CMPE 1666
Intermediate Programming

Windows Controls I

Controls

- Windows controls are basically form elements that can perform specific functions.
- Examples include button, checkbox, textbox, label, listbox, etc…
- Controls are based on the class System.Windows.Forms.Control

Controls

- When you are editing your form in designer mode, you can drag and drop controls from the Toolbox to the form.
- The code required to create the control on the form is automatically added by Visual Studio.

Common Control Properties

- The Control class has over 75 properties available.
- We will only examine a subset of the more useful properties.
- Control properties can be set at design time or at runtime using the program.

Common Control Properties

Property	Description
Anchor	Selects which edges of the control are anchored to its container.
BackColor	Selects the background color for the control.
Dock	Defines which borders of the control are bound to the container.
Enabled	If true the control is enabled, if false the control is disabled.
Font	Font used for displayed text.
Location	Location of the control in the form's client area.
Name	Name of the control.
Size	Size of the control.
TabIndex	Tab order of the control in the container (form).
TabStop	If true, user can give focus to the control with the tab key.
Text	Text associated with the control.
Visible	If true control is visible, otherwise invisible.

Common Control Methods

- Controls contain many methods that are specific to that control.
- There are some common methods that perform the same operations on most controls.

Common Control Methods

Method	Action
Focus	Provides the control the keyboard focus.
Hide	Hides the control by setting its Visible property to false.
Show	Shows the control by setting its Visible property to true.

Label

- The Label control displays text that cannot be changed by the user.
- The text can be modified by the program using the Text property.
- It is often used to label other controls, or to display results as text.

Common Label Properties

Property	Description
Text	The text displayed by the label.
Name	The program name of the control.
Font	Font used for the text.
ForeColor	The text color.
BackColor	The background color
TextAlign	The alignment of the text in the control.

Label Events

- The Label control has many similar events to the form.
- The common label event is to handle a mouse click.
- This event handler can be added by double clicking on the label in design mode.

TextBox

- The TextBox control can be used to input text from the user using the keyboard, or display text.
- A TextBox can be single line or multiline.
- A password TextBox will not display the keys that the user enters.

Common TextBox Properties

Property	Description
AcceptsReturn	If true in a multiline TextBox, pressing Enter creates a new line. Otherwise, the form’s default AcceptButton is clicked.
Name	The program identifier for the TextBox.
Multiline	If true, the TextBox can span multiple lines.
PasswordChar	When this is set to a character, the TextBox becomes a password TextBox. The character set is displayed as the user presses keys. If not set, the TextBox displays all text as it is entered.
ReadOnly	TextBox has gray background and text cannot be entered by the user.
ScrollBars	For a multiline TextBox, this determines which scrollbars appear.
Text	The text contained in the TextBox.

Common TextBox Properties

Property	Description
AcceptsTab	If true in a multiline TextBox, pressing tab inputs a tab character. Otherwise, the next control in the form's tab order gains the focus.
Lines	The lines of text in a multiline TextBox as an array of strings. The lines are from the Text property and are the strings separated by newline characters. You may assign an array of strings to this property, but you should not assign individual lines to it.
CharacterCasing	Can convert the characters to upper case or lower case as they are entered.
MaxLength	Gets or sets the maximum number of characters that the user can enter.
TextAlign	The alignment of the text within the TextBox
WordWrap	For a multiline TextBox, this determines if words are automatically wrapped over to the next line.

Common TextBox Methods

Method	Description
Clear	Clears the contents of the TextBox.
Copy	Copies the selected text to the clipboard.
Cut	Cuts (moves) the selected text to the clipboard.
Paste	Replaces the selected text with the contents of the clipboard.
Undo	Undoes the last edit.
Focus	Sets the keyboard focus to the textbox.

TextBox Events

- The TextBox has many events that are the same as a form.
- The common event for the TextBox is TextChanged, which occurs when the contents of the Text property are modified.

Button

- The Button control is used to issue a command to the program.
- A Button is usually clicked with the mouse.
- A Button designated as the form’s AcceptButton will be clicked when the Enter key is pressed.

Button

- A Button designated as the form’s CancelButton will be clicked when the Escape key is pressed.
- The common event for a button is Click, which occurs when the mouse is clicked on the button.

Button Properties

Property	Description
DialogResult	The Button can return a DialogResult value (such as Ok) if it is used in a dialog box.
Enabled	If true the Button is enabled.
FlatStyle	Selects the style of the Button.
Text	The text contained in the Button.
TextAlign	Alignment of the text in the Button.

MessageBox

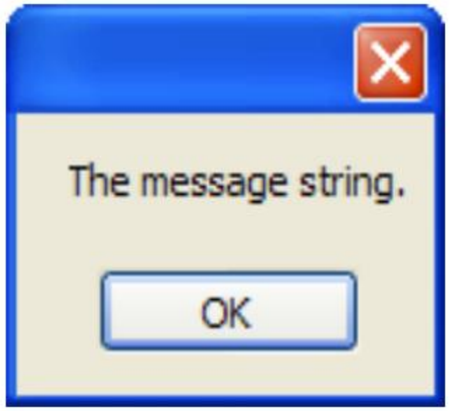
- The MessageBox is not a control, but a dialog box.
- We will examine it now since it is usually used to display error conditions.
- The class MessageBox is inherited from System.Windows.Forms

MessageBox.Show()

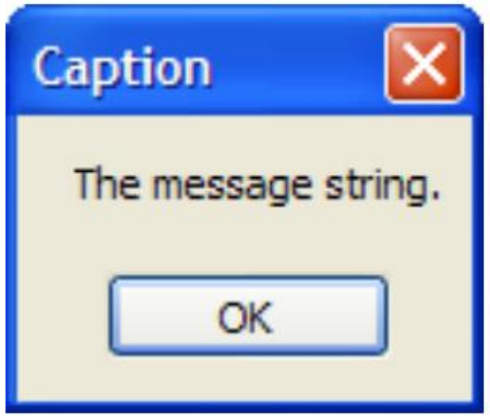
- A MessageBox can be displayed by calling the static method Show().
- There are many overloads of the Show() method, allowing you to specify how the MessageBox is displayed.

MessageBox.Show()

- MessageBox.Show("The message string.");


The message string.

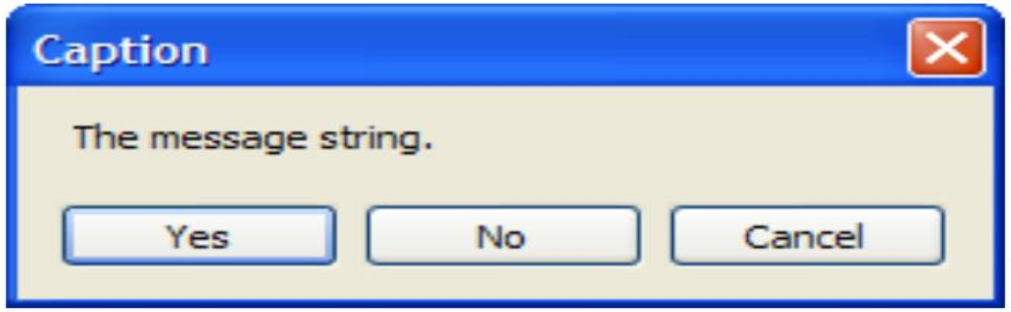
- MessageBox.Show("The message string.", "Caption");


Caption

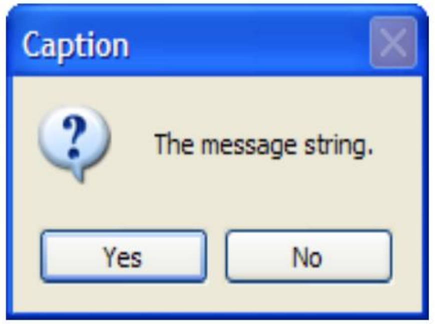
The message string.

MessageBox.Show()

- MessageBox.Show("The message string.", "Caption", MessageBoxButtons.YesNoCancel);



- MessageBox.Show("The message string.", "Caption", MessageBoxButtons.YesNo, MessageBoxIcon.Question);



MessageBox.Show()

- The Show method returns a value of type DialogResult to indicate which button was pressed.

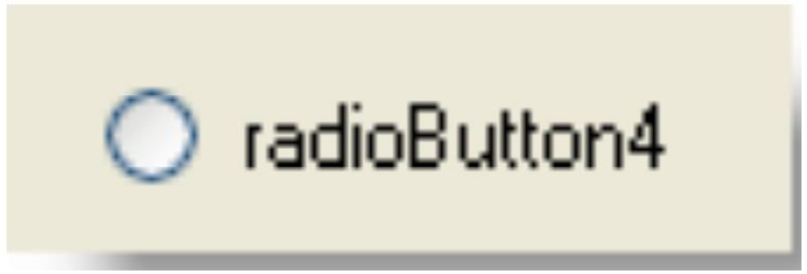
```
DialogResult drResult; drResult = MessageBox.Show("The message string.", "Caption", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (drResult == DialogResult.Yes) System.Diagnostics.Trace.WriteLine("Yes");
```

MessageBox Caption

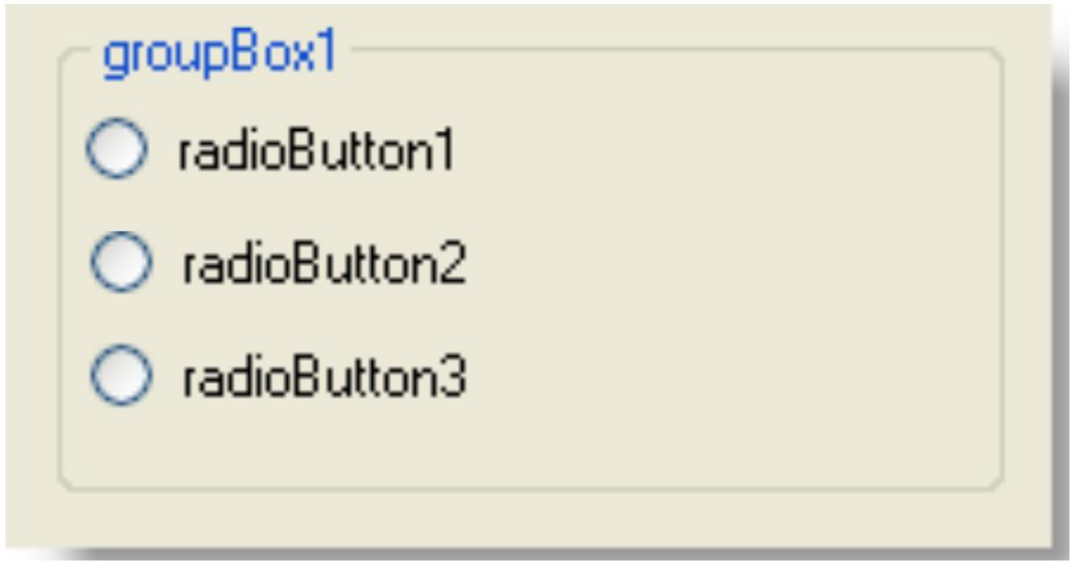
- The caption of the MessageBox usually contains the name of the application that caused it to appear.

RadioButton

- The RadioButton is a form of button that is usually used is to make a one of many selection.



RadioButton

- RadioButtons are usually placed in a GroupBox, which is labeled.
- 

RadioButton Properties

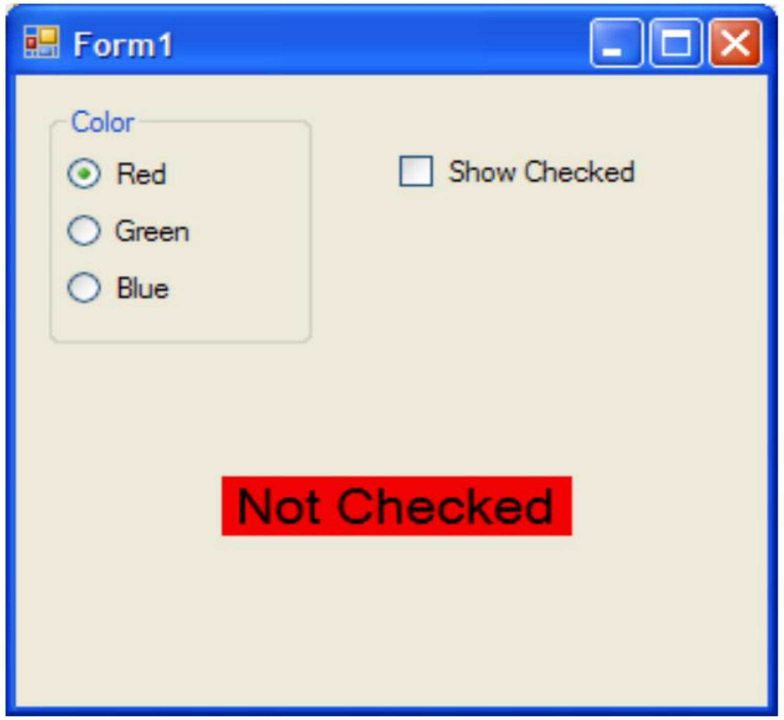
Property	Description
AutoCheck	Causes the radio button to automatically change state when clicked.
CheckAlign	Alignment of check and label, usually middle left.
Checked	Indicates if radio button is checked or not. (true or false)
TabIndex	Tab position for tab key selection.
TabStop	If true, the user can tab to the radio button.

RadioButton Events

- The common event for the radio button is the CheckChanged event.
- The CheckChanged event occurs when the radio button is checked or unchecked.
- The Checked property can be used to determine the current state.

RadioButton Events

- The radio buttons shown below select a color for the label.

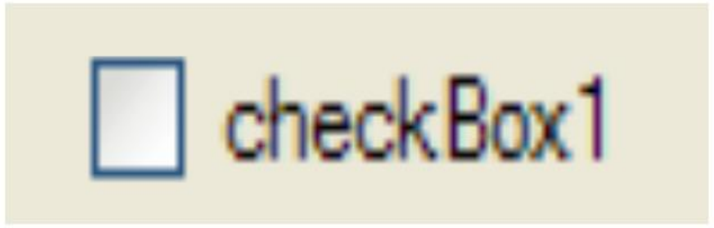


RadioButton Events

```
private void red_radioButton_CheckedChanged(object sender, EventArgs e) {
output_label.BackColor = Color.Red;
}
 private void green_radioButton_CheckedChanged(object sender, EventArgs e) {
                    output_label.BackColor = Color.Green;
}
 private void blue_radioButton_CheckedChanged(object sender, EventArgs e) {
          output_label.BackColor = Color.Blue;
}
```

CheckBox

- The CheckBox can be used to make true/false selections.
- A CheckBox can be two state, or three state.



CheckBox Properties

Property	Description
AutoCheck	Causes the checkbox to automatically change state when clicked.
CheckAlign	Alignment of check and label, usually middle left.
Checked	Indicates if checkbox is checked or not. (true or false)
CheckState	Indicates the check state for a three state checkbox. Can be Unchecked, Checked, or Indeterminate.
TabIndex	Tab position for tab key selection.
TabStop	If true, the user can tab to the radio button.
ThreeState	If true, the checkbox is three state.

CheckBox Events

- The common event for a checkbox is the CheckedChanged event, which occurs when the check state changes.
- A three state checkbox must use the CheckStateChanged event, and use the value of CheckState property to determine the current state.

CheckBox Events

- Example of handling CheckChanged event for a two state checkbox.

```
private void showCheck_checkBox_CheckedChanged(object sender,
EventArgs e) {
    if (showCheck_checkBox.Checked == true)
              output_label.Text = "Checked";
   else output_label.Text = "Not Checked";
}
```

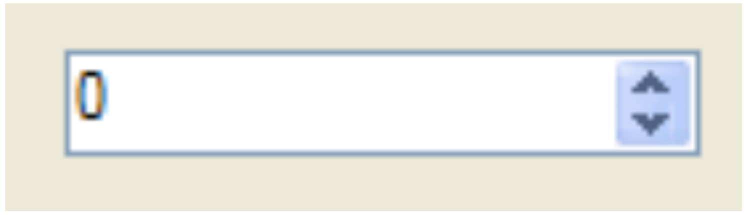
CheckBox Events

- Example of handling CheckStateChanged event for a three state checkbox.

```
private void gotobar_checkBox_CheckStateChanged(object sender, EventArgs e) {
     if (gotobar_checkBox.CheckState == CheckState.Unchecked)
               label1.Text = "No";
   if (gotobar_checkBox.CheckState == CheckState.Indeterminate)
                label1.Text = "Maybe";
if (gotobar_checkBox.CheckState == CheckState.Checked) label1.Text = "Yes!";
}
```

NumericUpDown

- The NumericUpDown control allows the user to set a value within a range using arrow buttons.



NumericUpDown

- You can set the minimum and maximum values, as well as the increment to be used.
- The Value property is the current number selected, and is of type Decimal.
- The user can also type in a value, unless the control is set to ReadOnly.

NumericUpDown Properties

Property	Description
DecimalPlaces	Number of decimal places to display.
Hexadecimal	If true, displays values in hexadecimal.
Increment	The amount to increment or decrement with each button click.
Maximum	The maximum value for the numeric up down control.
Minimum	The minimum value for the numeric up down control.
ReadOnly	If true, the user cannot enter a value.
TextAlign	Alignment of number in the edit portion.
ThousandsSeparater	Indicates if thousands separator will be used.

NumericUpDown Properties

Property	Description
UpDownAlign	Alignment of the arrow buttons to the edit portion.
Value	The current set value stored as a Decimal.

NumericUpDown Events

- The common event for the NumericUpDown control is ValueChanged.
- The ValueChanged event is fired if the arrow buttons are used, or the user enters a value and presses return.

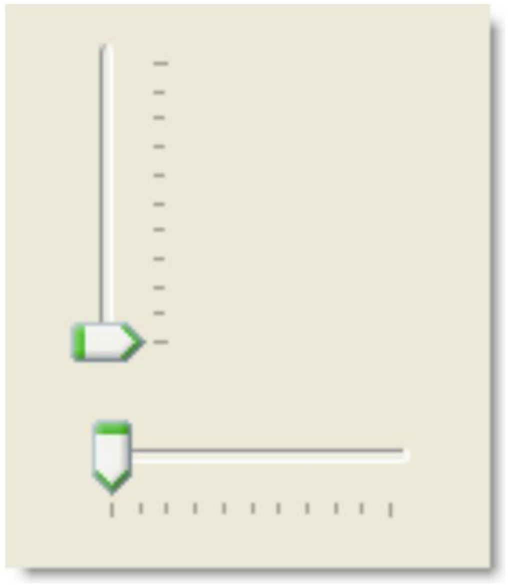
NumericUpDown Events

- Often, you wish to use the Value property as an integer.

```
private void numericUpDown1_ValueChanged(object sender, EventArgs e) {
        int iSeconds;
        iSeconds = (int)numericUpDown1.Value;
         or…
          iSeconds = Convert.ToInt32(numericUpDown1.Value);
 }
```

TrackBar

- Allows you to move a Slider left to right (or up and down) to adjust a value.



TrackBar Properties

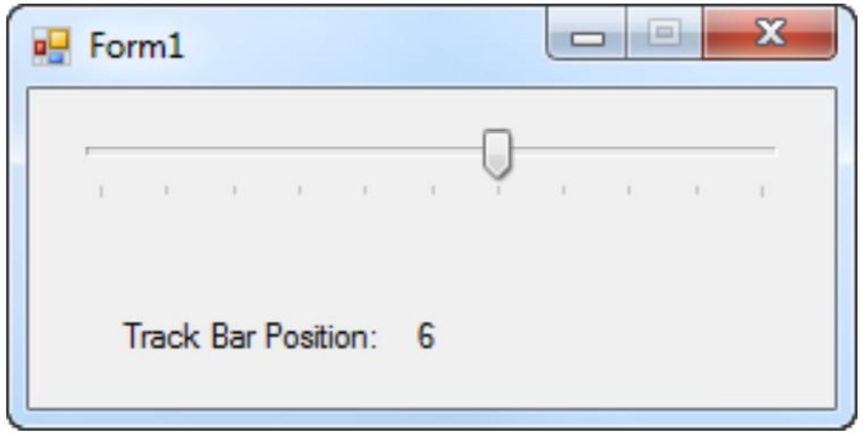
Property	Purpose
Name	The control name used in the program.
LargeChange	The number of positions the slider moves in response to mouse clicks or the Page Up or Page Down keys.
Maximum	The maximum value for the slider.
Minimum	The minimum value for the slider
Orientation	Horizontal or vertical.
RightToLeft	Slider direction from min to max value.
SmallChange	The number of positions the slider moves in response to the arrow keys.
TickFrequency	The number of positions between tick marks.
TickStyle	Position of the ticks relative to the control.
Value	The position of the slider as an integer

TrackBar Event

- The default event for the Trackbar is the scroll event.
- This is usually used to read the Value from the control.
- Setting the Value property will set the slider postion.

TrackBar Event

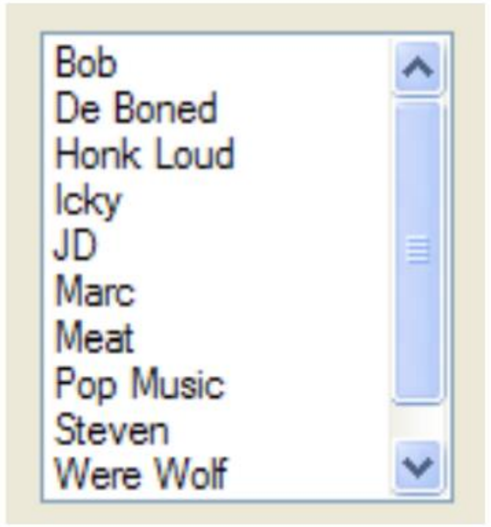
```
private void trackBar_Scroll(object sender, EventArgs e) {
	lblTrackPosition.Text = trackBar.Value.ToString();
}
```



Track Bar Position: 6

ListBox

- The ListBox will display a list of items from which the user can select one, or multiple items.



ListBox Design Properties

Property	Description
ColumnWidth	Width of each column in a multicolumn listbox.
HorizontalScrollBar	If true, listbox displays a horizontal scrollbar.
Items	The items stored in the listbox as a collection.
MultiColumn	If true, the listbox values are displayed in columns.
ScrollAlwaysVisible	If true, always display the vertical scroll bar.
ReadOnly	If true, the user cannot enter a value.
SelectionMode	Determines if more than one item can be selected.
Sorted	If true, items are sorted as they are added.

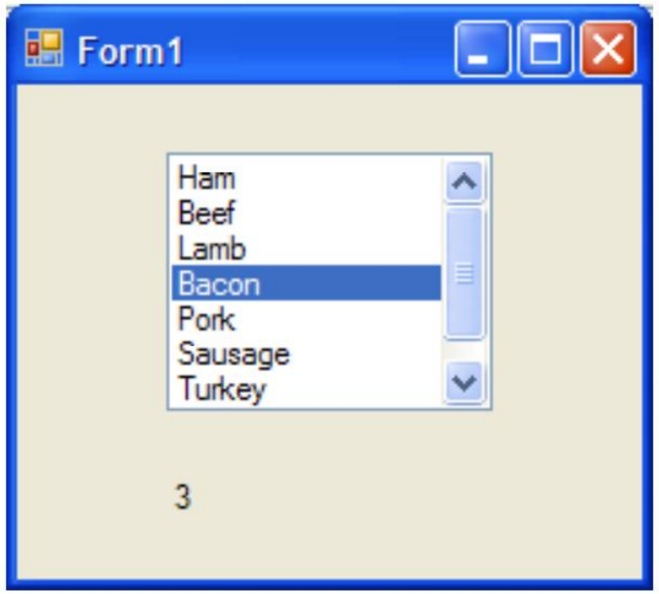
ListBox Events

- The common event for the ListBox is SelectedIndexChanged.
- The index value is zero-based, like an array index.
- The SelectedIndex property is the location of the item that was selected.

SelectedIndex

- Display the SelectedIndex in a label.

```
private void listBox1_SelectedIndexChanged(object sender, EventArgs e) { label1.Text =
listBox1.SelectedIndex.ToString();
}
```



Ham
Beef
Lamb
Bacon
Pork
Sausage
Turkey

3

SelectedIndex

- The SelectedIndex value is -1 if nothing is selected in the listbox.

SetSelected()

- The ListBox.SetSelected() method can be used to set or clear the selected item.

listBox1.SetSelected(int Index, bool);

Index – item to select

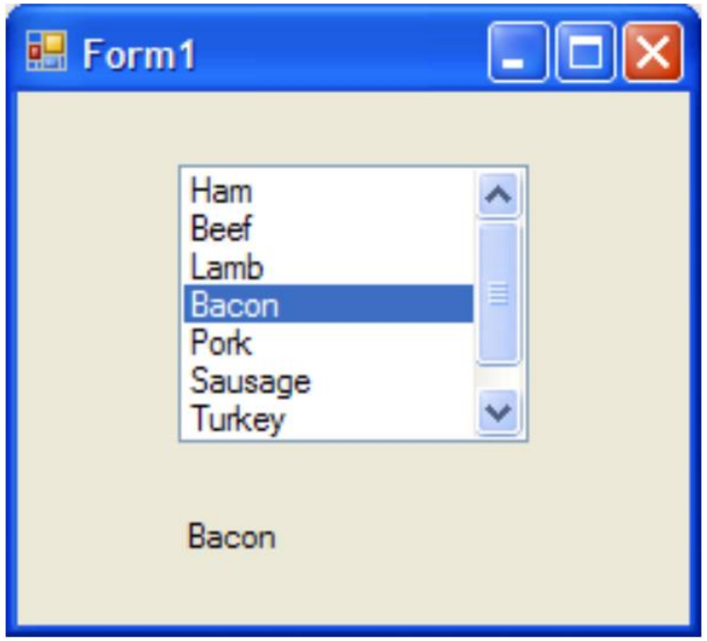
bool – true to select, false to deselect

SelectedItem

- The SelectedItem property returns a value of type **object** that has been selected.
- Usually you convert the object type to a string.
- In .NET programming, this conversion is called **unboxing**.
- The following example assigns the string of the selected object to the label.

SelectedItem

```
private void listBox1_SelectedIndexChanged(object sender,
EventArgs e) {
        label1.Text = listBox1.SelectedItem.ToString();
    }
```



Bacon

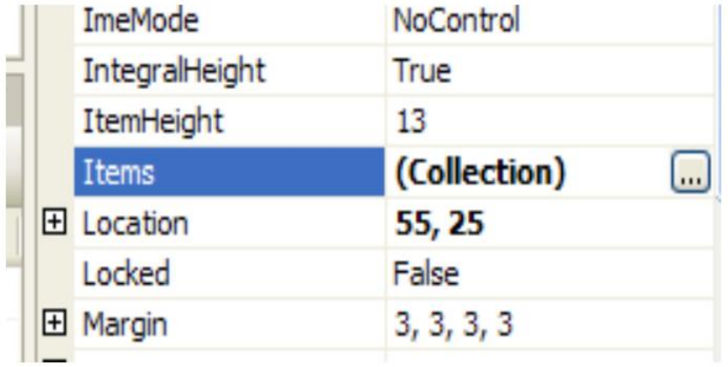
Text

- The Text property of the ListBox returns the selected string.

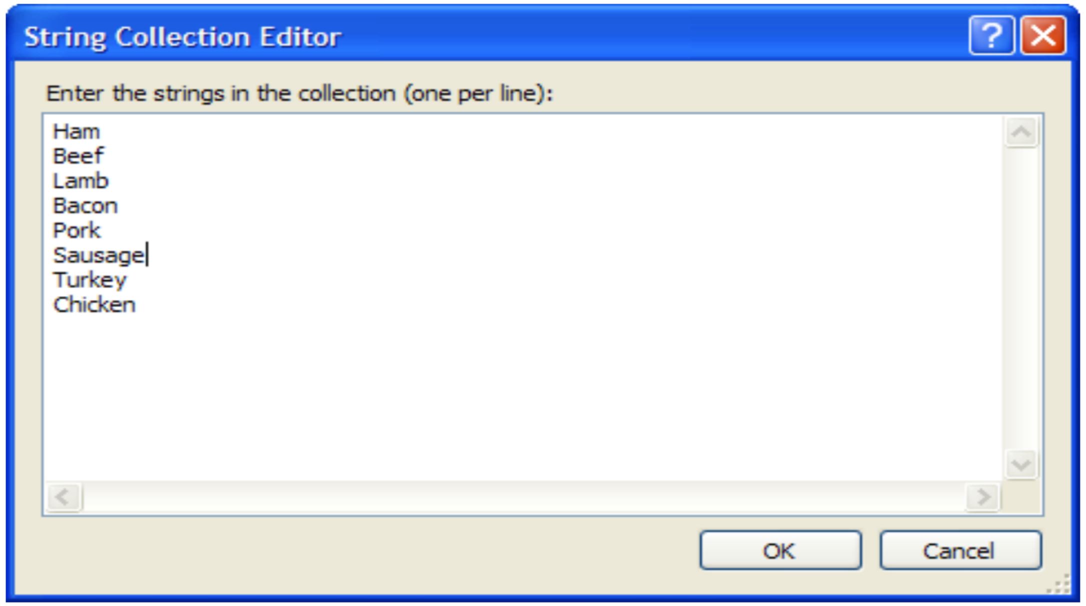
```
private void listBox1_SelectedIndexChanged(object sender, EventArgs e) {
      label1.Text = listBox1.Text;
}
```

ListBox.Items

- Items can be added using the Properties editor when the listbox is being designed.



ImeMode	NoControl
IntegralHeight	True
ItemHeight	13
Items	(Collection)
⊞ Location	55, 25
Locked	False
⊞ Margin	3, 3, 3, 3



String Collection Editor

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

```
Ham
Beef
Lamb
Bacon
Pork
Sausage
Turkey
Chicken
```

ListBox.Items.Add()

- Items can be added during program execution using **Items.Add**

```
listBox1.Items.Add("New Item");
```

ListBox.Items.Count

- Items.Count property is the number of items found in the listbox.

ListBox.Items.Clear()

- The Items.Clear() method removes all items from the listbox.

ListBox.Items.Remove()

- The Items.Remove() method removes the specified object (string) from the listbox.
- If the object is not found, nothing is removed.

```
listBox1.Items.Remove("JD");
```

ListBox.Items.RemoveAt()

- The Items.RemoveAt() method removes the object (string) from the listbox at the index passed as an argument.

```
listBox1.Items.RemoveAt(0);
```

ListBox.FindString()

- The FindString method returns the index of the first matching string in the listbox.
- It returns ListBox.NoMatches if a matching string is not found.

```
int iFoundAt = listBox1.FindString("Ham");
If  (iFoundAt == ListBox.NoMatches)
      label1.Text = "No Ham here!";
```

WebBrowser

- The WebBrowser control enables the user to navigate to web pages from within the form.
- It provides full support for Internet connectivity without any further programming.

WebBrowser Properties

Property	Description
AllowNavigation	Specifies if the WebBrowser control can browse to another page after initially loading.
ScrollBarsEnabled	If true, the WebBrowser may have scrollbars.
URL	Specifies the URL the WebBrowser has navigated to.

WebBrowser Methods

Method	Description
Navigate(url_string)	Loads the web page at the URL passed to Navigate as an argument.
GoBack()	Navigate back to the previous page.
GoForward()	Navigate forward to the next page.
GoHome()	Navigate to the home page specified in Internet settings for the current user.
GoSearch()	Navigate to the search page specified in Internet settings for the current user.

WebBrowser Events

- The common event for the WebBrowser control is DocumentCompleted, which is fired when the HTML page has completed loading.
- The Url property of the WebBrowser will contain the **actual** URL that was used to load the web page

MenuStrip

- The MenuStrip control can be used to add a menu to a form.
- Drag and drop a MenuStrip control to the form, and the control will appear below the form.
- Menu items can be added easily by using the Menu editor

MenuStrip

- A MenuStrip is composed of menu items, and may also include separators, ComboBoxes, and TextBoxes.
- Each menu item is a separate object with its own properties and methods.

MenuItem

- MenuItems are usually positioned below a menu name.
- Each MenuItem may have an image, and a short cut key associated with it.
- A MenuItem may also have a checked or unchecked state.

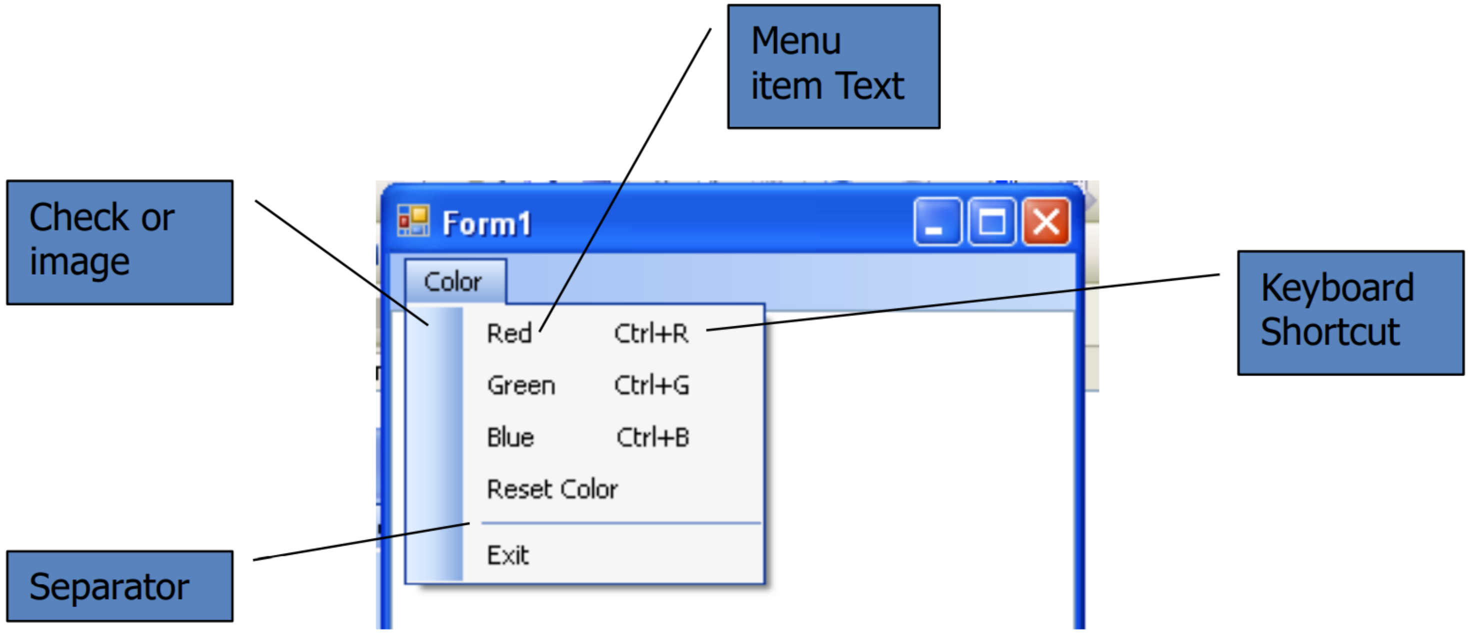
MenuItem Properties

Property	Description
Name	The menu item name. Usually the default name is best.
Checked	Indicates if the menuitem is in the checked state.
CheckOnClick	Indicates if the menu item should toggle its checked state when clicked.
CheckState	The current check state of the menu item. Used for three state menu items.
Enabled	Enables or disables the menuitem.
ShortcutKeys	The shortcut key associated with the menu item. Pressing the shortcut key fires the Click event for the menu item.
ShowShortcutKeys	If true, the shortcut keys are shown in the menu item.

MenuItem Alt Keys

- A MenuItem can be clicked by pressing the Alt key in combination with the underlined character.
- The underlined character is created by placing an ampersand & in front of the desired character in the menuitem name.

MenuItem Properties



MenuItem Events

- The common event for a MenuItem is Click.
- The action desired for the MenuItem is usually performed within the Click handler

MenuItem Event Handler

- The handler shown below will exit the application when the Exit menu item is clicked.

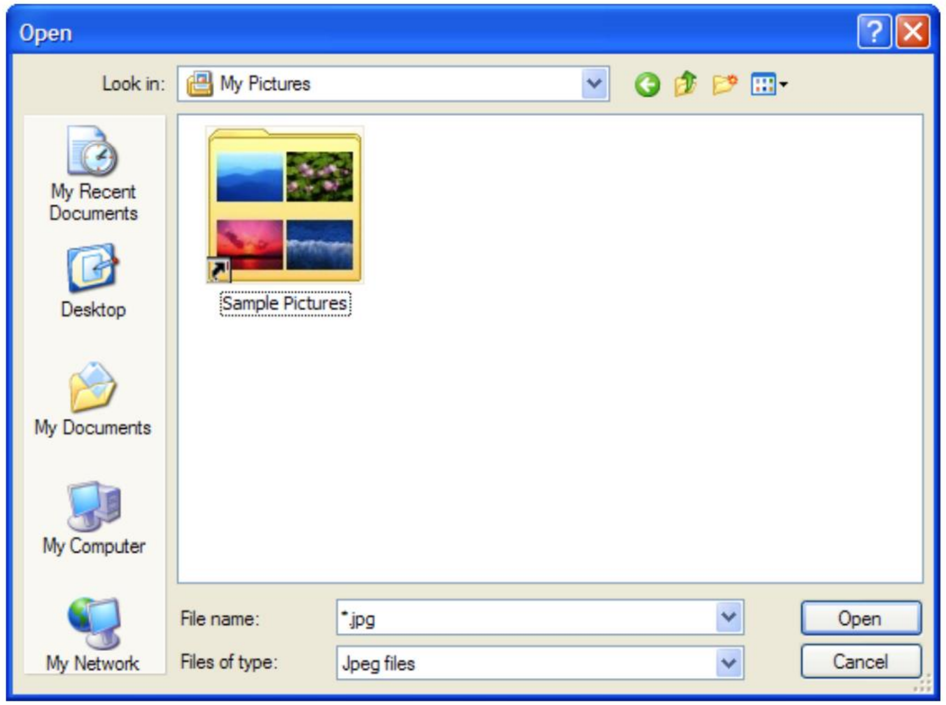
```
private void exitToolStripMenuItem_Click(object sender, EventArgs e) {
Application.Exit();
 }
```

Standard Dialogs

- .NET supports some of the standard system dialogs, such as OpenFile, SaveAsFile, Font, Color etc…
- These dialogs can be added to a Form application by dragging it to the Form from the ToolBox.

OpenFileDialog

- The OpenFileDialog is used to select a file to open.



File name: *.jpg

Files of type: Jpeg files

OpenFileDialog Properties

Property	Description
Name	The dialog name. Usually the default name is best.
AddExtension	Automatically add the file extension.
CheckFileExists	Indicates whether a warning appears if the user specifies a file that does not exist.
FileName	The name of the selected file, including the path.
Filter	The file filters to display in the dialog box, for example "Jpeg files|*.jpg|Gif files|*.gif|All files|*.*"
Title	Title of the dialog box.
SafeFileName	The name of the file without the path.

OpenFileDialog.ShowDialog()

- The ShowDialog method displays the dialog and returns a result.
- The return type is DialogResult.
- If the return value is DialogResult.OK, then a file was selected.
- If the return value is DialogResult.Cancel, then the dialog was closed or Cancel was pressed without selecting a file.

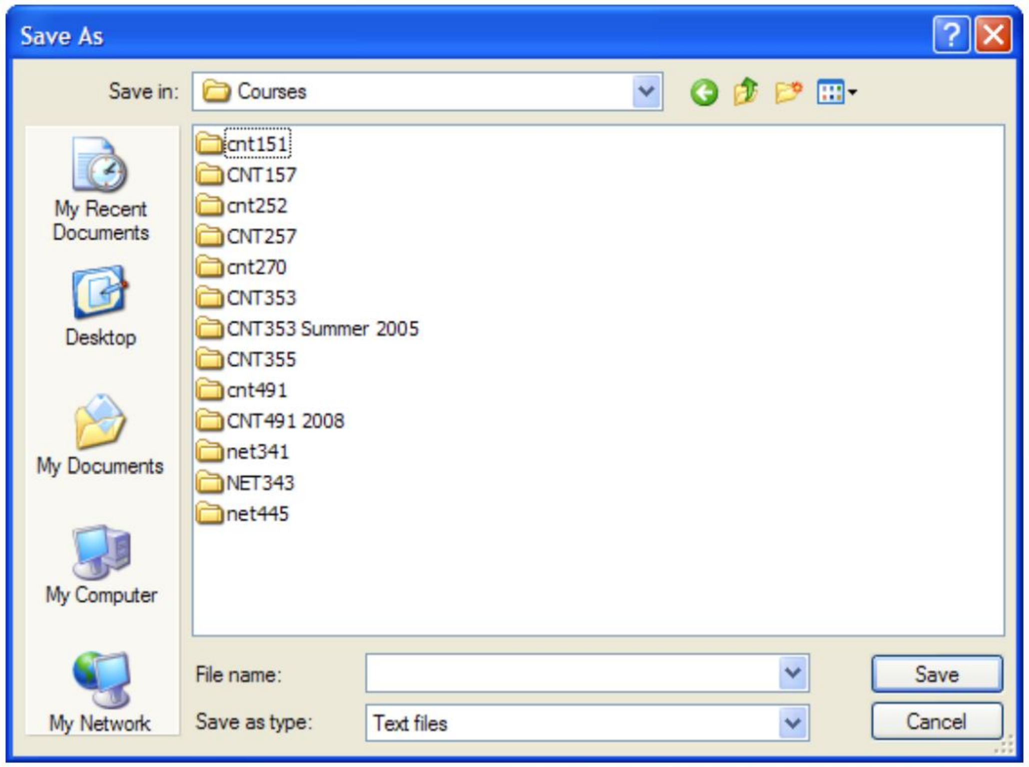
OpenFileDialog.ShowDialog()

- Show the OpenFileDialog, then display the name of the selected file in the form caption.

```
private void openToolStripMenuItem_Click(object sender, EventArgs e)
{
if (openFileDialog1.ShowDialog() == DialogResult.OK)
{
this.Text = openFileDialog1.SafeFileName;
//read in the file here
}
}
```

SaveAsFileDialog

- The SaveAsFileDialog is used to pick a location and file name for saving a file.



SaveAsFileDialog Properties

Property	Description
Name	The dialog name. Usually the default name is best.
AddExtension	Automatically add the file extension.
CheckFileExists	Checks that the specified file exists before returning from the dialog, usually false.
CheckPathExists	Checks that the specified path exists before returning from the dialog.
FileName	The name of the selected file, including the path, or the initial name of the file to display.
Filter	The file filters to display in the dialog box, for example "Jpeg files|*.jpg|Gif files|*.gif|All files|*.*"
Title	Title of the dialog box.

SaveAsFileDialog.ShowDialog()

- The ShowDialog method displays the dialog and returns a result.
- The return type is DialogResult.
- If the return value is DialogResult.OK, then a file was selected or entered by the user.
- If the return value is DialogResult.Cancel, then the dialog was closed or Cancel was pressed without selecting a file.

SaveAsFileDialog.ShowDialog()

- Show the SaveAsFileDialog, then display the name of the selected file in the form caption.

```
private void saveAsToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (saveFileDialog1.ShowDialog() == DialogResult.OK)
      {
              Text = saveFileDialog1.FileName;
              //save your file here
       }
    }
```

Displaying the FileName

- The FileName provided by the standard dialogs include the path to the file.
- If you wish to display the file name to the user, in the form caption, you should remove the path from the name.
- The string.Split() method can be used to separate the file name from the path.

Displaying the FileName

```
private void button1_Click(object sender, EventArgs e)
{
string sFileName = null;
string[] sPathFile = null;
if (saveFileDialog1.ShowDialog() == DialogResult.OK)
{
        sFileName = saveFileDialog1.FileName;
        sPathFile = sFileName.Split('\\’);
        Text = sPathFile[sPathFile.Length - 1];
   }
}
```

Displaying the FileName

- The OpenFileDialog class has a SafeFileName property which can be used to obtain only the filename, without the path.
- The SaveFileDialog class does not have a SafeFileName property, so string.Split() must be used to obtain the filename without the path.

Timer Component

- The Timer component is not a visible control.
- A Timer can be set to fire a TimerTick event at specified intervals.
- Timers are often used to implement animation or periodic events.

Timer Properties

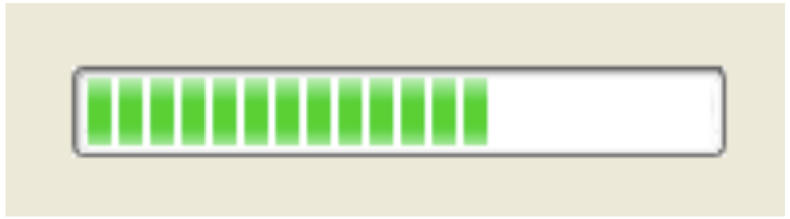
Property	Description
Name	The timer program name.
Enabled	If true, TimerTick events are fired. If false, no TimerTick events occur
Interval	The number of milliseconds to pass between TimerTick events.

Timer Events

- The Timer has only one event, Tick, which occurs once for every timing period if the Timer is enabled.
- Double-click on the Timer component to add a Tick event

ProgressBar

- The ProgressBar is used to display the progress of an operation, such as downloading a web page.



ProgressBar Properties

Property	Description
Name	The ProgressBar program name.
Maximum	The upper bound of the range displayed by the ProgressBar. Integer.
Minimum	The lower bound of the range displayed by the ProgressBar. Integer.
Step	The amount to increment the current value of the control when the PerformStep() method is called.
Style	Block, Continuous, Marquee.
Value	The current value displayed by the ProgressBar as an integer

ProgressBar Events

- The ProgressBar can implement most of the standard form events.
- The common event for the ProgressBar is Click, which is usually not used.

PictureBox

- The PictureBox control can be used to display images of many types.
- It can display jpeg, bitmap, icons, png, and gif files.

PictureBox Properties

Property	Description
Anchor	Gets or sets the edges to which the picturebox is bound.
Dock	Gets or sets the borders to which the picturebox is docked.
Image	Gets or sets the image being displayed.
Size	The size of the picturebox using the Size data type.
SizeMode	How the image is displayed. (Normal, StretchImage, CenterImage, AutoSize, Zoom)

PictureBox Methods

Method	Purpose
Load(path or url)	Loads an image file or from a url. May throw an exception.
PictureBox.Image.Save(path)	Saves the image to a file. May throw an exception
PictureBox.Image.RotateFlip(type)	Rotates or flips the image according to the type value.

Image Properties

Property	Purpose
Height	Height in pixels.
Palette	Color palette used by image.
PhysicalDimension	Dimensions as a Size
PixelFormat	The format used for the pixel.
Size	Size in pixels as a Size object.
Width	Width in pixels.

PictureBox Example

```
private void loadToolStripMenuItem_Click(object sender, EventArgs e)
{
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        try
        {
            pictureBox1.Load(openFileDialog1.FileName);
            this.Size = pictureBox1.Image.Size;
        }
        catch
        {
            MessageBox.Show("Could not load the file.", "PictureBoxPlay");
        }
    }
}
```

ImageList

- A component that can hold a series of images.
- The images become embedded into the resulting executable file.
- Can be used for animation by selecting a series of images for display.

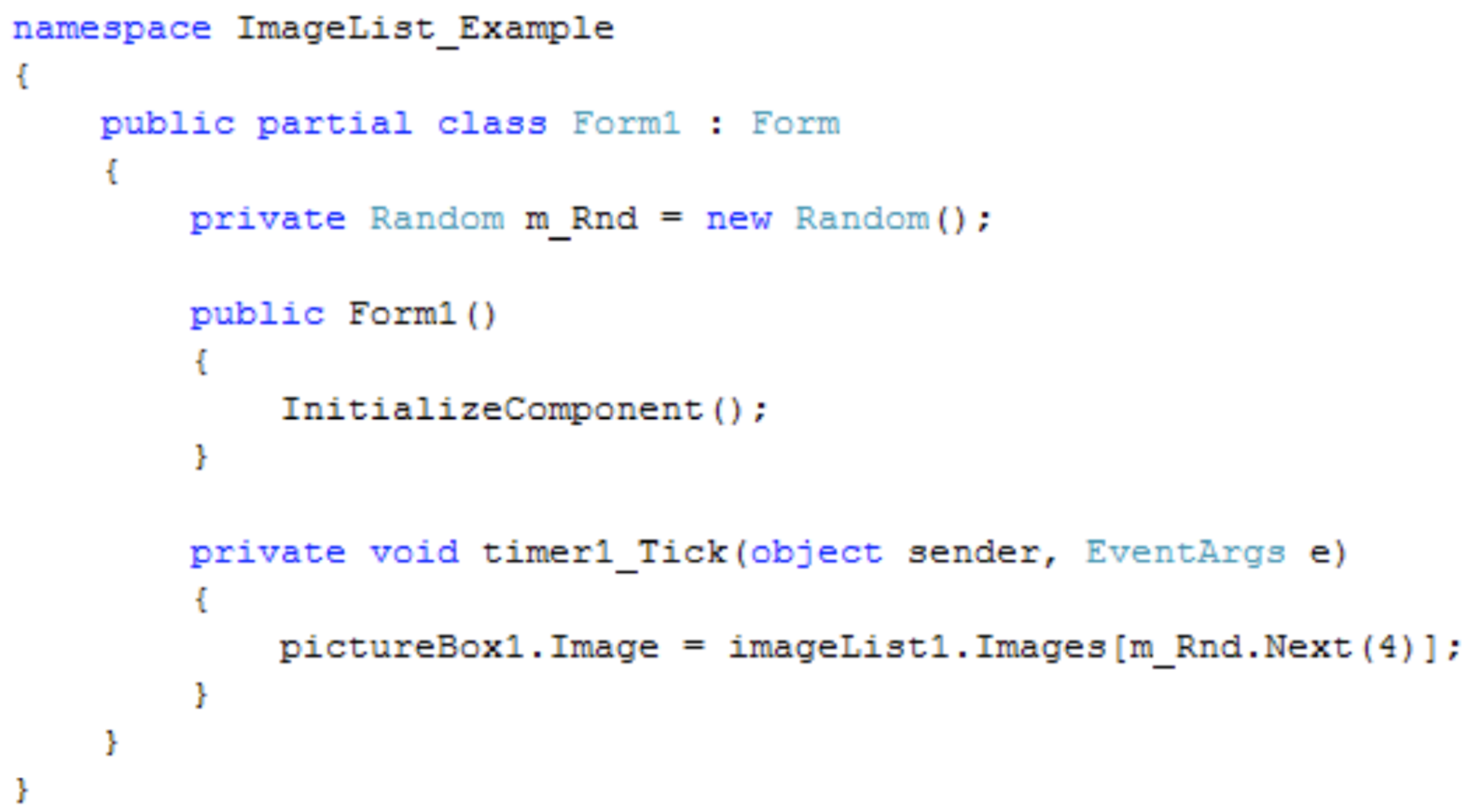
ImageList Properties

Property	Purpose
Name	The program name for the control.
ColorDepth	4, 8, 16, 24, 32 bits used per pixel for the color.
Images	A collection of the images. Images are embedded within the application executable.
ImageSize	The image size in pixels. Must be less than 256.
TransparentColor	The color used to represent a transparent pixel.
Images.Count	The number of images stored.
Images.Clear()	Clears the images from the ImageList.
Images.IsReadOnly	True if the collection is readonly.

ImageList

- Images stored in the ImageList can be accessed using an index similar to an array.
- The following example loads a random image in a PictureBox every second using a Timer

ImageList Example



```
namespace ImageList_Example
{
    public partial class Form1 : Form
    {
        private Random m_Rnd = new Random();

        public Form1()
        {
            InitializeComponent();
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            pictureBox1.Image = imageList1.Images[m_Rnd.Next(4)];
        }
    }
}
```

SoundPlayer

- A SoundPlayer can be used to play sounds asynchronously.
- Available in System.Media namespace.
- Can be constructed with the desired sound to play.

SoundPlayer Properties

Property	Purpose
IsLoadCompleted	True if the sound file has completed loading.
LoadTimeout	The time, in milliseconds, in which the load of a sound file must be completed before timeout.
SoundLocation	The location of the sound file as a path or URL.

SoundPlayer Methods

Method	Description
Load()	Loads a sound file from a path or URL.
SoundPlayer()	Constructor can load a sound file or use an embedded resource.
Play()	Plays the sound asynchronously.
PlaySynch()	Plays the sound synchronously. Program stops until sound completed.
PlayLooping()	Continuously plays the sound file.
Stop()	Stop playing the sound.

KeyBoard Events

- Keyboard input is provided to an application as keyboard events.
- The argument of the keyboard event, e, can be used to determine the key that was pressed, and the state of the keyboard.

KeyBoard Events

Event	Description
KeyDown	Occurs when a key is first pressed.
KeyPress	Occurs when the control has focus and the key is pressed then released. Used to input the Unicode value of the key.
KeyUp	Occurs when a key is released.
PreviewKeyDown	Occurs before the KeyDown event when a key is pressed with the focus of this control. Used to intercept a key press before it is processed.

KeyBoard Events

- When a standard Unicode key is pressed then released, the following events occur:
    1. PreviewKeyDown
    2. KeyDown
    3. KeyPress
    4. KeyUp

KeyBoard Events

- When a nonstandard Unicode key is pressed (function key or arrow) then released, the following events occur:

    1. PreviewKeyDown
    2. KeyDown
    3. KeyUp

KeyBoard Events

- If a key is pressed and held, then the sequence of events repeats automatically.
- Keyboard events are directed to the control that has the current focus.
- The argument type for the event args is unique for each event.

KeyEventArgs

- KeyEventArgs are passed to the event handler for both KeyDown and KeyUp events.
- The members of the object can be used to determine the key, and the state of Shift, Alt, and Ctrl.

KeyEventArgs

Member	Description
Alt	If true, the Alt key is currently pressed.
Control	If true, the Ctrl key is currently pressed.
KeyCode	The identifier of the key pressed.
KeyData	The identifier of the key pressed combined with the Alt, Ctrl, and Shift states.
KeyValue	The integer scancode of the key that was pressed.
Modifiers	The state of Alt, Ctrl, and Shift.
Shift	If true, Shift was pressed.
Handled	You can set this to true to prevent the key from being passed on to the underlying control.

KeyPressEventArgs

- The KeyPressEventArgs is an object that contains information about the KeyPress event.
- Recall that a KeyPress event only occurs for a key that is a valid Unicode character

KeyPressEventArgs

Member	Description
Handled	You can set this to true to prevent the key from being passed on to the underlying control.
KeyChar	The Unicode value of the key that was pressed.

PreviewKeyDownEventArgs

- The PreviewKeyDownEventArgs is an object that provides information about the PreviewKeyDown event.
- Recall that the PreviewKeyDown event occur prior to Windows processing the key press.

PreviewKeyDownEventArgs

Member	Description
Alt	If true, the Alt key is currently pressed.
Control	If true, the Ctrl key is currently pressed.
KeyCode	The identifier of the key pressed.
KeyData	The identifier of the key pressed combined with the Alt, Ctrl, and Shift states.
KeyValue	The integer scancode of the key that was pressed.
Modifiers	The state of Alt, Ctrl, and Shift.
Shift	If true, Shift was pressed.

Form.KeyPreview

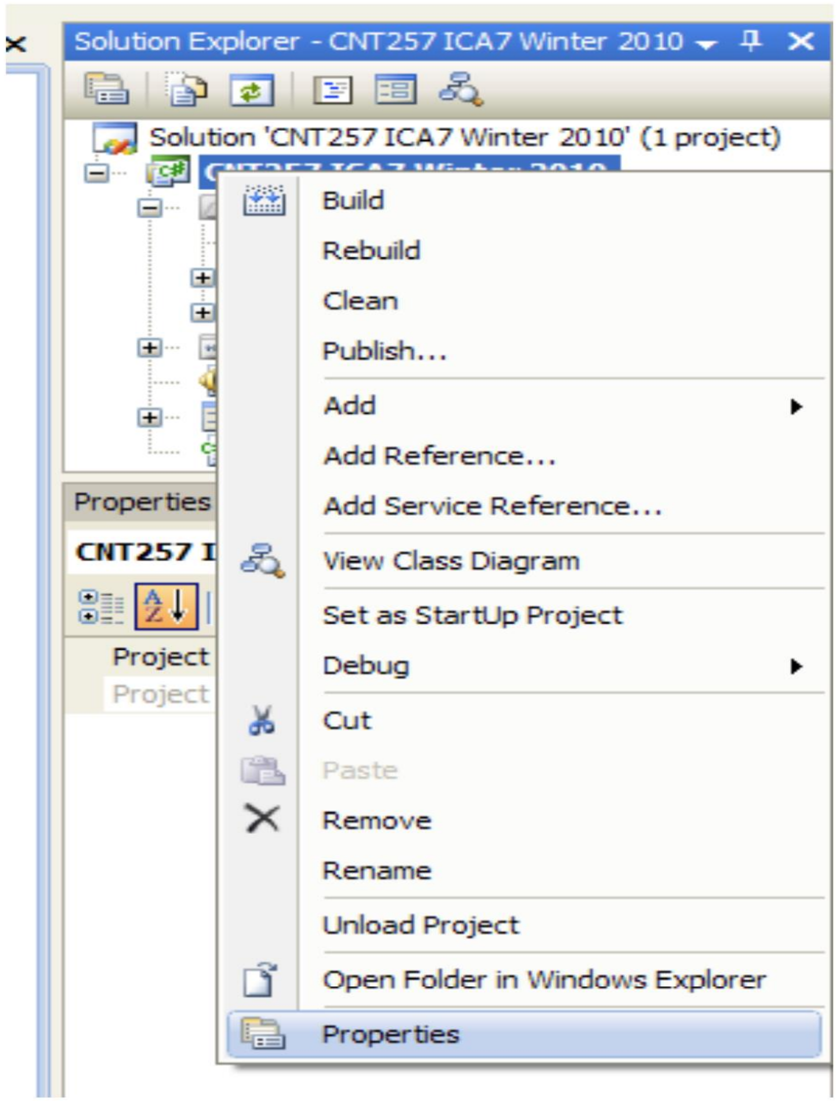
- Normally keyboard events are sent to the control with the focus, and not to the form.
- If it is desired to have the form process keyboard messages, change the Form property KeyPreview to true.
- Keyboard messages will now be sent to the control and to the form.

Adding Embedded Resources

- Resources, such as sounds or images, can be embedded within the executable file for quick access.
- This will make it easier to install the program at a new location.
- Requires access to the resources tab (and file) of the project solution.

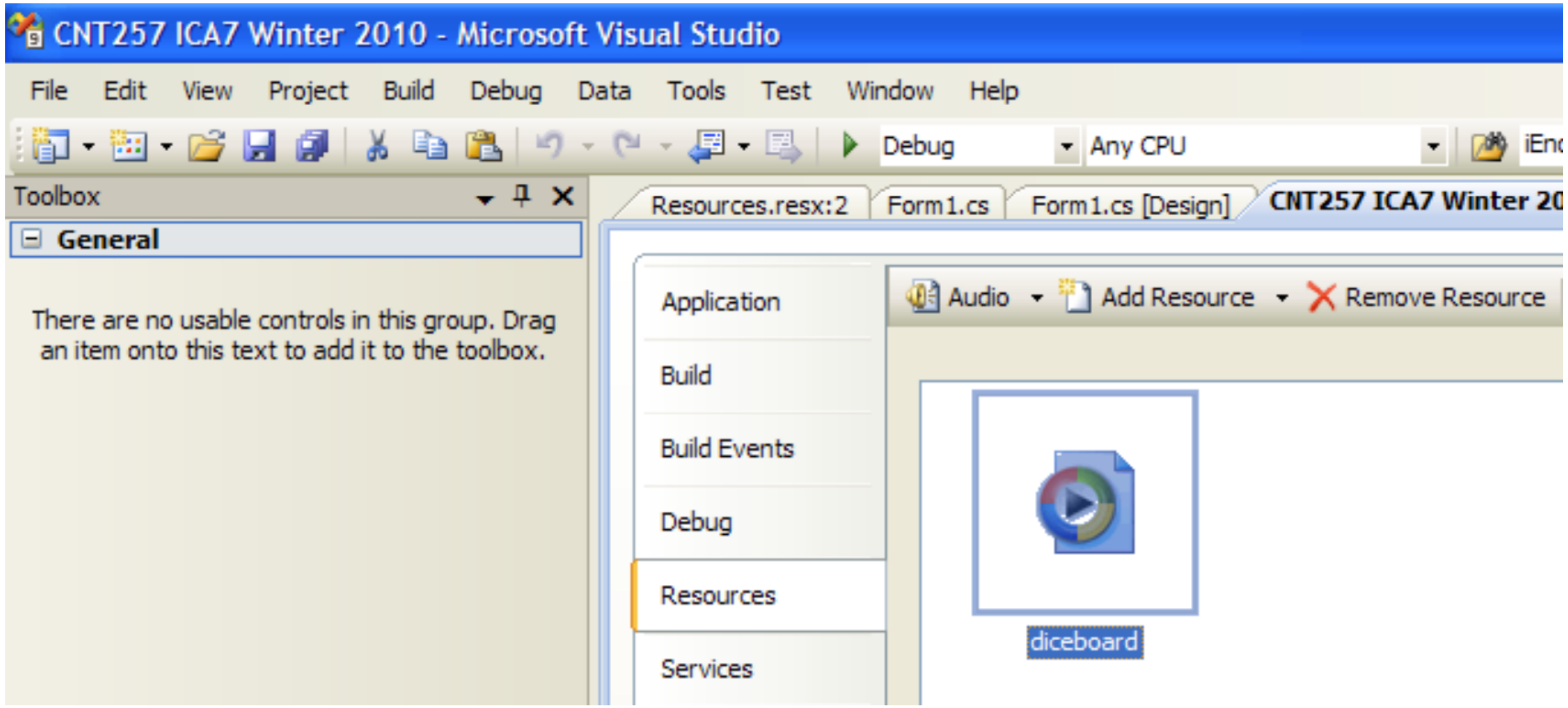
Adding Embedded Resources

- Right-click on the project, and select Properties at the bottom of the menu



Adding Embedded Resources

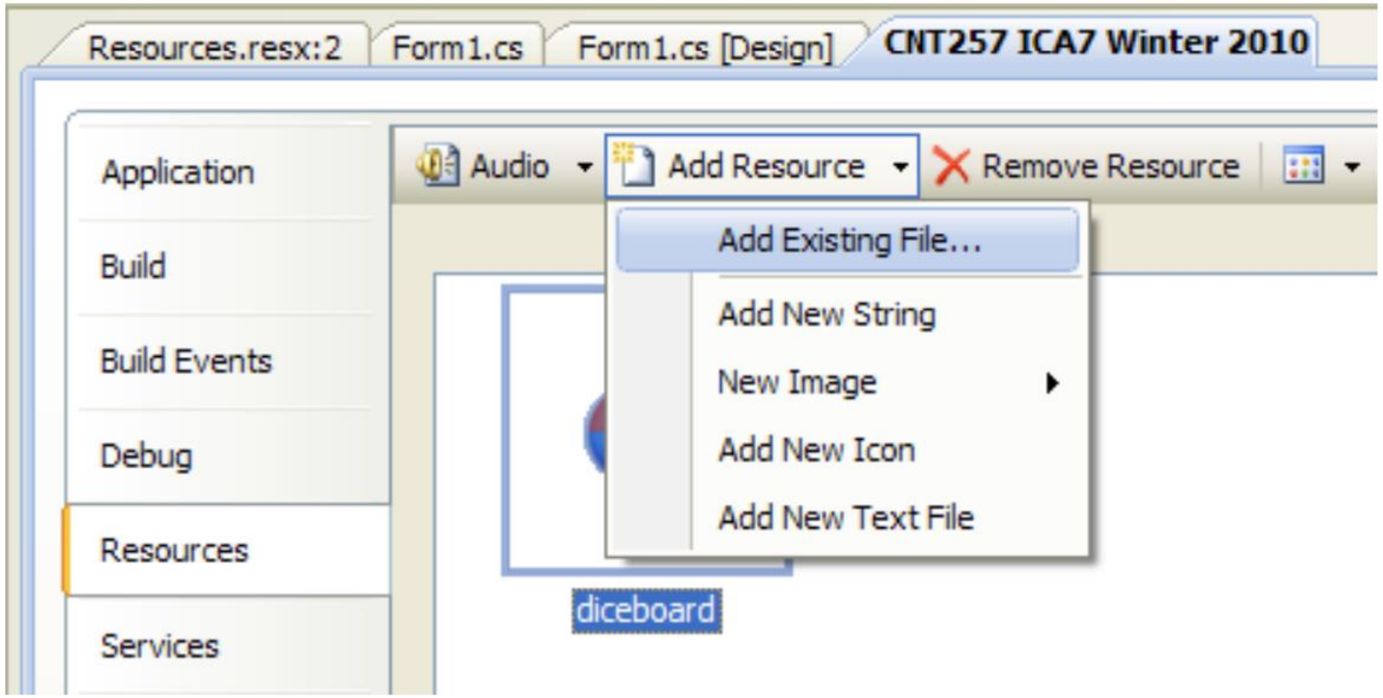
- Select Resources in the tabs.



There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.

Adding Embedded Resources

- Use "Add Resource" then select "Add Exsiting File…" if it already exist



Adding Embedded Resources

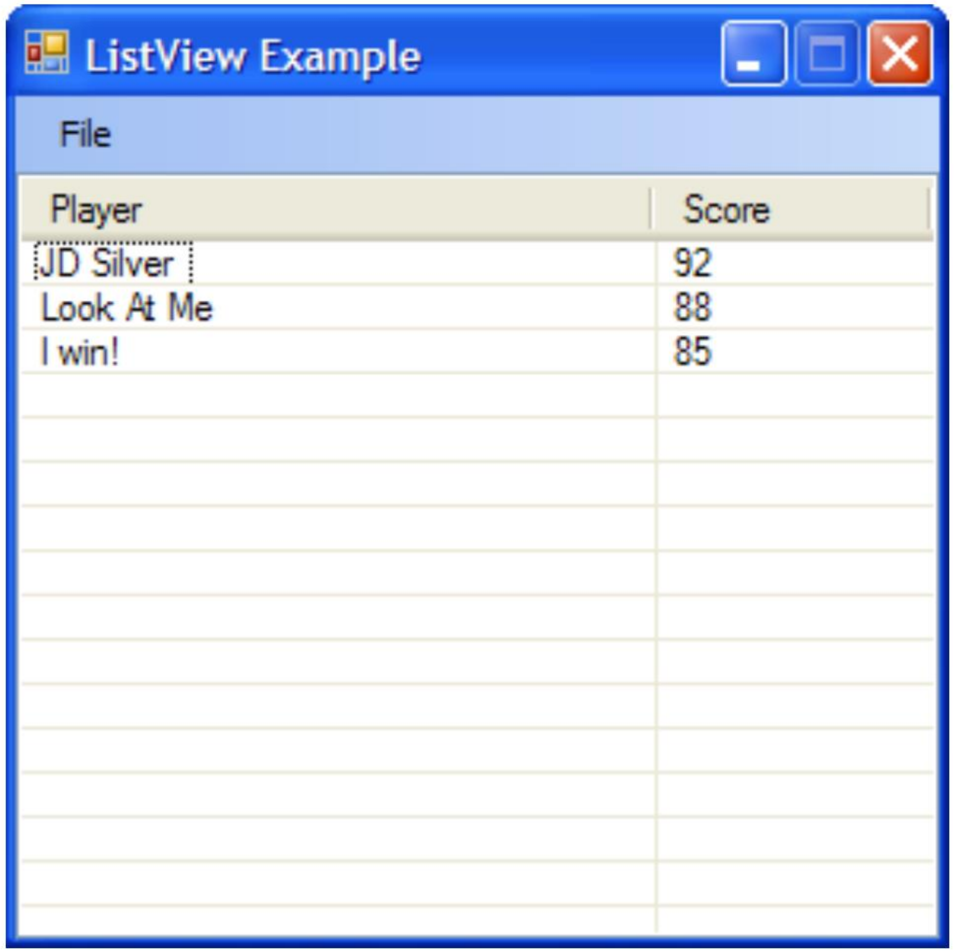
- Select the file and add it to the resources.
- The resource will be shown with a name.
- The path to the resource is

ProjectNamespace.Properties.Resources.ResourceName

ListView

- The ListView control is useful to display data in columns.
- The fields of a structure or class can be displayed in columns, with each row representing a single structure or object.
- The columns can have a heading, which is used to describe the contents of the column.

ListView



Player	Score
JD Silver	92
Look At Me	88
I win!	85

ListView Properties

Property	Purpose
View	The viewing mode used. For this course, we will use Details mode. Other modes include large and small images, list or tile.
Items	The ListViewItems stored in the ListView.
Columns	The collection of columns to be used in the ListView.
Gridlines	Turns on or off the gridlines to be displayed.
CheckBoxes	Turns on or off checkboxes for each ListViewItem.
FullRowSelect	If true the user can select a full row instead of a single cell in a column.
MultiSelect	If true the user can select multiple rows (or cells) in the ListView

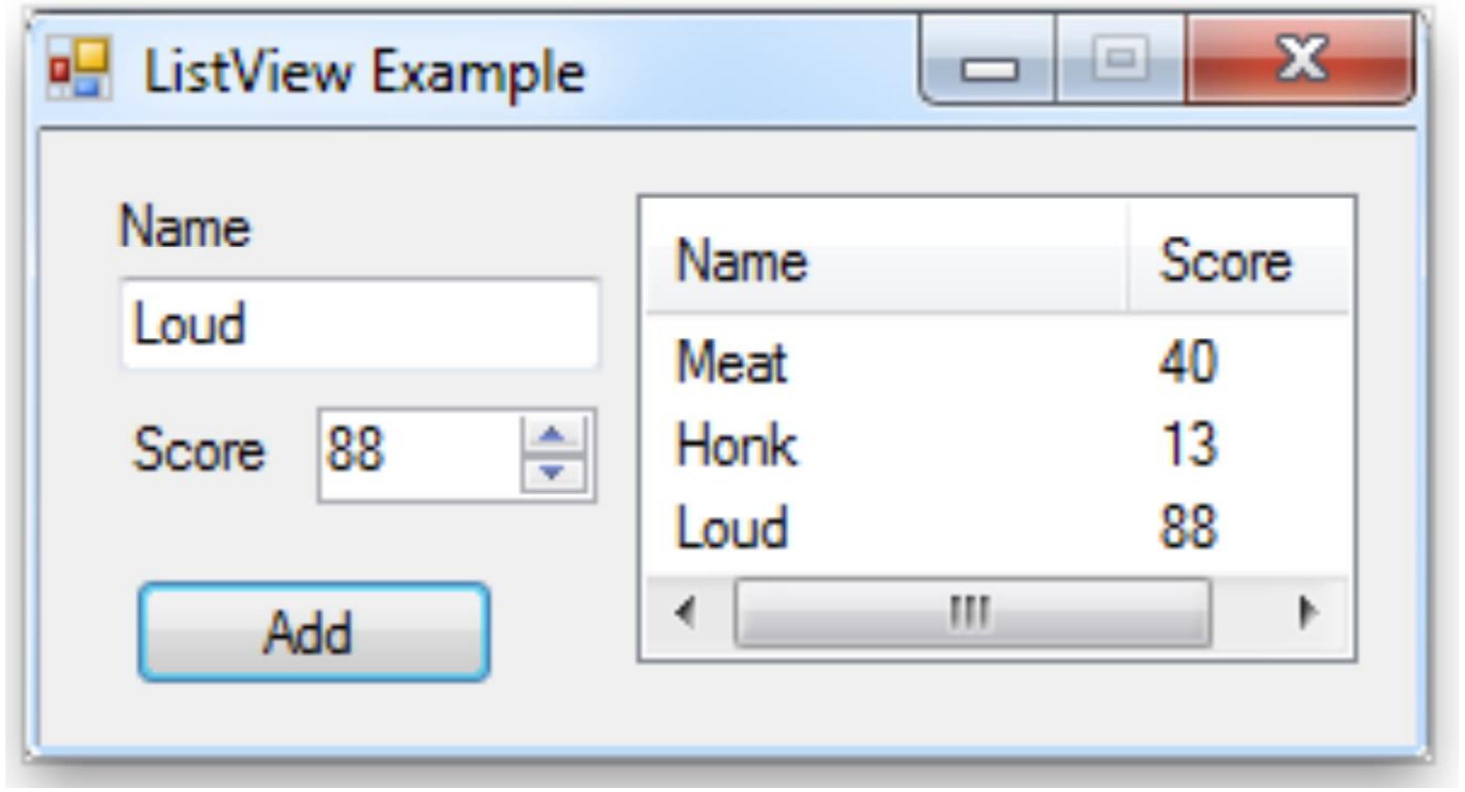
ListView

- Adding an item to a ListView requires the creation of a ListViewItem, which will add a row to the ListView.
- The ListViewItem will be given the data for the first column in the row to be displayed.
- Data for the rest of the columns are added by adding SubItems to the ListViewItem.

ListView

- The following program will display the name in the first column, and a score in the second column when the Add button is pressed.
- The name will be passed to a newly created ListViewItem, and the score will be added as a SubItem to the ListViewItem.
- The ListViewItem will then be added to the ListView.

ListView



ListView Example

Name

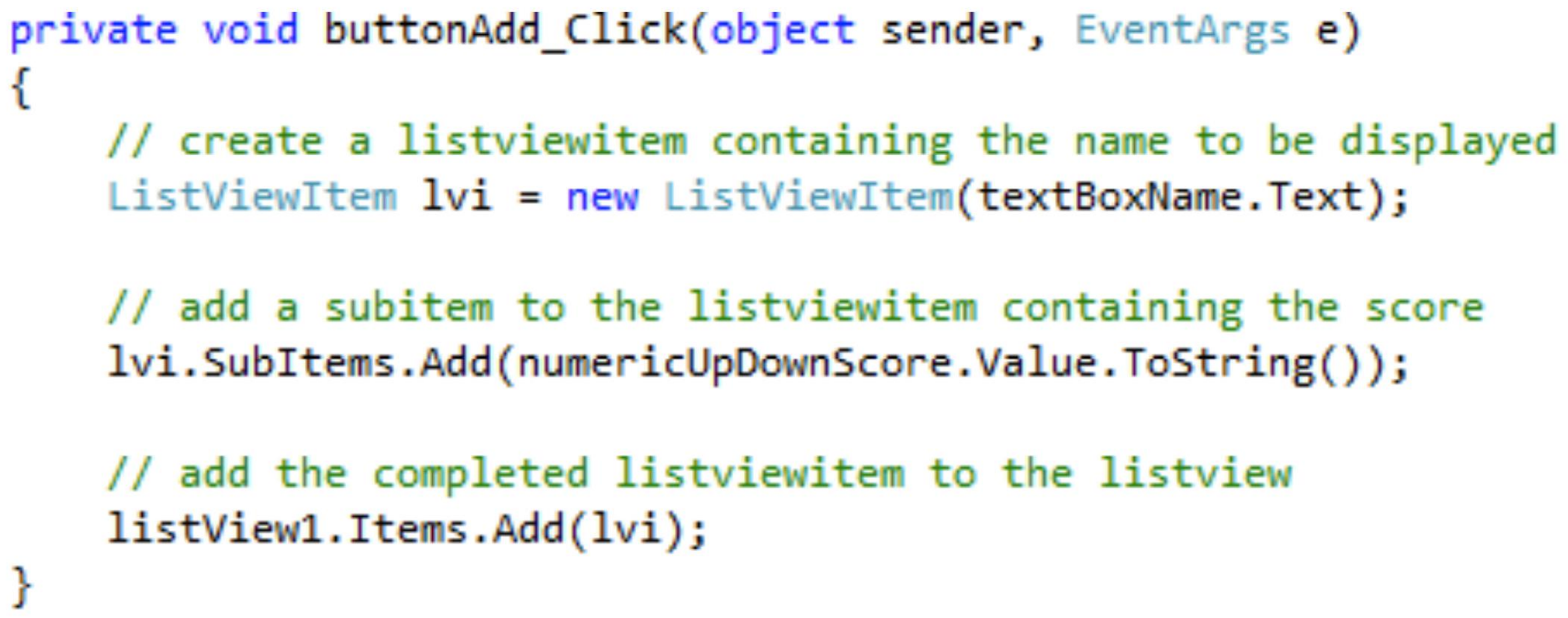
Loud

Score 88

Add

Name	Score
Meat	40
Honk	13
Loud	88

ListView



```
private void buttonAdd_Click(object sender, EventArgs e)
{
    // create a listviewitem containing the name to be displayed
    ListViewItem lvi = new ListViewItem(textBoxName.Text);

    // add a subitem to the listviewitem containing the score
    lvi.SubItems.Add(numericUpDownScore.Value.ToString());

    // add the completed listviewitem to the listview
    listView1.Items.Add(lvi);
}
```

ListView

- The user can select a row or a cell within the ListView.
- If the FullRowSelect property is true, then an entire row will be selected, rather than a single cell.
- If the MultiSelect property is true, then the user can select multiple rows in the ListView.

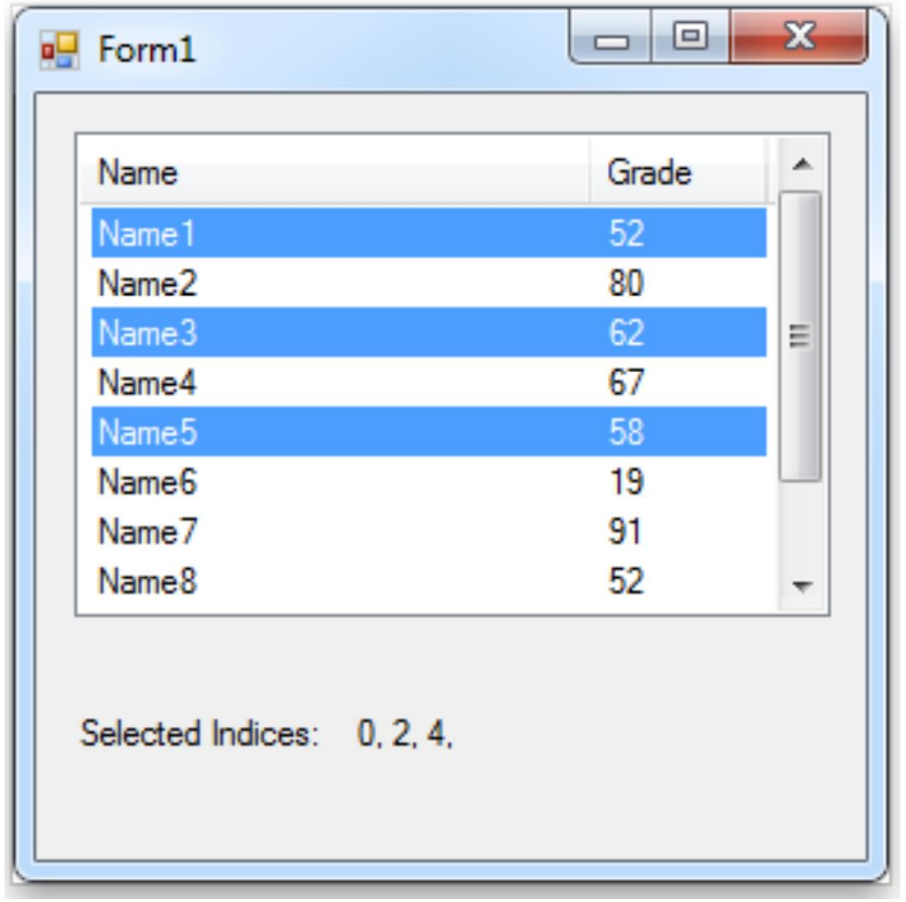
ListView

- When the user selects a row, or multiple rows, the ListView sends a SelectedIndexChanged event.
- The rows that were selected are provided by the SelectedIndices property as a ListView.SelectedIndexCollection.
- The collection contains integer values, which are indexes to the locations selected. ListView
- In the following example, a ListView is set u

ListView

- In the following example, a ListView is set up in Detail view mode.
- Each line of the ListView displays a student name and a random grade.
- The ListView is filled with data when the form is loaded.

ListView

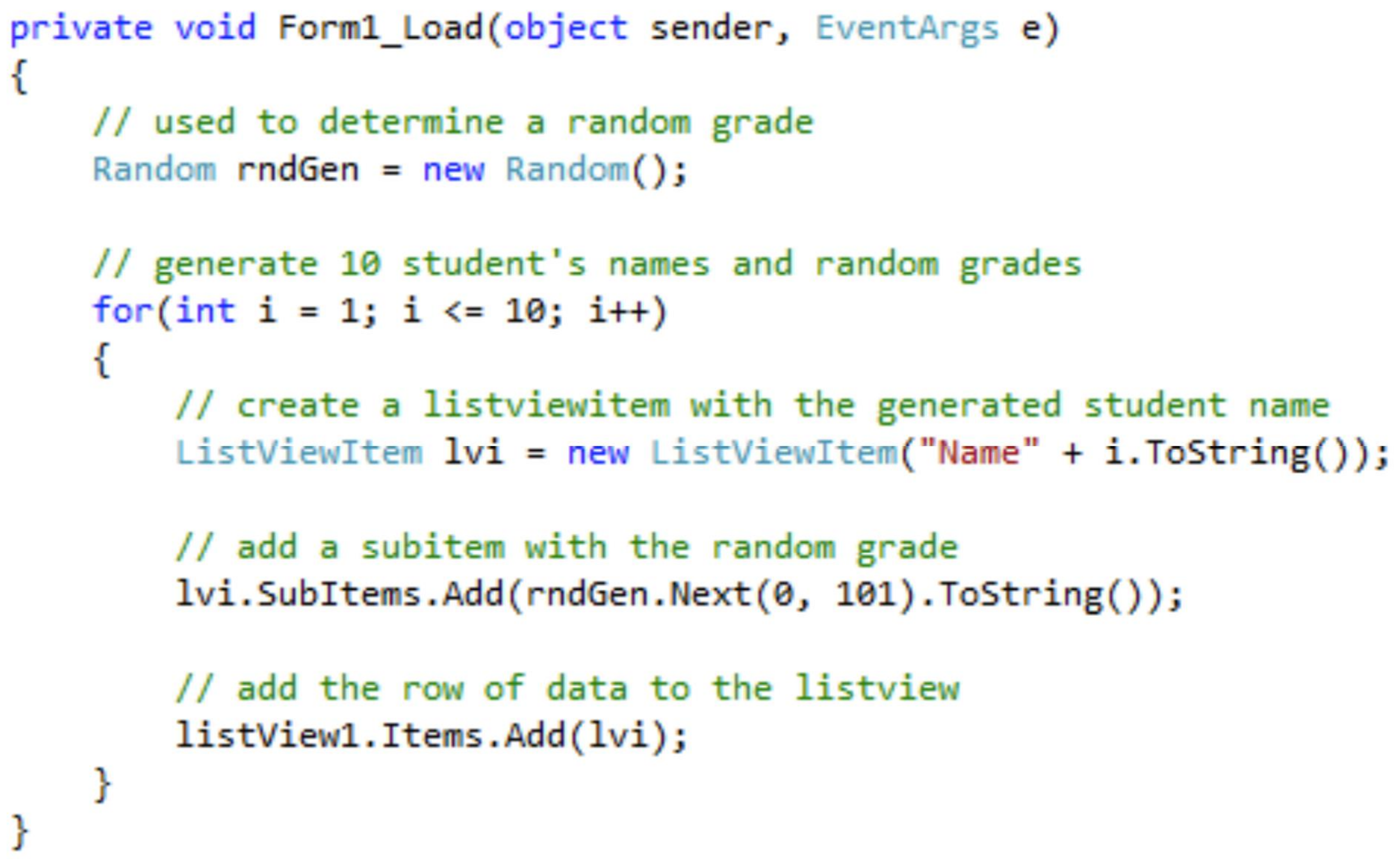


Form1

Name	Grade
Name1	52
Name2	80
Name3	62
Name4	67
Name5	58
Name6	19
Name7	91
Name8	52

Selected Indices: 0, 2, 4,

ListView

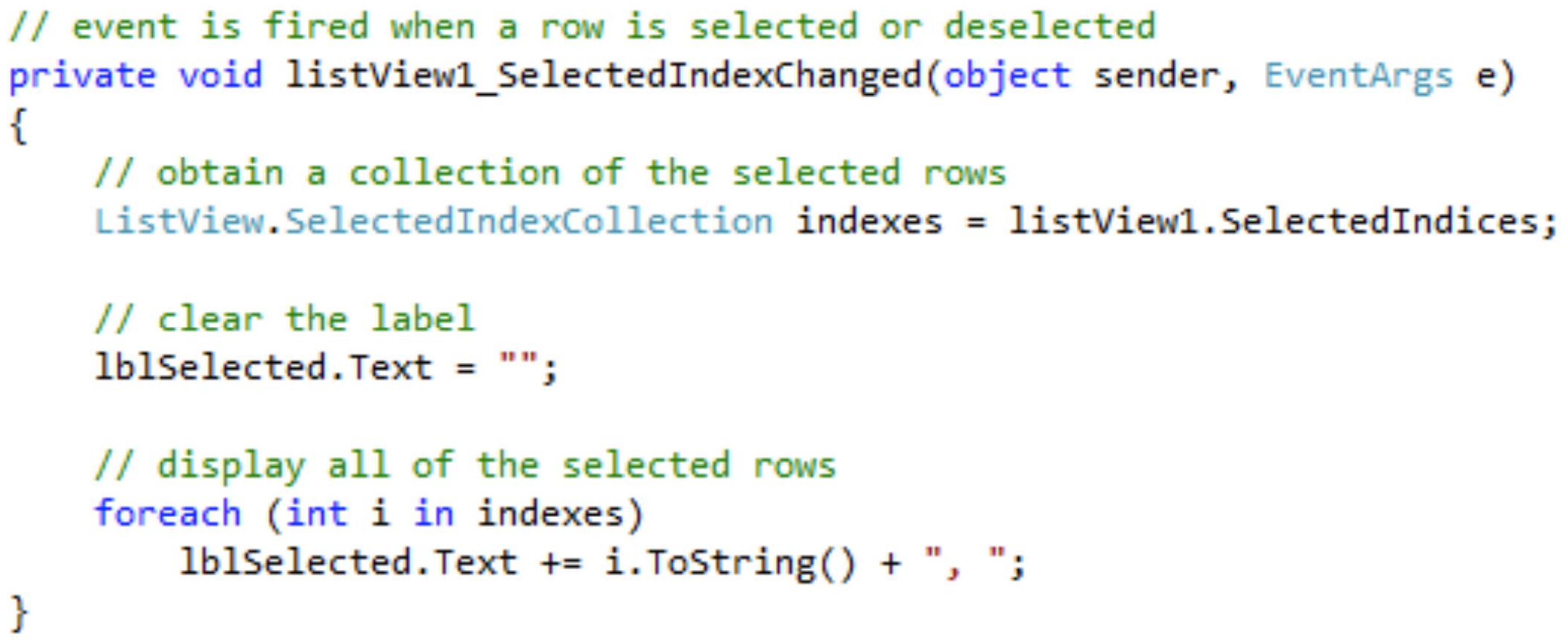


```
private void Form1_Load(object sender, EventArgs e)
{
    // used to determine a random grade
    Random rndGen = new Random();

    // generate 10 student's names and random grades
    for(int i = 1; i <= 10; i++)
    {
        // create a listviewitem with the generated student name
        ListViewItem lvi = new ListViewItem("Name" + i.ToString());

        // add a subitem with the random grade
        lvi.SubItems.Add(rndGen.Next(0, 101).ToString());

        // add the row of data to the listview
        listView1.Items.Add(lvi);
    }
}
```

ListView

- When rows are selected in the ListView a SelectedIndexChanged event is fired.
- The SelectedIndices property provides a SelectedIndexCollection of integer indexes specifying the selected rows.
- In the example, the indexes of the selected rows are displayed in a label.

ListView



```
// event is fired when a row is selected or deselected
private void listView1_SelectedIndexChanged(object sender, EventArgs e)
{
    // obtain a collection of the selected rows
    ListView.SelectedIndexCollection indexes = listView1.SelectedIndices;

    // clear the label
    lblSelected.Text = "";

    // display all of the selected rows
    foreach (int i in indexes)
        lblSelected.Text += i.ToString() + ", ";
}
```