

# Visual Basic.NET Windows Control



# ROAD MAP

- Introduction
- Textbox Control
- ListBox, CheckedListBox and ComboBox Controls
- ScrollBar and TrackBar Controls



# INTRODUCTION

- Windows basic controls
  - Its properties to build a user interface



# Textbox Control

- Primary mechanism for displaying and entering text
- A small text editor that provides all the basic text-editing facilities
  - inserting and selecting text
  - scrolling if the text doesn't fit in the control's area
  - exchanging text with other applications through the Clipboard

# Textbox Control

- Versatile data-entry tool
  - entering a single line of text, such as a number or a password, or for entering simple text files

Name	<input type="text" value="Administrator"/>
Password	<input type="password" value="*****"/>


The TextBox control is one of the most common and versatile elements of the Windows user interface.

It can hold a few characters, or any number of lines, as indicated by its scrollbars.

Last Name	<input type="text" value="Evans"/>
First Name	<input type="text" value="Peter"/>
Address	<input type="text" value="1515 Palm Avenue Suite 201"/>
Tel #	<input type="text" value="(805) 555.1234"/>
Fax #	<input type="text" value="(805) 555.1213"/>
Notes	<input type="text" value="This TextBox is used for entering free text."/>



# Basic Properties

- Multilines
  - Scrollbars
  - WordWrap
  - AcceptsReturn, AcceptsTab
  - Maxlength
- 



# MultiLines

- Determine whether the TextBox control will hold a single line or multiple lines of text
- By default, the control holds a single line of text
- Set True





# Scrollbars

- Controls the attachment of scroll bars to the TextBox control if the text exceeds the control's dimensions
- Single-line text boxes
  - can't have a scroll bar attached, even if the text exceeds the width of the control
- Multiline text boxes
  - can have a horizontal or a vertical scroll bar, or both.

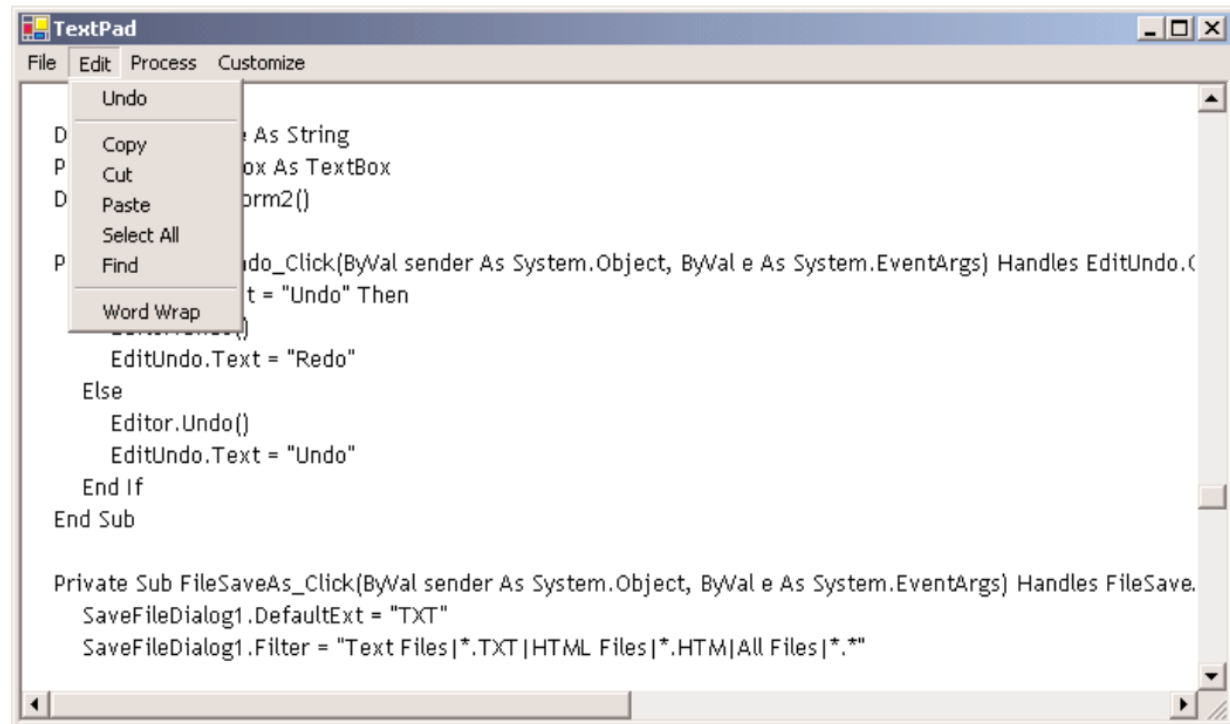


# WordWrap

- Determines whether the text is wrapped automatically when it reaches the right edge of the control
- The default value of this property is True

**FIGURE 6.2**

Turn off the WordWrap property to display program listings or other lines that shouldn't break arbitrarily.



# AcceptsReturn, AcceptsTab

- Specify how the TextBox control reacts to the Return (Enter) and Tab keys
- The Enter key activates the default button on the form, if there is one
- Activated with Enter key

**TIP** *This is a very important issue in designing practical user interfaces. You shouldn't force your users to switch between the keyboard and the mouse all the time. Follow the Windows standards (the Enter key for the default button, the Tab key to move from one control to the next, and shortcuts) to make sure that your application can be used without the mouse. Data-entry operators would rather work without the mouse at all.*

- The AcceptsTab property determines how the control reacts to the Tab key



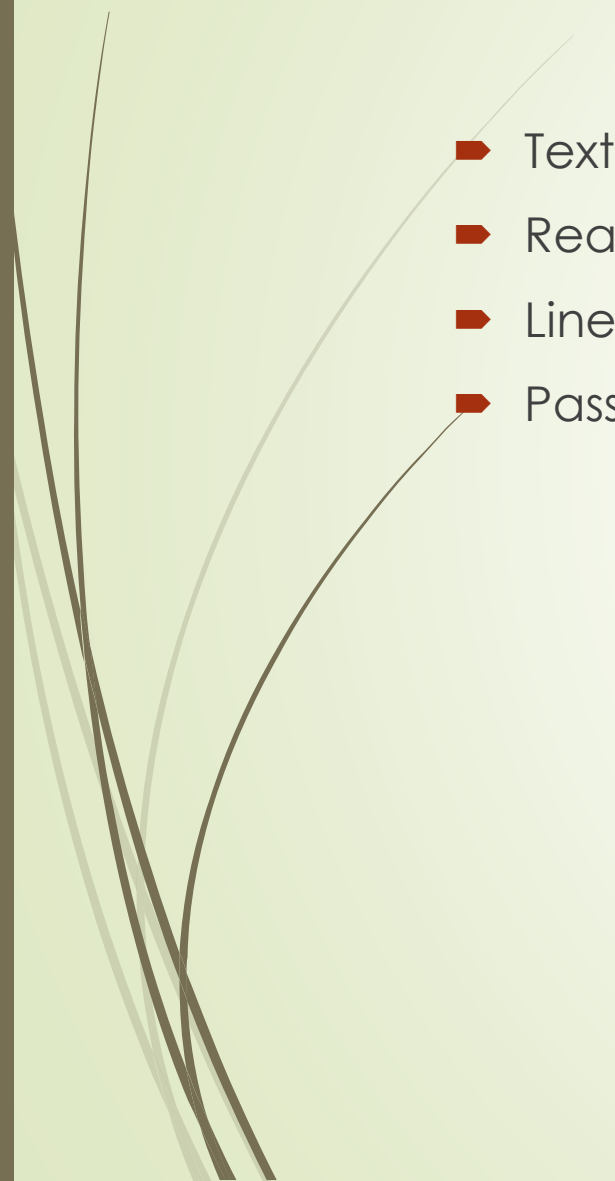
# Maxlength

- Determine the number of characters the TextBox control will accept
- Default: 32767

**NOTE** *The MaxLength property of the TextBox control is often set to a specific value in data-entry applications. This prevents users from entering more characters than can be stored in a database field.*



# Text-Manipulation Properties

- Text
  - Read Only, Locked
  - Lines
  - PasswordChar
- 



# Text

- Hold the control's text
- Available at design time so that you can assign some initial text to the control
- Single-line TextBox
  - controls, set the Text property to a short string, as usual
- Multiline TextBox controls
  - open the Lines property and enter the text on the String Collection Editor window, which will appear



# ReadOnly, Locked

- Display text on a TextBox control but prevent users from editing it
  - Set True
  - put text on the control from within your code, and users can view it, yet they can't edit it

# Lines


- Access the text on the control
- read-only
- Lines is a string array where each element holds a line of text

```
Dim iLine As Integer
For iLine = 0 To TextBox1.Lines.GetUpperBound(0)- 1
    { process string TextBox1.Lines(iLine) }
Next
```






# PasswordChar

- Available at design time, this property turns the characters typed into any character you specify
  - Default value: is an empty string,
- 



# Text Selection Properties

- Selected Text
  - SelectionStart, SelectionLength
  - HideSelection
- 

# Selected Text

- Returns the selected text, enabling you to manipulate the current selection from within your code

```
TextBox1.SelectedText = TextBox1.SelectedText.ToUpper
```

or use the UCase() function of VB6:

```
TextBox1.SelectedText = UCase(TextBox1.SelectedText)
```

To delete the current selection, assign an empty string to the SelectedText property:

```
TextBox1.SelectedText = ""
```

# SelectionStart, SelectionLength

- SelectionStart

- Return or sets the position of the first character of the selected text in the control's text

- SelectionLength

- Return or sets the length of the selected text.

**TIP** The *SelectionStart* and *SelectionLength* properties always have a value even if no text has been selected. In this case, *SelectionLength* is 1, and *SelectionStart* is the current location of the pointer in the text. If you want to insert some text at the pointer's location, simply assign it to the *SelectedText* property.

# SelectionStart, SelectionLength

➤ Example:

```
Dim seekString As String
Dim textStart As Integer
seekString = "Visual"
textStart = TextBox1.Text.IndexOf(seekString)
If textStart > 0 Then
    TextBox1.SelectionStart = textStart
    TextBox1.SelectionLength = seekString.Length
End If
TextBox1.ScrollToCaret()
```



# HideSelection

- The selected text on the TextBox will not remain highlighted when the user moves to another control or form
- To change this...
  - Using HideSelection
- Default: True
  - the text doesn't remain high lighted when the text box loses the focus.

# Text-Selection Methods

- Select Method

```
TextBox1.Select(start, length)
```

- Equivalent to setting the SelectionStart and SelectionLength properties

```
TextBox1.Select(99, 6)
```





# Undoing Edits

- Edit:
  - insertion or deletion of characters.
- TextBox control can automatically undo the most recent edit operation
  - If CanUndo property is True, call Undo method

# ListBox, CheckedListBox and ComboBox Controls

## ■ ListBox Controls

- Occupy a user-specified amount of space on the form and is populated with a list of items
- If the list of items is longer than can fit on the control, a vertical scroll bar appears automatically
- The items must be inserted in the ListBox control through the code or via the Properties window

## ■ ComboBox Control


- an expandable ListBox control
- the user can expand it to make a selection and collapse it after the selection is made
- can enter new information in the ComboBo

## ■ CheckedListBoc Control

- a check box appears in front of each item
- The user can select any number of items by checking the boxes in front of them



# Properties (ListBox & ComboBox)

- 
- IntegralHeight
  - Items
  - MultiColumn
  - SelectionMode
  - Sorted
  - Text

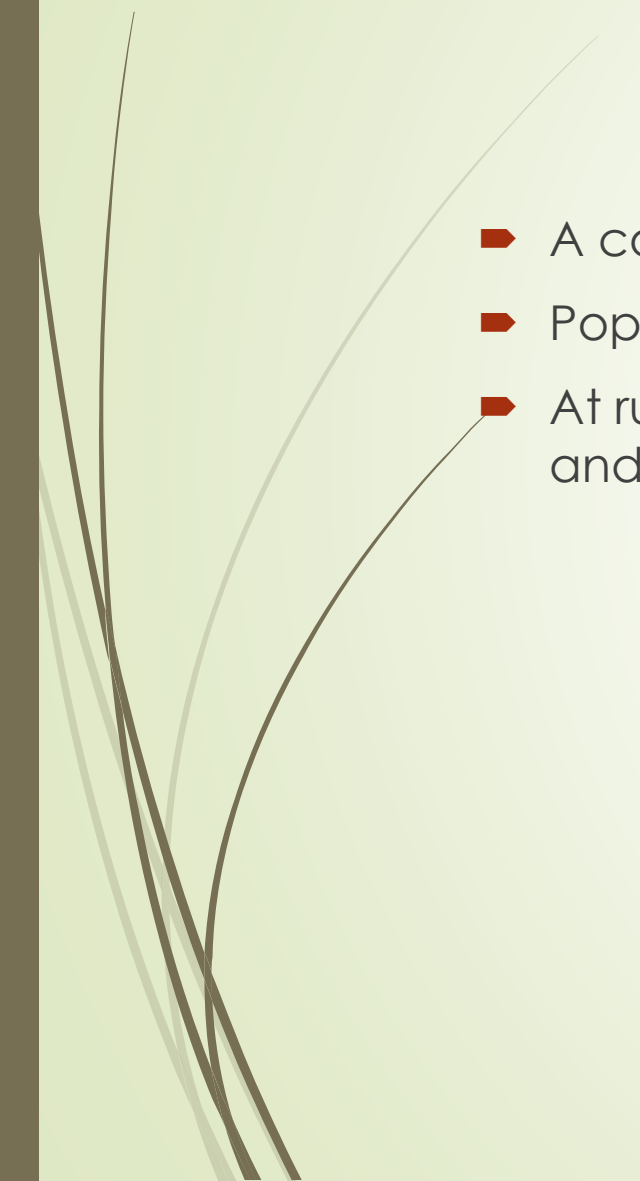


# IntegralHeight

- a Boolean value (True/False)
- Whether the control's height will be adjusted to avoid the partial display of the last item



# Items

- A collection that holds the items on the control
  - Populate this list through the String Collection Editor window
  - At runtime you can access and manipulate the items through the methods and properties of the Items collection (later)
- 



# Multicolumns

- Display its items in multiple columns
- Set its MultiColumn property to True
- Problem with multicolumn ListBoxes:
  - you can't specify the column in which each item will appear
  - Set this property to True for ListBox controls with a relatively small number of Items
  - do so only when you want to save space on the form

# SelectionMode

- Determines how the user can select the list's items and must be set at design time
- whether the user can select multiple items and which method will be used for multiple selections

**TABLE 6.2: THE SELECTIONMODE ENUMERATION**

VALUE	DESCRIPTION
None	No selection at all is allowed.
One	(Default) Only a single item can be selected.
MultiSimple	Simple multiple selection: A mouse click (or pressing the spacebar) selects or deselects an item in the list. You must click all the items you want to select.
MultiExtended	Extended multiple selection: Press Shift and click the mouse (or press one of the arrow keys) to expand the selection. This will highlight all the items between the previously selected item and the current selection. Press Ctrl and click the mouse to select or deselect single items in the list.





# Sorted

- Insert the proper place and maintaining some sort of organization
- Set the control's Sorted property to True
- The ListBox control
  - a text control
  - Don't sort numeric data properly
  - Format them with leading zeros ( "010" "005" for number 10 and 5 )
  - Sorted in ascending and case-sensitive order
  - Cannot Change the default setting

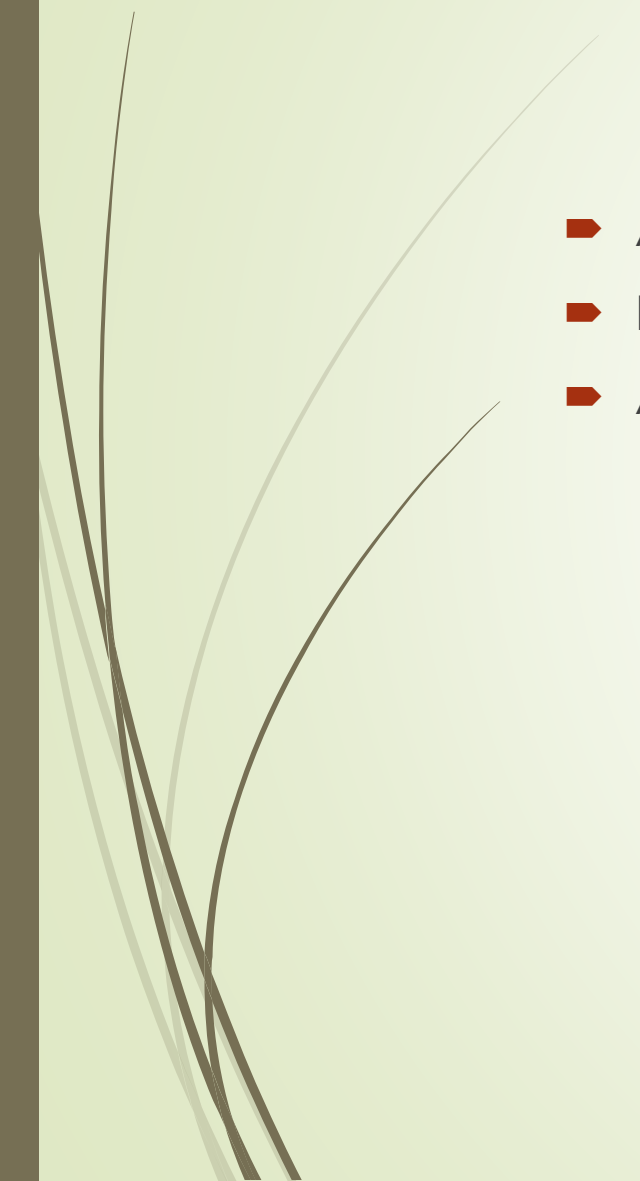


# Texted

- Return the selected text on the control
- 



# The Items Collection

- Add items to the list
  - Remove items from the list
  - Access individual items in the list
- 



# Add

- ▶ Items.Add or Items.Insert
- ▶ You can add any object to the ListBox control, but items are usually strings
- ▶ The Add method appends new items to the end of the list, unless the Sorted property has been set to True.

```
ListBox1.Items.Add(item)
```



# Clear

- Removes all the items from the control

```
List1.Items.Clear
```



# Count

- The number of items in the list
- For...Next loop
  - loop's counter must go from 0 to `ListBox1.Items.Count - 1`



# CopyTo

- ▀ retrieves all the items from a ListBox control and stores them to the array passed to the method as argument
- ▀ (name, index)

```
ListBox1.CopyTo(destination, index)
```





# Insert

- Insert an item at a specific location

```
ListBox1.Items.Insert(index, item)
```



# Remove

- Remove an item from the list
- First find its position

```
ListBox1.Items.Remove(index)
```

```
ListBox1.Remove(0)
```

```
ListBox1.Items.Remove(item)
```



# Contains

- Accepts an object as argument and returns a True/False value indicating whether the collection contains this object or not

```
Dim itm As String = "Remote Computing"  
If Not ListBox1.Items.Contains(itm) Then  
    ListBox1.Items.Add(itm)  
End If
```



# Selecting Items

- The ListBox control allows the user to select either one or multiple items
- SelectionMode Property
- SelectedItems property
- SelectedItems.Count → reports the number of selected items
- Example: iteration through all selected items

```
Dim itm As Object
For Each itm In ListBox1.SelectedItems
    Console.WriteLine(itm)
Next
```

# AddItem Button

- use the InputBox() function

```
Private Sub btnAdd1_Click(ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) Handles btnAdd1.Click  
    Dim ListItem As String  
    ListItem = InputBox("Enter new item's name")  
    If ListItem.Trim <> "" Then  
        sourceList.Items.Add(ListItem)  
    End If  
End Sub
```

# Remove Selected Item(s) Button

- The Remove Selected Items button
  - scan all the items of the left list and remove the selected one(s)
  - To delete an item, you must have at least one item selected
- Not safe to remove items by name

## LISTING 6.11: THE REMOVE BUTTONS

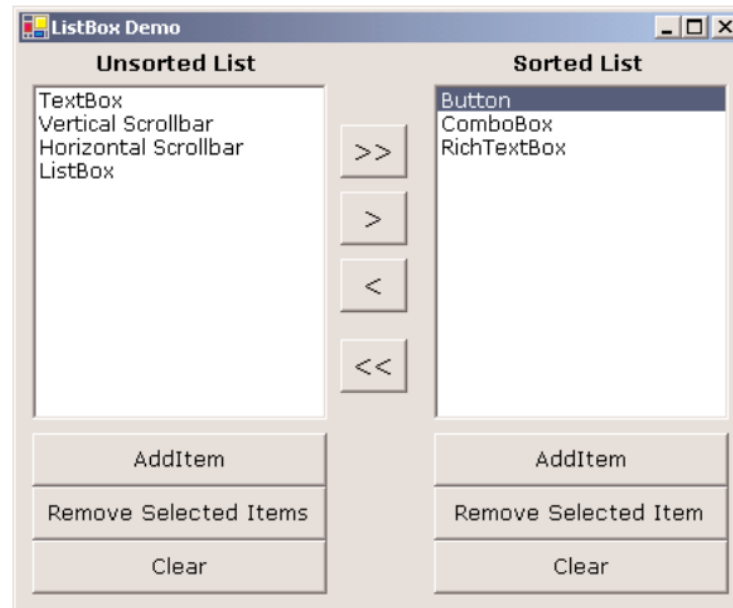
```
Private Sub btnRemoveSelDest_Click(ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) Handles btnRemoveSelDest.Click  
    destinationList.Items.Remove(destinationList.SelectedItem)  
End Sub  
Private Sub btnRemoveSelSrc_Click(ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) Handles btnRemoveSelSrc.Click  
    Dim i As Integer  
    For i = 0 To sourceList.SelectedIndices.Count - 1  
        sourceList.Items.RemoveAt(sourceList.SelectedIndices(0))  
    Next  
End Sub
```

# The Arrow Button

- The two Buttons with the single arrows transfer selected items from one list to another

**FIGURE 6.7**

ListDemo demonstrates most of the operations you'll perform with ListBoxes.





# The Arrow Button

- First Arrow Button
  - transfer a single element only after it ensures that the list contains a selected item
    - Add the item to the second list
    - Remove the item from the original list
- Second Arrow Button
  - transfer items in the opposite direction

## LISTING 6.12: THE RIGHT ARROW BUTTON

```
Private Sub btnMoveDest_Click(ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) Handles btnMoveDest.Click  
    Dim i As Integer  
    While sourceList.SelectedIndices.Count > 0  
        destinationList.Items.Add(sourceList.SelectedItems(i))  
        sourceList.Items.Remove(sourceList.SelectedItems(i))  
    End While  
End Sub
```



# The ComboBox Control

- Recall:
- Similar to the ListBox control in the sense that it contains multiple items of which the user may select one
- BUT it typically occupies less space on-screen
- An expandable ListBox control
- Normally, it displays one line with the selected item
- Difference (vs ListBox)
  - ComboBox allows the user to specify items that don't exist in the list
  - the Text property of the ComboBox is read-only at runtime

# The ComboBox Control

## ➤ Three Types of ComboBox Control

**TABLE 6.3: STYLES OF THE COMBOBOX CONTROL**

VALUE	EFFECT
DropDown	(Default) The control is made up of a drop-down list and a text box. The user can select an item from the list or type a new one in the text box.
DropDownList	This style is a drop-down list, from which the user can select one of its items but can't enter a new one.
Simple	The control includes a text box and a list that doesn't drop down. The user can select from the list or type in the text box.



# The ScrollBar and TrackBar Control

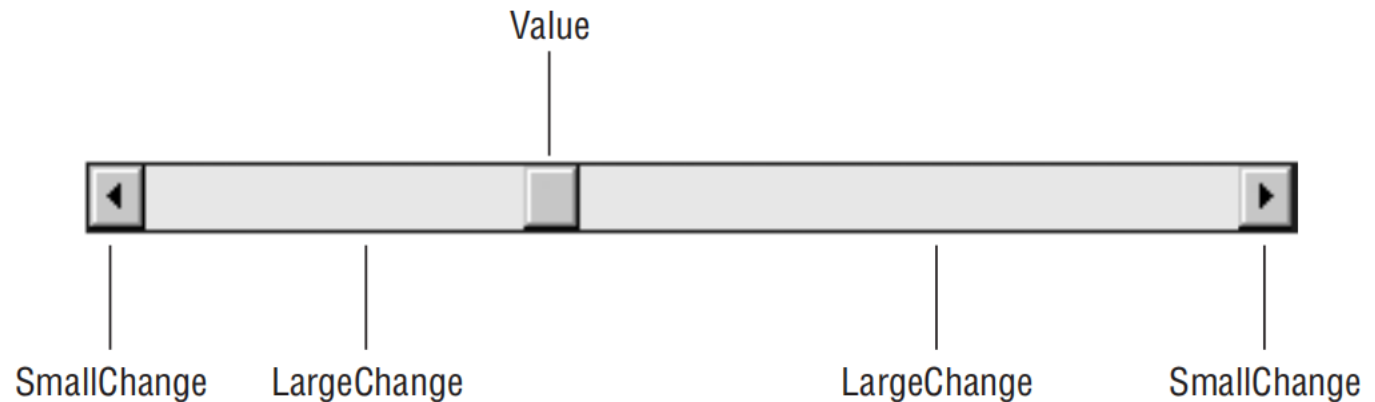
- Let the user specify a magnitude by scrolling a selector between its **minimum** and **maximum** values
- ScrollBar
  - Move up and down to navigate through document
  - should be used when the exact value isn't as important as the value's effect on another object or data element
- TrackBar
  - Do not cover a continuous range of values
  - should be used when the user can type a numeric value and the value your application expects is a number in a specific range

# The ScrollBar Control

- A long stripe with an indicator that lets the user select a value between the two ends of the control
- can be positioned either vertically or horizontally
  - Use the Orientation property
- The left (or bottom) end of the control → minimum value
  - the other end is the control's maximum value

**FIGURE 6.13**

The basic properties of the ScrollBar control





# The ScrollBar Control

- Minimum

- The control's minimum value
- Default value = 0
- Can be set to negatives since it's integer

- Maximum

- The control's maximum value
- Default value = 100
- Can be set to any value

- Value

- The control's current value
- Specified by Indicator's position



# The ScrollBar Control's Events

- **By clicking the two arrows at its ends**
  - The value of the control changes by the amount specified with the SmallChange property
- **By clicking the area between the indicator and the arrows**
  - The value of the control changes by the amount specified with the LargeChange property
- **By dragging the indicator with the mouse.**



# The ScrollBar Control's Events

**TABLE 6.4:** THE ACTIONS THAT CAN CAUSE THE SCROLL EVENT

MEMBER	DESCRIPTION
EndScroll	The user has stopped scrolling the control.
First	The control was scrolled to the Minimum position.
LargeDecrement	The control was scrolled by a large decrement (user clicked the bar between the button and the left arrow).
LargeIncrement	The control was scrolled by a large increment (user clicked the bar between the button and the right arrow).
Last	The control was scrolled to the Maximum position.
SmallDecrement	The control was scrolled by a small decrement (user clicked the left arrow).
SmallIncrement	The control was scrolled by a small increment (user clicked the right arrow).
ThumbPosition	The button was moved.
ThumbTrack	The button is being moved.

# The TrackBar Control

- Similar to ScrollBar Control
- BUT lacks the granularity of ScrollBar
- SmallChange and LargeChange properties are available

**NOTE** Granularity is how specific you want to be in measuring. In measuring distances between towns, a granularity of a mile is often adequate. In measuring (or specifying) the dimensions of a building, the granularity could be on the order of a foot or an inch. The TrackBar control lets you set the type of granularity that's necessary for your application.

# Exercise

- (1) If the TextBox control contains the string "ABCDEFGHI," then the following statement will select the range "DEFG":

- (2) Replaced following with a single call to the Select method:

```
TextBox1.SelectionStart = selStart - 1  
TextBox1.SelectionLength = word.Length
```

- (3) What does MultiExtended mean in the SelectionMode Enumeration?
- (4) How to sort the following items?

"BA" "aA" "AA" "aa" "ba" "Aa"



# Exercise

- (5) Insert “new Item” at index 0 in ListBox1
- (6) What's the meaning of the LargeDecrement in ScrollBar Control's Events?

# Answers

- (1) `TextBox1.Select(3, 4)`
- (2) `TextBox1.Select(selStart - 1, word.Length)`
- (3) Extended multiple selection: Press Shift and click the mouse (or press one of the arrow keys) to expand the selection. This will highlight all the items between the previously selected item and the current selection. Press Ctrl and click the mouse to select or deselect single items in the list.
- (4) "AA" "Aa" "aA" "aa" "BA" "ba"



# Answers



➤ (5)

```
ListBox1.Items.Insert(0, "new item")
```

➤ (6)

The control was scrolled by a large decrement (user clicked the bar between the button and the left arrow).