheritance
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;

public class VisualInheritance : System.Windows.Forms.Form
{
   private System.Windows.Forms.Label bugsLabel;
   private System.Windows.Forms.Button learnMoreButton;
   private System.Windows.Forms.Label label1;

   [STAThread]
   static void Main()
   {
      Application.Run( new VisualInheritance() );
   }

   private void learnMoreButton_Click( object sender,
      System.EventArgs e )
   {
      MessageBox.Show(
         "Bugs, Bugs, Bugs is a product of Bug2Bug.com",
         "Learn More", MessageBoxButtons.OK,
         MessageBoxIcon.Information );
   }
}
```

Learn More display method

Outline

VisualInheritance.cs

Program Output



13.11 User-Defined Controls

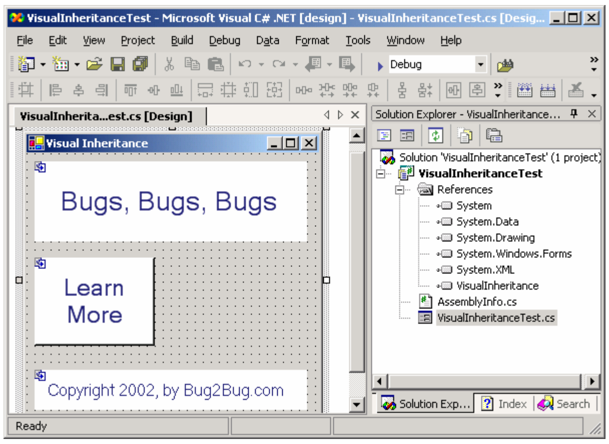

VisualInheritanceTest - Microsoft Visual C# .NET [design] - VisualInheritanceTest.cs [Design]
File Edit View Project Build Debug Data Format Tools Window Help
Debug
VisualInherita...est.cs [Design]
Visual Inheritance
Bugs, Bugs, Bugs
Learn More
Copyright 2002, by Bug2Bug.com
Solution Explorer - VisualInheritance...
Solution 'VisualInheritanceTest' (1 project)
VisualInheritanceTest
References
System
System.Data
System.Drawing
System.Windows.Forms
System.XML
VisualInheritance
AssemblyInfo.cs
VisualInheritanceTest.cs
Solution Exp...
Index
Search
Ready

Fig. 13.39 Visual Inheritance through the Form Designer.

Outline

VisualInheritanceTest.cs

```
// Fig. 13.40: VisualInheritanceTest.cs
// Derived Form using visual inheritance.
using System;
using System.Collections;
using System.ComponentModel;
using System.Drawing;
using System.Windows.Forms;

public class VisualInheritanceTest :
   VisualInheritance.VisualInheritance
{
   private System.Windows.Forms.Button learnProgramButton;

   // invoke when user clicks Learn the Program Button
   private void learnProgramButton_Click( object sender,
      System.EventArgs e )
   {
      MessageBox.Show(
         "This program was created by Deitel & Associates",
         "Learn the Program", MessageBoxButtons.OK,
         MessageBoxIcon.Information );
   }

   public static void Main( string[] args )
   {
      Application.Run( new VisualInheritanceTest() );
   }
}
```

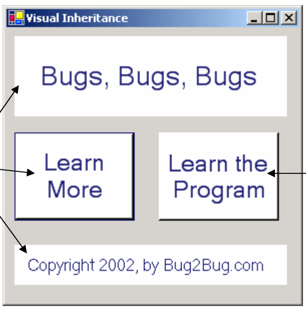
VisualInheritanceTest class is derived from VisualInheritance class

Display message box

Outline

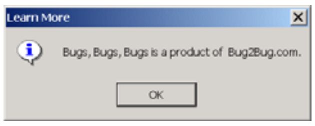
VisualInheritanceTest.cs

Program Output



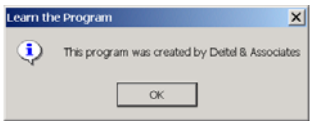
Derived class cannot modify these controls

Derived class can modify this control


Learn More

Bugs, Bugs, Bugs is a product of Bug2Bug.com.

OK


Learn the Program

This program was created by Deitel & Associates

OK

13.11 User-Defined Controls

- Custom controls that inherit from other classes
  - Ex: can change appearance of a label

13.11 User-Defined Controls

Custom Control Techniques and **`PaintEventArgs`** Properties	Description
Inherit from Windows Forms control	Add functionality to a preexisting control. If overriding method **`OnPaint`**, call base class **`OnPaint`**. Can only add to the original control appearance, not redesign it.
Create a ***`UserControl`***	Create a **`UserControl`** composed of multiple preexisting controls (and combine their functionality). Cannot override **`OnPaint`** methods of custom controls. Instead, add drawing code to a **`Paint`** event handler. Can only add to the original control appearance, not redesign it.
Inherit from class ***`Control`***	Define a brand-new control. Override **`OnPaint`** method, call base class method **`OnPaint`** and include methods to draw the control. Can customize control appearance and functionality.
`PaintEventArgs` *Properties*	*Use this object inside method* ***`OnPaint`*** ***or*** ***`Paint`*** ***to draw on the control.***
`Graphics`	Indicates the graphics object of control. Used to draw on control.
`ClipRectangle`	Specifies the rectangle indicating boundary of control.

Fig. 13.41 Custom control creation.

Outline

`ClockUserControl.cs`

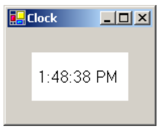
```
// Fig. 13.42: ClockUserControl.cs
// User-defined control with a timer and a label.

using System;
using System.Collections;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Windows.Forms;

public class ClockUserControl
   : System.Windows.Forms.UserControl
{
   private System.Windows.Forms.Timer clockTimer;
   private System.Windows.Forms.Label displayLabel;

   // update label at every tick
   private void clockTimer_Tick(
      object sender, System.EventArgs e )
   {
      // get current time (Now), convert to string
      displayLabel.Text = DateTime.Now.ToLongTimeString();

   } // end method clockTimer_Tick

} // end class ClockUserControl
```

Timer

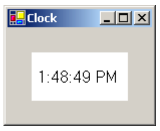
Label

Update label method

Display current time



1:48:38 PM



1:48:49 PM

13.11 User-Defined Controls

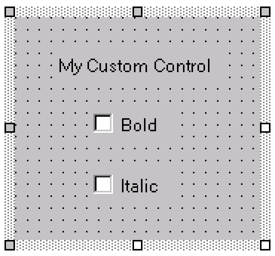
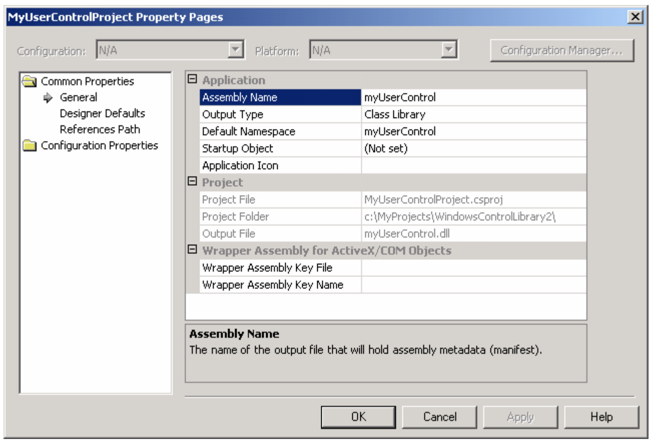


Fig. 13.43 Custom-control creation.

13.11 User-Defined Controls


MyUserControlProject Property Pages

Configuration: N/A Platform: N/A Configuration Manager...

Common Properties
- General
- Designer Defaults
- References Path

Configuration Properties

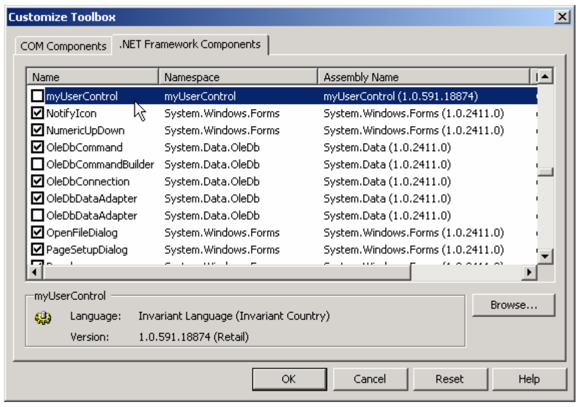
Application	
Assembly Name	myUserControl
Output Type	Class Library
Default Namespace	myUserControl
Startup Object	(Not set)
Application Icon	
Project	
Project File	MyUserControlProject.csproj
Project Folder	c:\MyProjects\WindowsControlLibrary2\
Output File	myUserControl.dll
Wrapper Assembly for ActiveX/COM Objects	
Wrapper Assembly Key File	
Wrapper Assembly Key Name	

Assembly Name
The name of the output file that will hold assembly metadata (manifest).

OK Cancel Apply Help

Fig. 13.44 Project properties dialog.

13.11 User-Defined Controls



Name	Namespace	Assembly Name
☐ myUserControl	myUserControl	myUserControl (1.0.591.18874)
☑ NotifyIcon	System.Windows.Forms	System.Windows.Forms (1.0.2411.0)
☑ NumericUpDown	System.Windows.Forms	System.Windows.Forms (1.0.2411.0)
☑ OleDbCommand	System.Data.OleDb	System.Data (1.0.2411.0)
☐ OleDbCommandBuilder	System.Data.OleDb	System.Data (1.0.2411.0)
☑ OleDbConnection	System.Data.OleDb	System.Data (1.0.2411.0)
☑ OleDbDataAdapter	System.Data.OleDb	System.Data (1.0.2411.0)
☐ OleDbDataAdapter	System.Data.OleDb	System.Data (1.0.2411.0)
☑ OpenFileDialog	System.Windows.Forms	System.Windows.Forms (1.0.2411.0)
☑ PageSetupDialog	System.Windows.Forms	System.Windows.Forms (1.0.2411.0)

myUserControl
Language: Invariant Language (Invariant Country)
Version: 1.0.591.18874 (Retail)

Fig. 13.45 Custom control added to the **ToolBox**.

13.11 User-Defined Controls

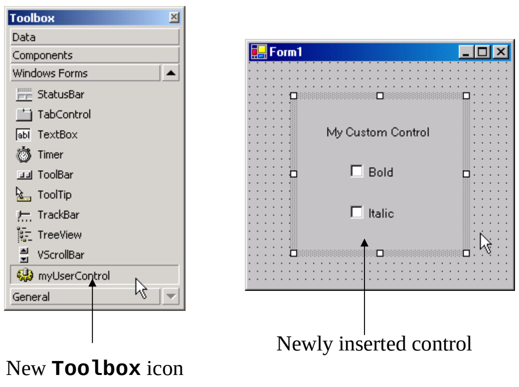


Fig. 13.46 Custom control added to a **Form**.