Programming with Microsoft Visual Basic.NET

Lesson 3

What have we learnt in last lesson?

- Introduction to relational and logical operators
- Using If…Then…Else and Select Case for decisions making
- Using Do Loop and For Loop for iteration

Introduction to Control

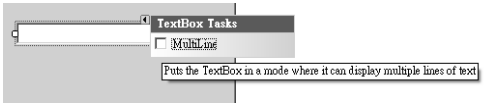
TextBox, PictureBox, Label, Button, ListBox, ComboBox, Checkbox, Radio Button

Control Properties

- All controls have a properties - these change how the control looks and reacts
  - Examples: Font size and color, its location on the form, text justification, replace text with a specific image
- Each control type has its own list of properties
  - Example: Button and label properties are different
- Properties are edited using the **Properties** Window
  - Properties window will show current properties for the control that the cursor is on
  - All controls have a name that is used to refer to them from within the code

TextBox Control

- Text boxes are used to get input from the user or to display text.
- The TextBox control is generally used for editable text, although it can also be made read-only.
  - Text boxes can display multiple lines, wrap text to the size of the control, and add basic formatting.
  - The TextBox control allows a single format for text displayed or entered in the control.
- The text displayed by the control is contained in the Text property. By default, you can enter up to **2,048** characters in a text box. If you set the ***Multiline*** property to true, you can enter up to **32 KB** of text.

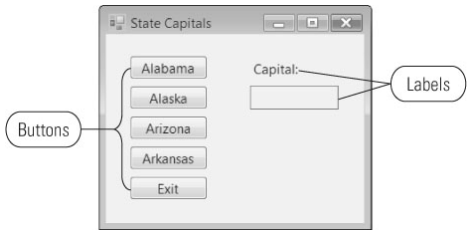


Masked TextBox Control

- You can bind data to a MaskedTextBox control just as you can to any other control.
- However, if the format of your data in the database does not match the format expected by your mask definition, you will need to reformat the data.

Label Control

- The Label controls are used to display text or images that cannot be edited by the user.
- They are used to identify objects on a form — to provide a description of what a certain control will do if clicked, for example, or to display information in response to a run-time event or process in your application.
  - For example, you can use labels to add descriptive captions to text boxes, list boxes, combo boxes, and so on.



Button Control

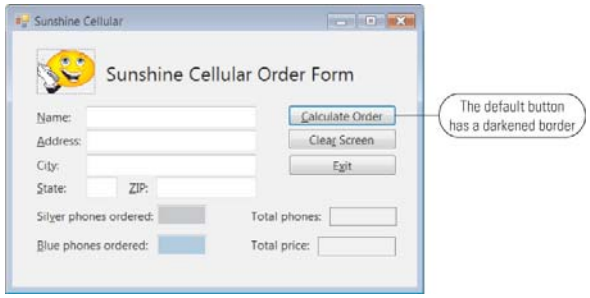
- The Button control allows the user to click it to perform an action.
  - When the button is clicked, it looks as if it is being pushed in and released. Whenever the user clicks a button, the **Click** event handler is invoked. You place code in the Click event handler to perform any action you choose.
- The text displayed on the button is contained in the **Text** property.
  - If your text exceeds the width of the button, it will wrap to the next line. However, it will be clipped if the control cannot accommodate its overall height.
  - The appearance of the text is controlled by the **Font** property and the **TextAlign** property.
- The Button control can also display images using the Image and ImageList properties.

Designating Default and Cancel Buttons

- **Default Button:** activated when user presses Enter key
  - ***AcceptButton*** property:
    - A form property that designates the name of the default button
    - Only one per form
- **Cancel Button:** activated when user presses Esc key
  - ***CancelButton*** property:
    - Form property that designates name of cancel button
    - Only one per form

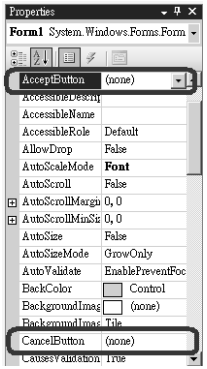
Accept Button in Form Properties

- The program will carry out the event handler processing associated with the **AcceptButton** if the user clicks the button or if the user presses the **[Enter]** key
- Select the form, then click the **AcceptButton** property name and select the default button from the list.



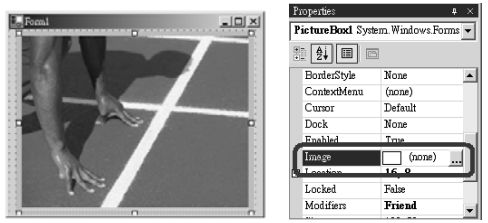
Cancel Button in Form Properties

- When the user presses the [ESC] key, the event handler processing for the button identified as the Cancel button will be executed.
- Select the form, then click the **CancelButton** property name and select the cancel button from the list.



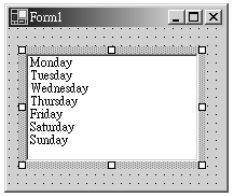
Picture Box Control

- A **PictureBox** control can hold an image, which can be an icon (.ico), a GIF (.gif), a JPEG (.jpg), or a Bitmap (.bmp).
- Picture box control: used to display an image on a form
  - ***Image*** property: specifies the image to display
  - ***SizeMode*** property: handles how the image will be displayed
  - ***Settings*** property: Normal, StretchImage, AutoSize, CenterImage, or Zoom



List Box Control

- A ListBox control displays a list of items from which the user can select one or more.
- When the ***MultiColumn*** property is set to true, the list box displays items in multiple columns and a horizontal scroll bar appears.
- When ***ScrollAlwaysVisible*** is set to true, the scroll bar appears regardless of the number of items.
- The ***SelectionMode*** property determines how many list items can be selected at a time.



ListBox Manipulation

- To add the new to the item to the end of the list box:
  - Listbox.Items.Add("*NewItem*")
- To remove all items in the list box:
  - ListBox.Items.Clear( )
- To remove a specified items in the list box :
  - Items.Remove("*RemoveItem*")
- To identify the select item in the list box
  - The *SelectedIndex* property returns an integer value that corresponds to the first selected item in the list box.
  - The ***SelectedItem*** return the string for user selected

The SelectedItem and SelectedIndex Properties

- SelectedItem property:
  - Contains the value of the selected item in the list
  - If nothing is selected, it contains the empty string
- SelectedIndex property:
  - Contains the index of the selected item in the list
  - If nothing is selected, it contains the value -1
- Default list box item:
  - The item that is selected by default when the interface first appears

Common Properties for ListBox Control

Properties of the ListBox Class	Description
	Inherits properties, methods, and events from the Control class
Items	Gets the items of the list box
Multicolumn	Indicates whether the list box supports multiple columns
SelectedIndex	Indicates the index of the currently selected item in the list box
SelectedIndices	Indicates the indices of all the currently selected items in the list box
SelectedItem	Identifies the currently selected item in the list box
SelectedItems	Identifies all the currently selected items in the list box
SelectionMode	Indicates whether the user can select a single item or multiple items from the list box
Sorted	Indicates whether the items in the list box are sorted
Methods of the ListBox Class	
ClearSelected	Deselects all items in the list box
GetSelected	Indicates whether the specified item is selected in the list box
SetSelected	Selects (or deselects) the specified item in the list box
Common Event	
SelectedIndexChanged	Occurs when the item selected in the list box changes

Example: Determine the Number of Items in List Box

- Items.Count property: stores the number of items in a list box. Count value is always one higher than the highest index in the list box

DETERMINE THE NUMBER OF ITEMS IN A LIST BOX

Syntax

object.**Items.Count**

Example 1

```
Dim numberOfAnimals As Integer
numberOfAnimals = animalListBox.Items.Count
```

assigns the number of items contained in the animalListBox to an Integer variable

Example 2

```
Dim numCodes As Integer
Dim index As Integer
numCodes = codeListBox.Items.Count
Do While index < numCodes
    MessageBox.Show(Convert.ToString(codeListBox.Items(index)))
    index = index + 1
Loop
```

displays the items contained in the codeListBox in message boxes; you also can use the statement MessageBox.Show(codeListBox.Items(index).ToString)

Example: Access an Item in List Box

ACCESS AN ITEM IN A LIST BOX

Syntax

object.**Items(***index***)**

Example 1

```
Dim animalType As String = String.Empty
animalType = Convert.ToString(animalListBox.Items(0))
```

assigns the first item in the animalListBox to a String variable; you also can use the statement animalType = animalListBox.Items(0).ToString

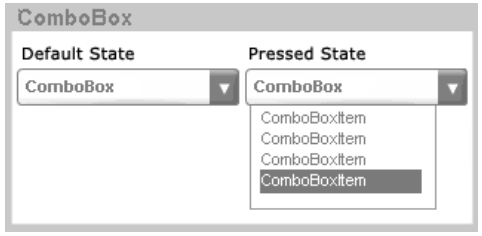
Example 2

```
Dim code As Integer
code = Convert.ToInt32(codeListBox.Items(2))
```

assigns the third item in the codeListBox to an Integer variable; you also can use the statement Integer.TryParse(codeListBox.Items(2).ToString, code)

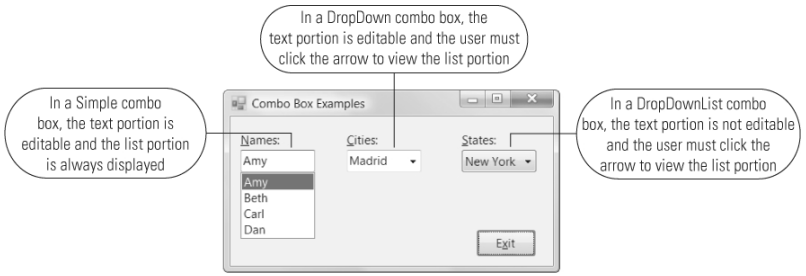
ComboBox Control

- The **ComboBox** control appears in two parts:
  - The top part is a text box that allows the user to type a list item.
  - The second part is a list box that displays a list of items from which the user can select one.
- Essentially combo box control contain the same properties, events, and methods of a list box control



Style for ComboBox

- Three styles of combo boxes:
  - Simple
  - DropDown (the default)
  - DropDownList

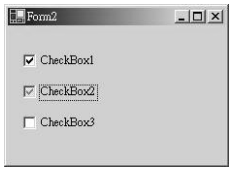


Common Properties for ComboBox Control

Properties of the ComboBox Class	Description
	Inherits properties, methods, and events from the Control class
Items	Identifies the items contained within the combo box
MaxDropDownItems	Indicates the maximum number of items that will be shown in the drop-down portion of the combo box
SelectedIndex	Indicates the index of the currently selected item in the combo box
SelectedItem	Identifies the currently selected item in the combo box
Sorted	Indicates whether the items in the combo box are sorted
Text	Identifies the text that is displayed by default in the combo box
Common Event	
SelectedIndexChanged	Occurs when the item selected in the combo box changes

CheckBox Control

- **CheckBox** Control allows the user to select or deselect one or more items in any group, which means you can select more than one check box at a time.
- You can change the content of ***text*** property of the check box to what you want it to appear in the check box on the form. You can also change each individual object so each has different properties from others of the same type.
- When a check box is checked, the ***checked*** property becomes true.



Radio Button Control

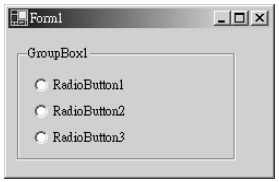
- **Radio Button** Control allows the user to select or deselect only one item in any group, which means you can only select one radio button at a time.
- You can change the content of ***text*** property of the radio button to what you want it to appear in the radio button on the form. You can also change each individual object, so each has different properties from others of the same type.
- When a radio button is checked, the ***checked*** property becomes true.

Using Radio Buttons

- Minimum number of radio buttons in a group is two
  - Must select a radio button to deselect another
- Recommended maximum number in a group: seven
- Windows standard is to set one as the default radio button
  - Shows as selected when the screen appears
  - Should be the most likely selection or the first radio button in the group
- Set the `Checked` property to True to make it the default radio button

Group Box Control

- **Group box** is used as containers for other controls. Group Boxes are used to hold other Objects. Objects grouped together inside a Group Box can be referred to at the same time. Usually, groups of radio buttons or check boxes are placed in group boxes.
  - It used as containers for other controls such as radio buttons and check boxes
  - It improves readability of form



Tick the Check Box / Radio Button

- Set the Checked Property of Check Boxes and Radio Buttons
  - Selects/Deselects Check Box or Radio Button at design or run time
  - Set Checked Property
    - True = Checked, selected;
      - RadioButton.Checked = True
    - False = Unchecked, deselected
      - CheckBox.Checked = False

Control Handling

Coding for Controls

With …End With Statement

- You can use the **With**…**End With** statement to change several properties at once in Code which will run more efficiently.

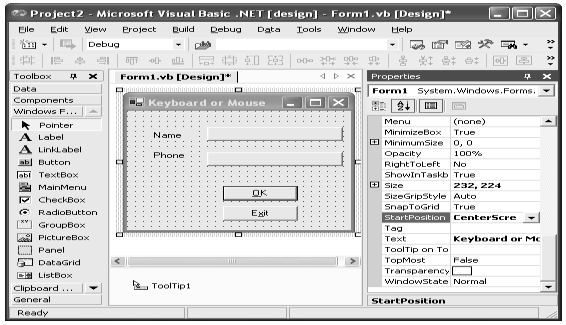
```
With ControlName
        .Property1 = Value1
        .Property2 = Value2
        .Property3 = Value3
End With
```

=

```
ControlName.Property1 = Value1
ControlName.Property2 = Value2
ControlName.Property3 = Value3
```

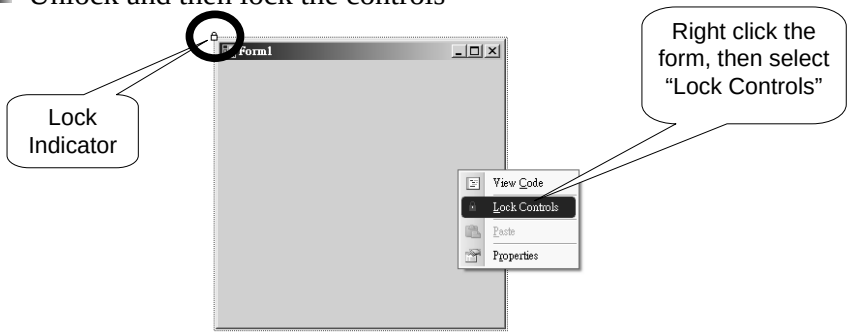
Startup Form Location

- When project runs, the form's screen position can by set by StartPosition property of the form.
- There are several choices of the starting position for the StartPosition Property
  - WindowsDefaultLocation
  - WindowsDefaultBounds
  - Manual
  - CenterScreen
  - CenterParent



Lock Them Down

- Locking the controls
  - Prevents them from being moved inadvertently as you work in the IDE
- Unlock and then lock the controls



Visibility

- The visibility of a control depends on the selection a user makes in controls.
- The visibility of a control can be set at run time.
- When the ***Visible*** property is set to true, the control will be display and visible to user.

Now Property

- Returns a Date value containing the current date and time according to your system.
- Remarks
  - To set the system date, use the ***Today*** property.
  - To set the system time, use the ***TimeOfDay*** property.
  - To return the current system date and time, use the ***Now*** property to return the current system date and time.

Color Selection

- Change the Color of Objects
  - Use VB Color Constants from the Color Class
  - View complete list in Help
    - ForeColor (color of text)
      - TextBox.ForeColor = Color.Red
    - BackColor (color of background)
      - TextBox.BackColor = Color.White
- Available Colors
  - Help: look up the Color Property
  - Click on SystemDrawing Properties, then click on Color Members

Key Press

Handling Key Press Event

Handling the Key Press Event

- Can prevent a text box from accepting an inappropriate character by coding the text box's KeyPress event
- **KeyPress event**: occurs each time the user presses a key while the control has the focus
  - Use the e parameter's KeyChar property to determine the pressed key
  - Use the e parameter's Handled property to cancel the key if it is inappropriate
- Use ControlChars.Back constant to represent the Backspace key on the keyboard

Coding the KeyPress Event

USE THE KEYPRESS EVENT

Example 1

```
Private Sub ageTextBox_KeyPress(ByVal sender As Object, _
    ByVal e As System.Windows.Forms.KeyPressEventArgs) _
    Handles ageTextBox.KeyPress
    ' allows the text box to accept only numbers
    ' and the Backspace key for editing

    If (e.KeyChar < "0" OrElse e.KeyChar > "9") _
        AndAlso e.KeyChar <> ControlChars.Back Then
            e.Handled = True
    End If
End Sub
```

Example 2

```
Private Sub registeredTextBox_KeyPress(ByVal sender As Object, _
    ByVal e As System.Windows.Forms.KeyPressEventArgs) _
    Handles registeredTextBox.KeyPress
    ' allows the text box to accept only the letters
    ' Y, y, N, or n and the Backspace key for editing

    If e.KeyChar <> "Y" AndAlso e.KeyChar <> "y" _
        AndAlso e.KeyChar <> "N" AndAlso e.KeyChar <> "n" _
        AndAlso e.KeyChar <> ControlChars.Back Then
            e.Handled = True
    End If
End Sub
```