Visual Basic.NET Using Procedures

ROAD MAP

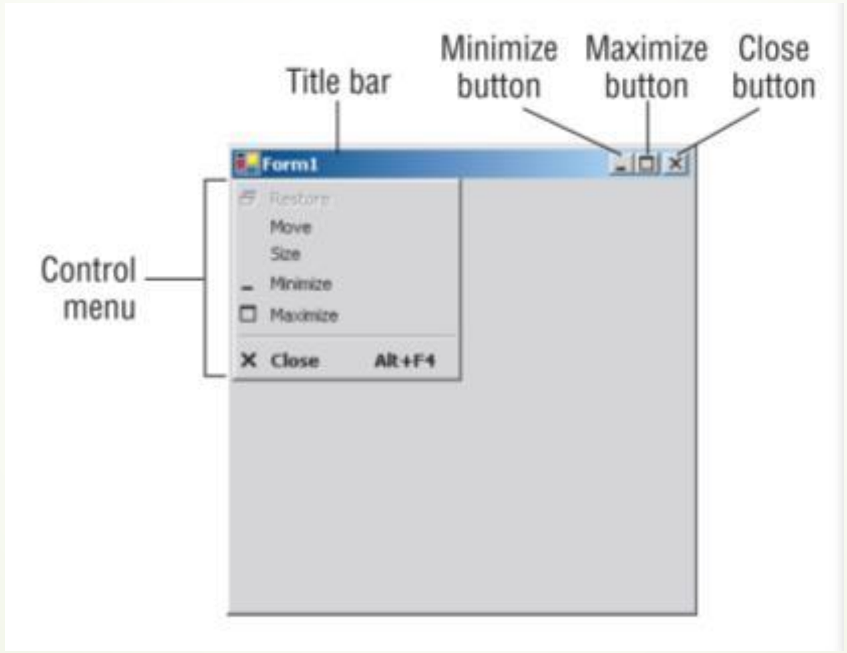
- Appearance of Forms
- Properties of Form Control
- Loading and Showing Forms
- Designing Menus
- Manipulating Menus at Runtime
- Building Dynamic Forms

Appearance of Forms

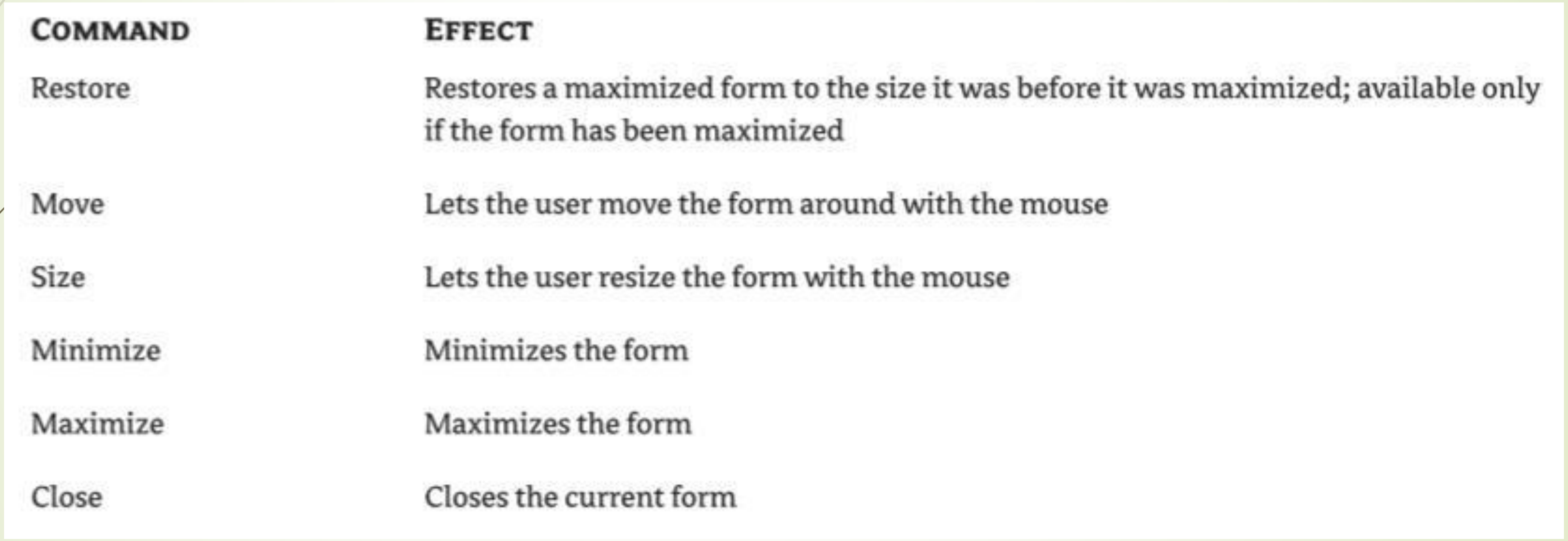
- Application is made up of
  - one or more forms
  - usually more than one
- Main characteristic:
  - title bar
  - form’s caption

The Elements of the Form

Example:



Commands of the Control Menu



COMMAND	EFFECT
Restore	Restores a maximized form to the size it was before it was maximized; available only if the form has been maximized
Move	Lets the user move the form around with the mouse
Size	Lets the user resize the form with the mouse
Minimize	Minimizes the form
Maximize	Maximizes the form
Close	Closes the current form

Properties of the Form Control

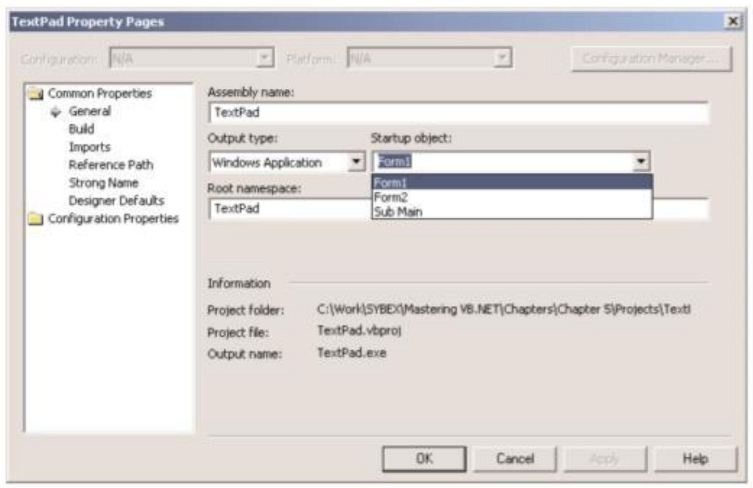
- Accept Button, Cancel Button
- Auto Scale
- Auto Scroll
- Border Style
- Control Box
- Minimize…, Maximize…
- Size Grip Style
- Start Position
- Top, Left
- Top Most
- Width, Height

Loading and Showing Forms

- The Startup Form
  - A typical application has more than a single form
- Controlling One Form from within Another
  - Sharing Variables between Forms
- Forms vs. Dialog Boxes
- The Multiple Forms Project

The Startup Form

- Specify the form that's displayed when the application in the properties window



Assembly name: TextPad
Output type: Windows Application
Startup object: Form1
Root namespace: TextPad

Information
Project folder: C:\Work\SYBEX\Mastering VB.NET\Chapters\Chapter 5\Projects\TextI
Project file: TextPad.vbproj
Output name: TextPad.exe

- Or start an application with a subroutine without loading a form.
- Then open the Project Properties dialog box and specify that the project's startup object is the subroutine Main().

```
Module StartUpModule
   Sub Main()
      System.Windows.Forms.Application.Run(New AuxiliaryForm())
   End Sub
End Module
```

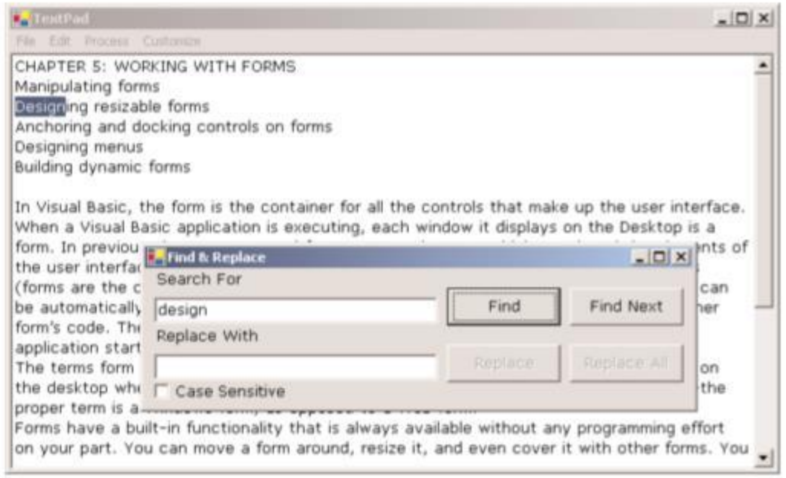
Controlling One Form from within Another

- The Find & Replace form acts on the contents of a control on another form.

- Sharing Variables between Forms
  - via public variables.
  - FRM is a variable that references the form in which the public variables were declared.

```
Public Shared NumPoints As Integer
Public Shared DataValues(100) As Double
```

```
FRM.NumPoints = 99
FRM.DataValues(0) = 0.3395022
```



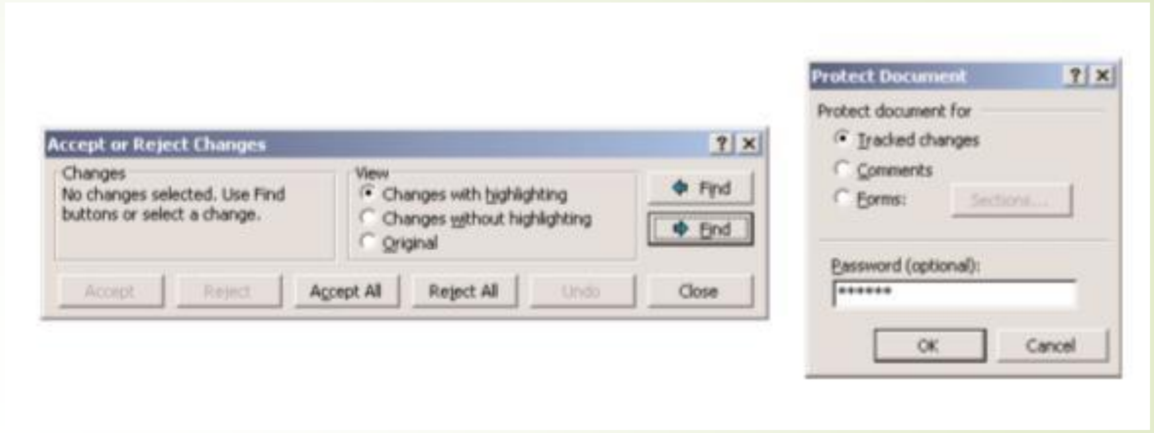
```
CHAPTER 5: WORKING WITH FORMS
Manipulating forms
Designing resizable forms
Anchoring and docking controls on forms
Designing menus
Building dynamic forms

In Visual Basic, the form is the container for all the controls that make up the user interface.
When a Visual Basic application is executing, each window it displays on the Desktop is a
form. In previou                                                                    nts of
the user interfa
(forms are the c                                                                    can
be automatically                                                                    her
form's code. Th
application start
The terms form                                                                      on
the desktop wh                                                                      the
proper term is a
Forms have a built-in functionality that is always available without any programming effort
on your part. You can move a form around, resize it, and even cover it with other forms. You
```

Search For: design

Forms vs. Dialog Boxes

- Dialog boxes
  - special types of forms with rather limited functionality
  - we use it to prompt the user for data
- Typical dialog boxes used by Word
- A simple dialog box that prompts users for a username and password


Accept or Reject Changes

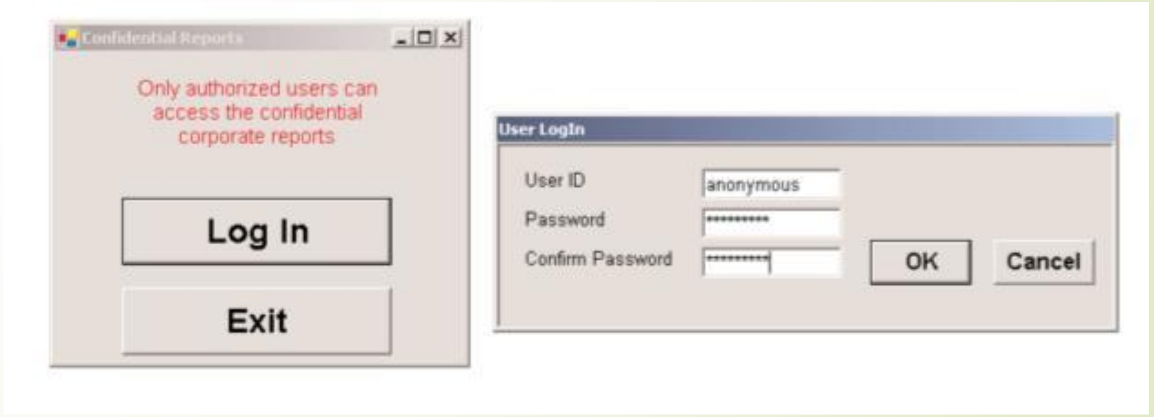
Changes: No changes selected. Use Find buttons or select a change.

View: (●) Changes with highlighting ( ) Changes without highlighting ( ) Original

Protect Document

Protect document for: (●) Tracked changes ( ) Comments ( ) Forms:

Password (optional): ******


Confidential Reports

Only authorized users can access the confidential corporate reports

User LogIn

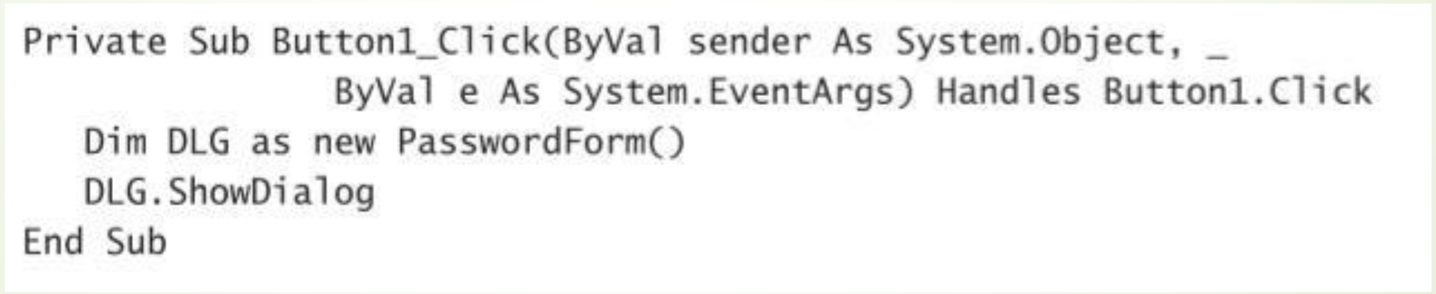
User ID: anonymous

Password: **********

Confirm Password: *********

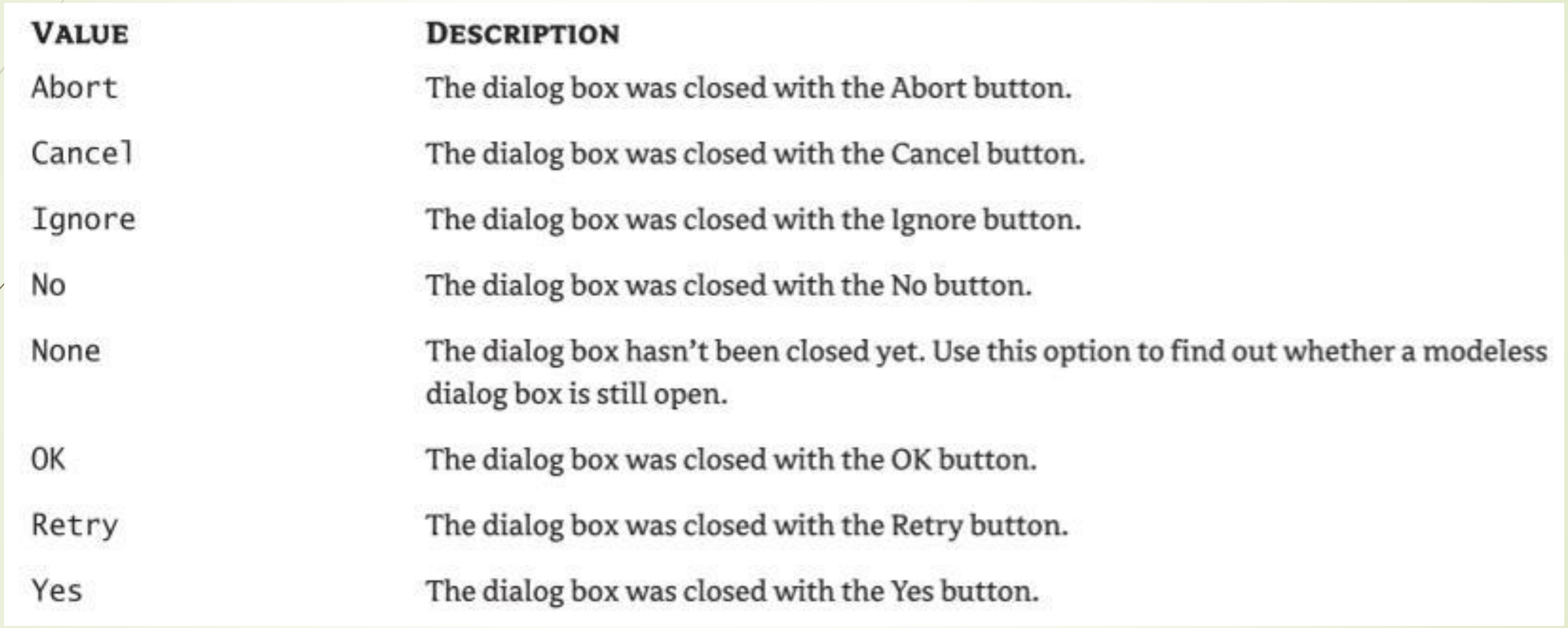
Initiate a dialog box from within another form's code

- The process of displaying a dialog box is no different than displaying another form.
- Enter the following code in the event handler from which you want to initiate the dialog box:



```
Private Sub Button1_Click(ByVal sender As System.Object, _
                ByVal e As System.EventArgs) Handles Button1.Click
   Dim DLG as new PasswordForm()
   DLG.ShowDialog
End Sub
```

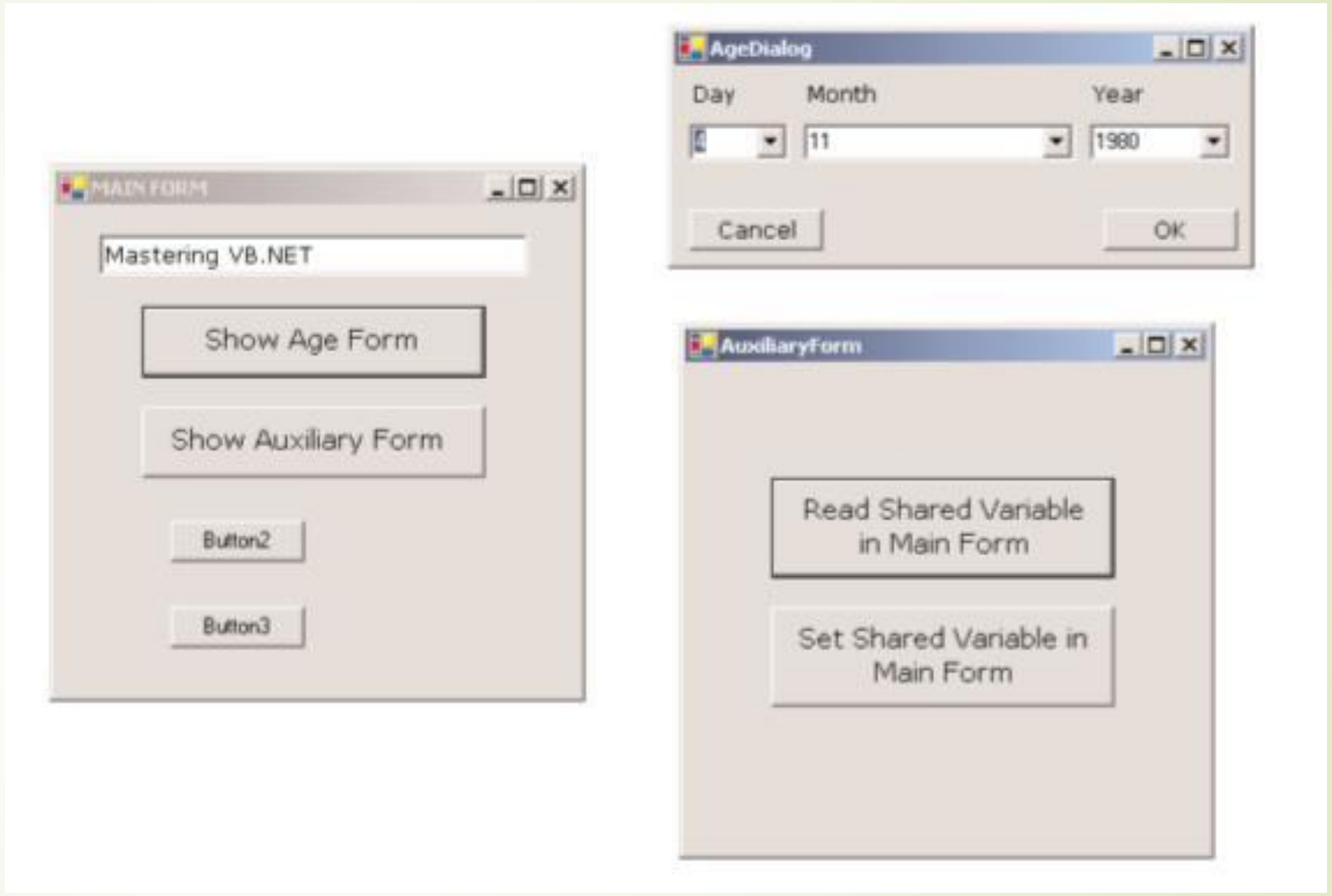
The Dialog Result Enumeration



VALUE	DESCRIPTION
Abort	The dialog box was closed with the Abort button.
Cancel	The dialog box was closed with the Cancel button.
Ignore	The dialog box was closed with the Ignore button.
No	The dialog box was closed with the No button.
None	The dialog box hasn't been closed yet. Use this option to find out whether a modeless dialog box is still open.
OK	The dialog box was closed with the OK button.
Retry	The dialog box was closed with the Retry button.
Yes	The dialog box was closed with the Yes button.

Exercise: The Multiple Forms Project

- A main form
- An auxiliary form
- A dialog box

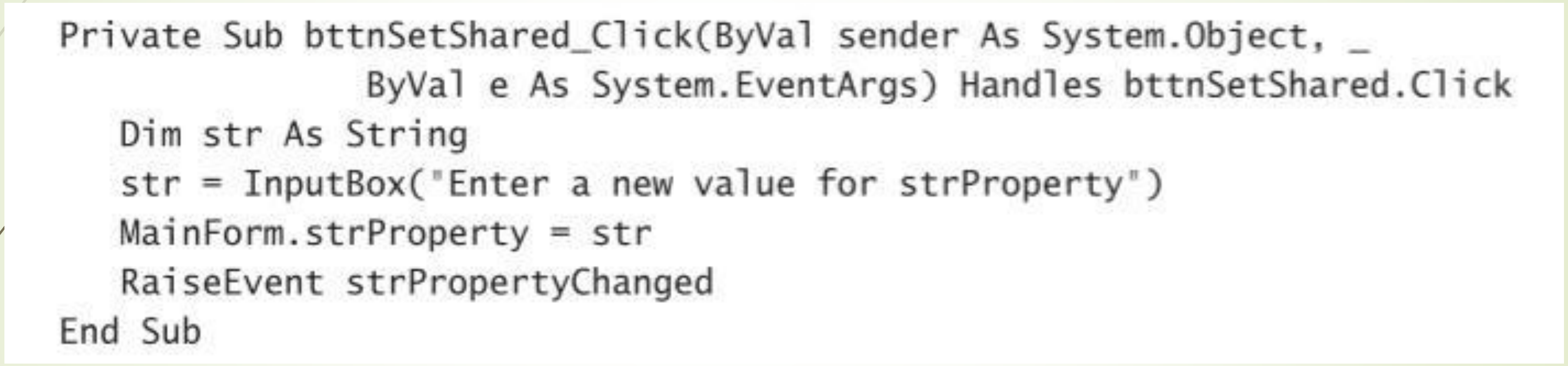


Arguments

- This variable is, in effect, a property of the main form and is declared with the following statements:
  - Public Shared strProperty As String = “Mastering VB.NET”

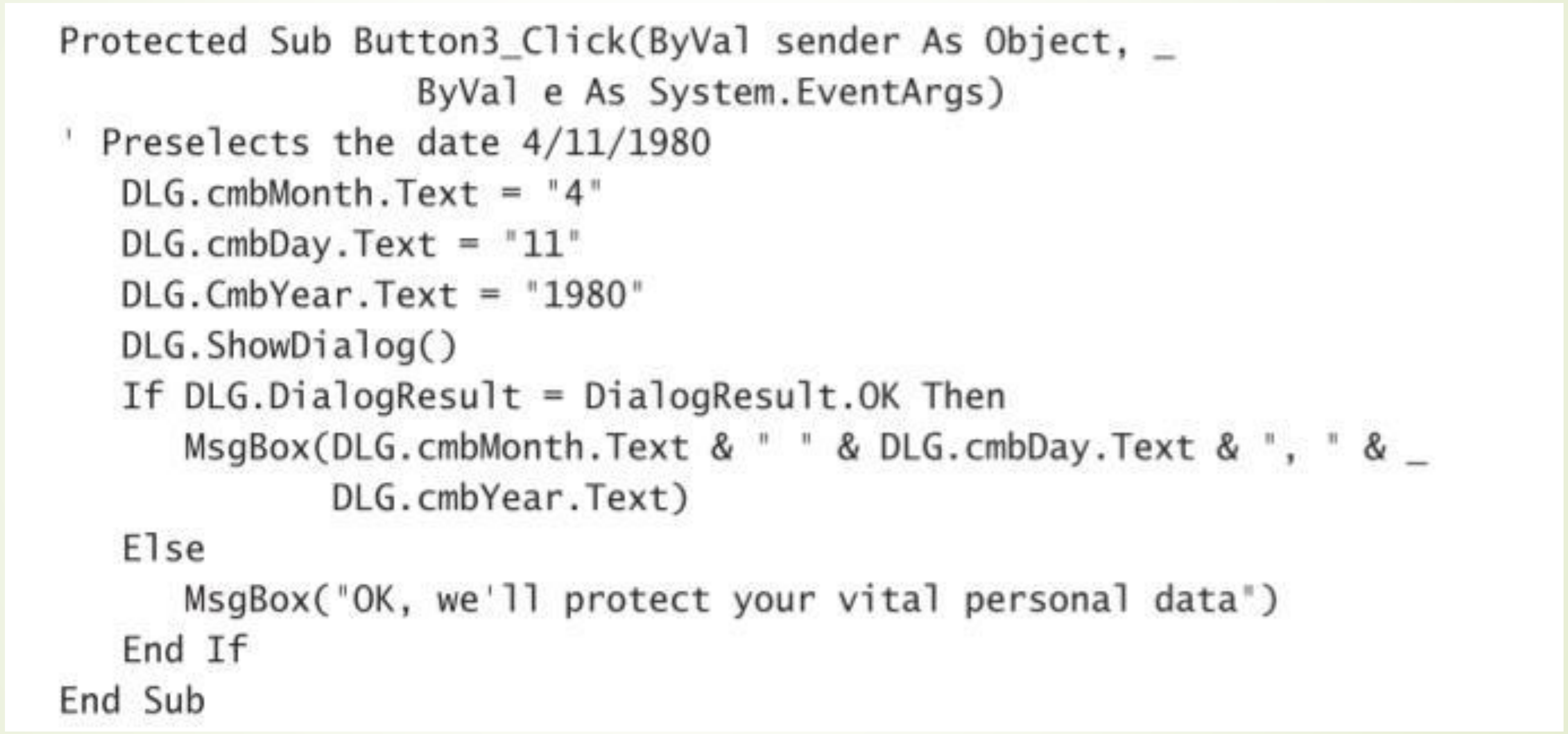
- The declaration must appear in the form’s declarations section:
  - Dim FRM As New AuxiliaryForm()

Raising an Event



```
Private Sub bttnSetShared_Click(ByVal sender As System.Object, _
               ByVal e As System.EventArgs) Handles bttnSetShared.Click
   Dim str As String
   str = InputBox("Enter a new value for strProperty")
   MainForm.strProperty = str
   RaiseEvent strPropertyChanged
End Sub
```

Displaying a Dialog Box and Reading Its Values



```
Protected Sub Button3_Click(ByVal sender As Object, _
                 ByVal e As System.EventArgs)
' Preselects the date 4/11/1980
   DLG.cmbMonth.Text = "4"
   DLG.cmbDay.Text = "11"
   DLG.CmbYear.Text = "1980"
   DLG.ShowDialog()
   If DLG.DialogResult = DialogResult.OK Then
      MsgBox(DLG.cmbMonth.Text & " " & DLG.cmbDay.Text & ", " & _
             DLG.cmbYear.Text)
   Else
      MsgBox("OK, we'll protect your vital personal data")
   End If
End Sub
```

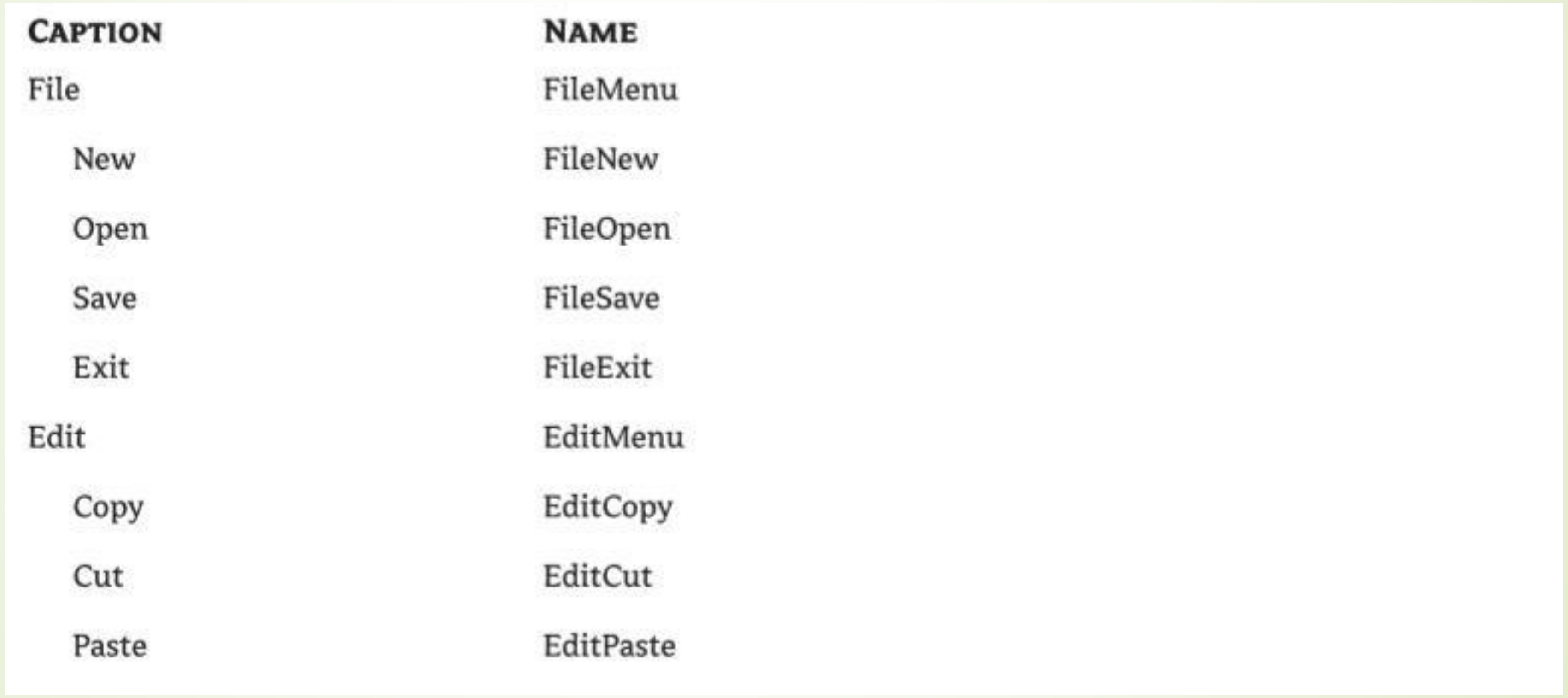
Setting the Dialog Box’s Dialog Result Property

```
Protected Sub bttnOK_Click(ByVal sender As Object, ByVal e As System.EventArgs)
   Me.DialogResult = DialogResult.OK
End Sub
Protected Sub bttnCancel_Click(ByVal sender As Object, _
                               ByVal e As System.EventArgs)
   Me.DialogResult = DialogResult.Cancel
End Sub
```

Designing Menus

- The Menu Editor
- The MenuItem Object’s Properties
- Manipulating Menus at Runtime
- Iterating a Menu’s Items

The Captions and Names of the File and Edit Menus



Caption	Name
File	FileMenu
New	FileNew
Open	FileOpen
Save	FileSave
Exit	FileExit
Edit	EditMenu
Copy	EditCopy
Cut	EditCut
Paste	EditPaste

The MenuItem Object’s Properties

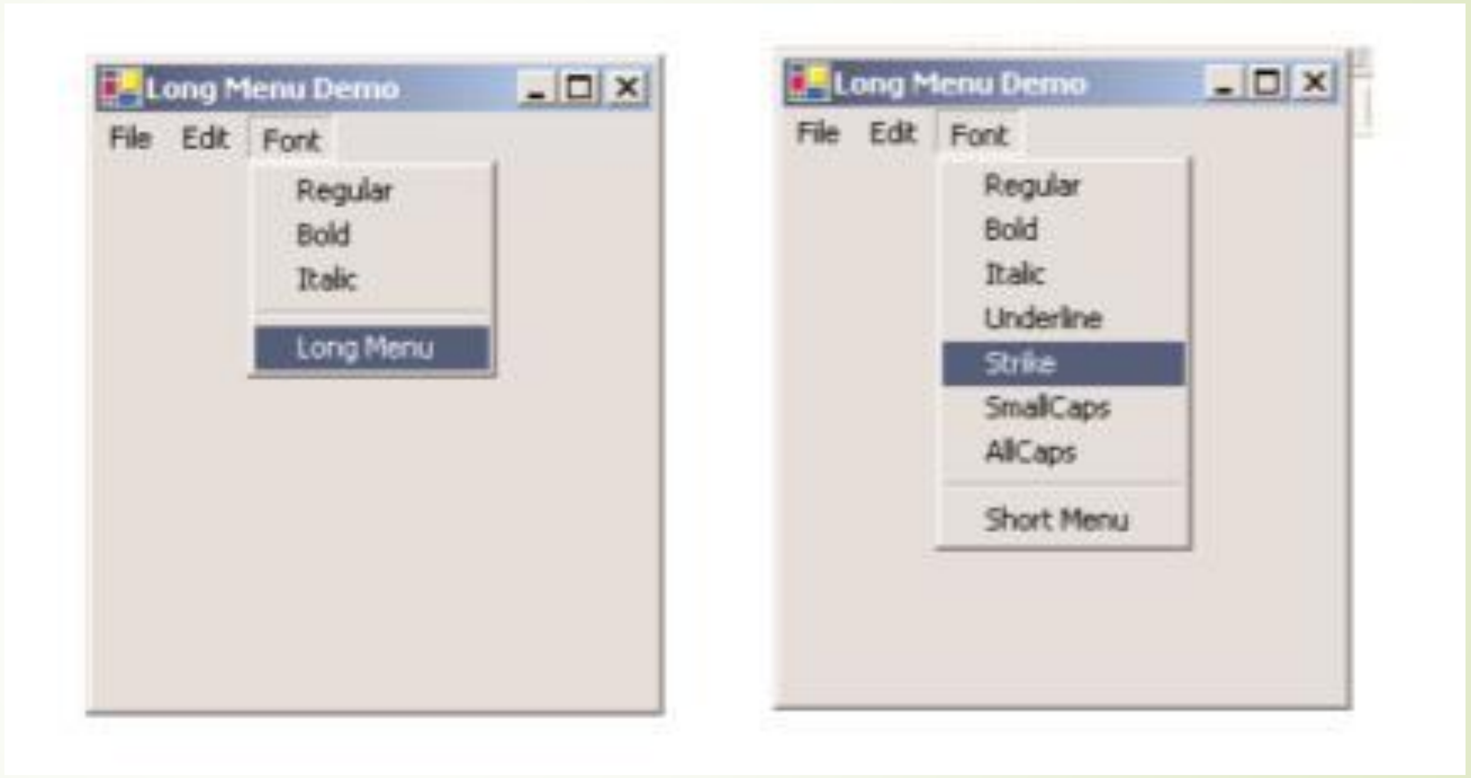
- Checked
- DefaultItem
- Enabled
- IsParent
- Mnemonic
- Visible
- MDIList

Manipulating Menus at Runtime

- Dynamic menus change at runtime to display more or fewer commands
  - Creating short and long versions of the same menu
  - Adding and removing menu commands at runtime

Creating Short and Long Menus

- The two versions of the Font menu of the LongMenu application

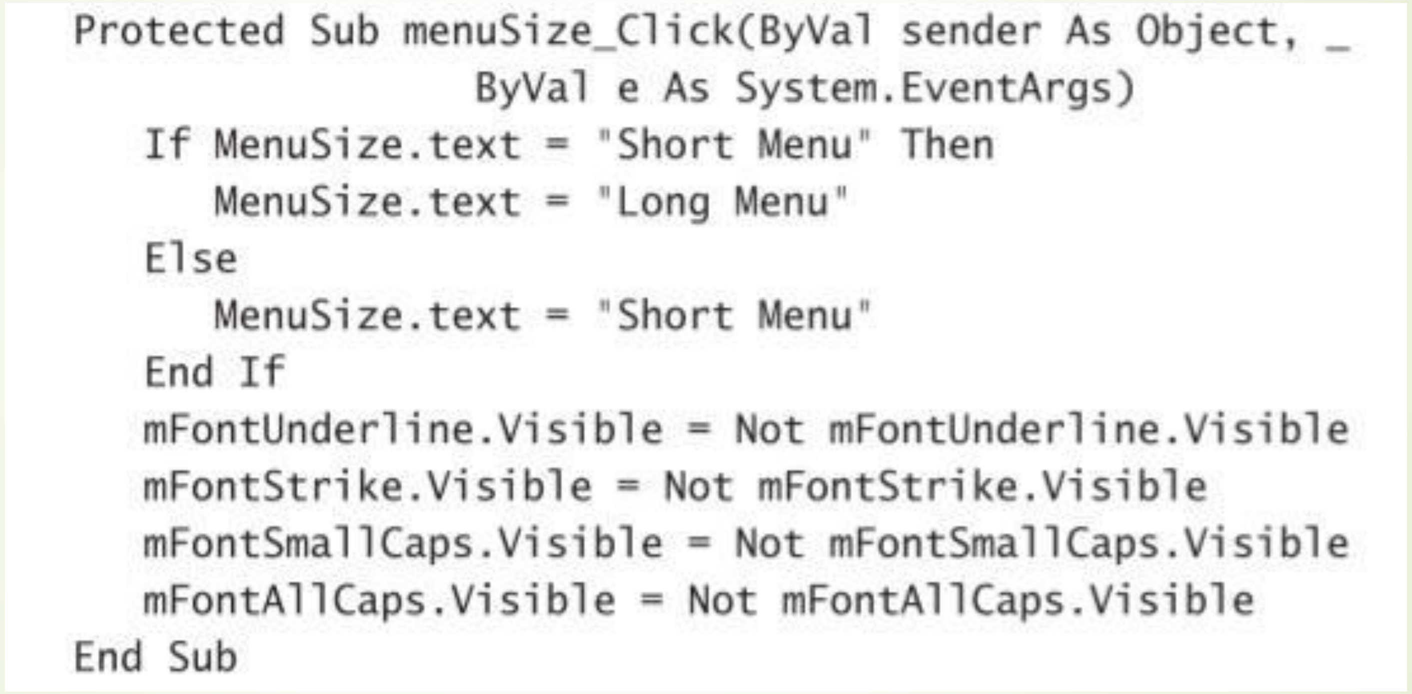


Returning Multiple Values

- Stats() function: return Avg and StDev

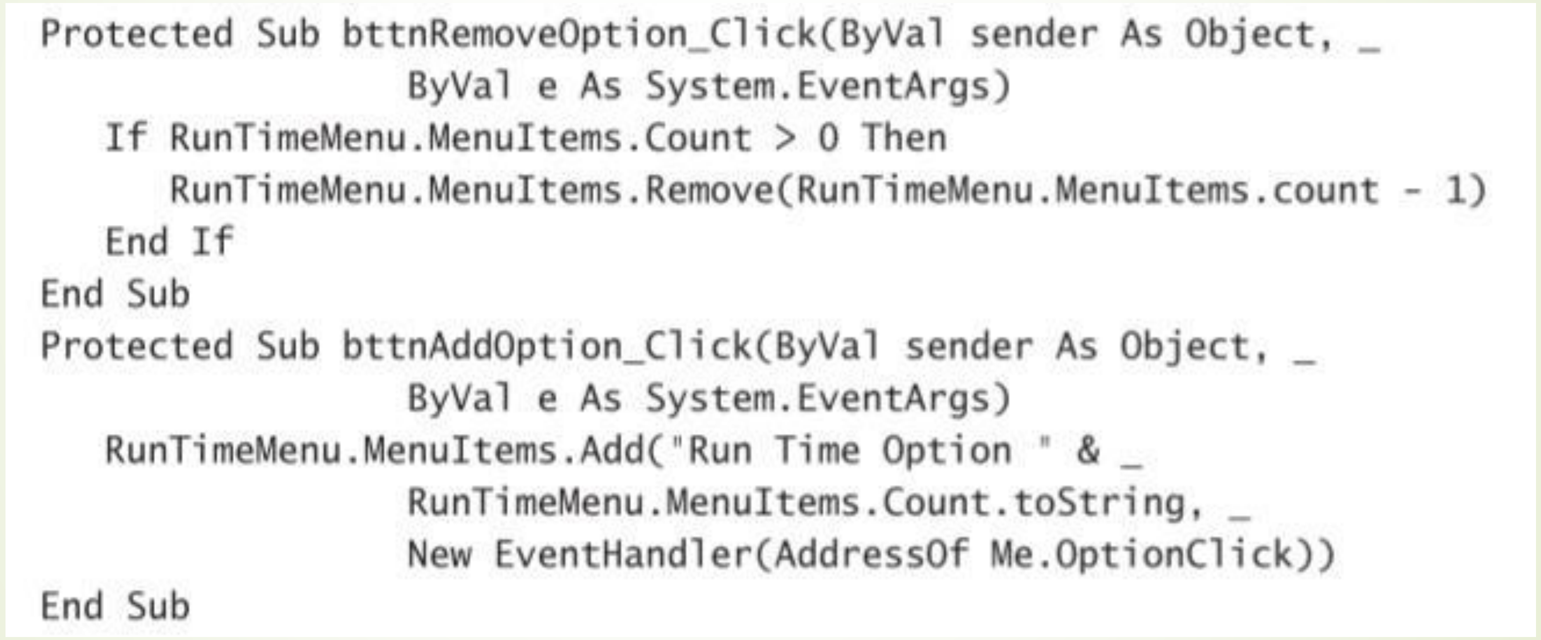
```
Function Stats(ByRef Data() As Double, ByRef Avg As Double, _
               ByRef StDev As Double) As Integer
   Dim i As Integer, sum As Double, sumSqr As Double, points As Integer
   points = Data.Length
   For i = 0 To points - 1
      sum = sum + Data(i)
      sumSqr = sumSqr + Data(i) ^ 2
   Next
   Avg = sum / points
   StDev = System.Math.Sqrt(sumSqr / points - Avg ^ 2)
   Return(points)
End Function
```

The MenuSize Menu Item’s Click Event



```
Protected Sub menuSize_Click(ByVal sender As Object, _
                 ByVal e As System.EventArgs)
   If MenuSize.text = "Short Menu" Then
      MenuSize.text = "Long Menu"
   Else
      MenuSize.text = "Short Menu"
   End If
   mFontUnderline.Visible = Not mFontUnderline.Visible
   mFontStrike.Visible = Not mFontStrike.Visible
   mFontSmallCaps.Visible = Not mFontSmallCaps.Visible
   mFontAllCaps.Visible = Not mFontAllCaps.Visible
End Sub
```

Adding and Removing MenuItems at Runtime



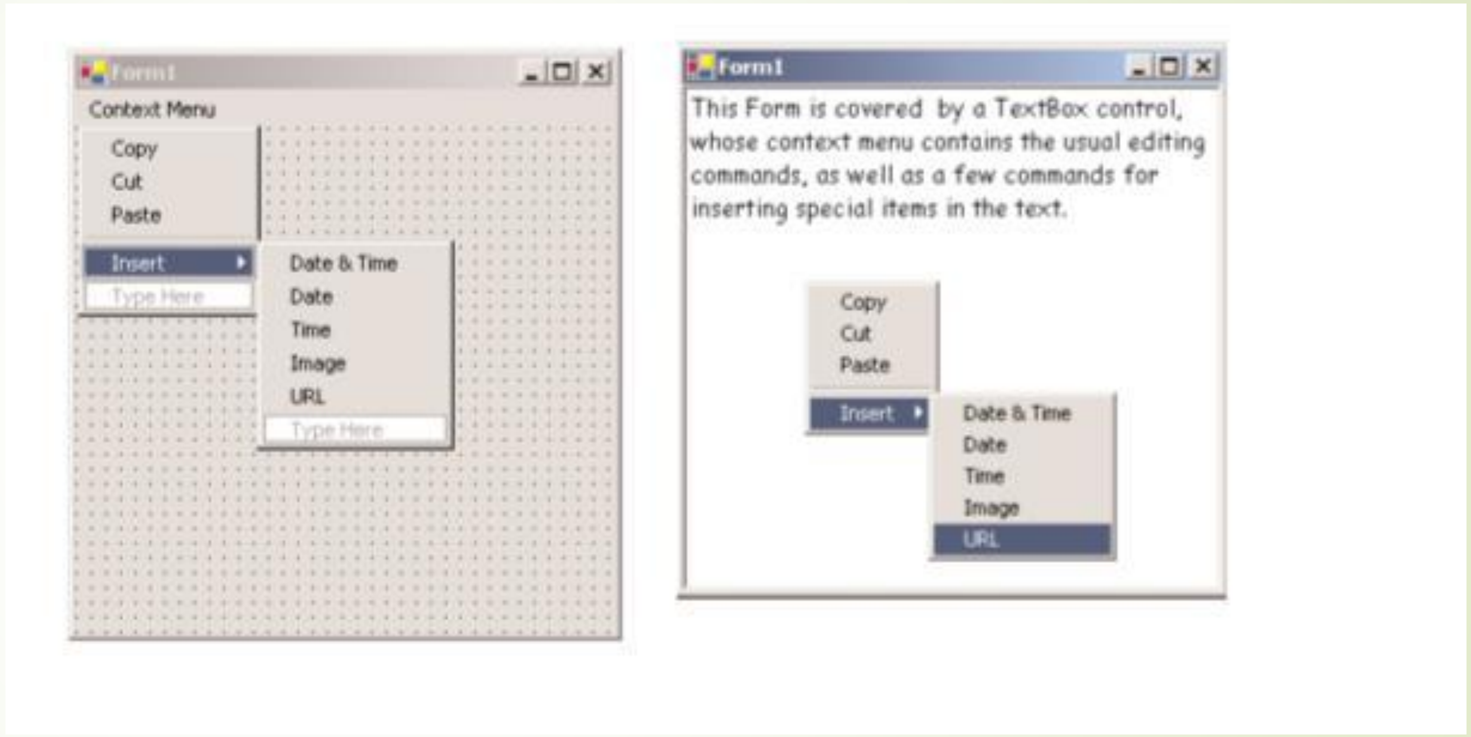
```
Protected Sub bttnRemoveOption_Click(ByVal sender As Object, _
                 ByVal e As System.EventArgs)
   If RunTimeMenu.MenuItems.Count > 0 Then
      RunTimeMenu.MenuItems.Remove(RunTimeMenu.MenuItems.count - 1)
   End If
End Sub
Protected Sub bttnAddOption_Click(ByVal sender As Object, _
                 ByVal e As System.EventArgs)
   RunTimeMenu.MenuItems.Add("Run Time Option " & _
                 RunTimeMenu.MenuItems.Count.toString, _
                 New EventHandler(AddressOf Me.OptionClick))
End Sub
```

Programming Dynamic Menu Items

```
Private Sub OptionClick(ByVal sender As Object, ByVal e As EventArgs)
   Dim itemClicked As New MenuItem()
   itemClicked = CType(sender, MenuItem)
   Console.WriteLine("You have selected the item " & itemClicked.Text)
End Sub
```

Creating Context Menus

- A context menu, (left) at design time and (right) at runtime



This Form is covered by a TextBox control, whose context menu contains the usual editing commands, as well as a few commands for inserting special items in the text.

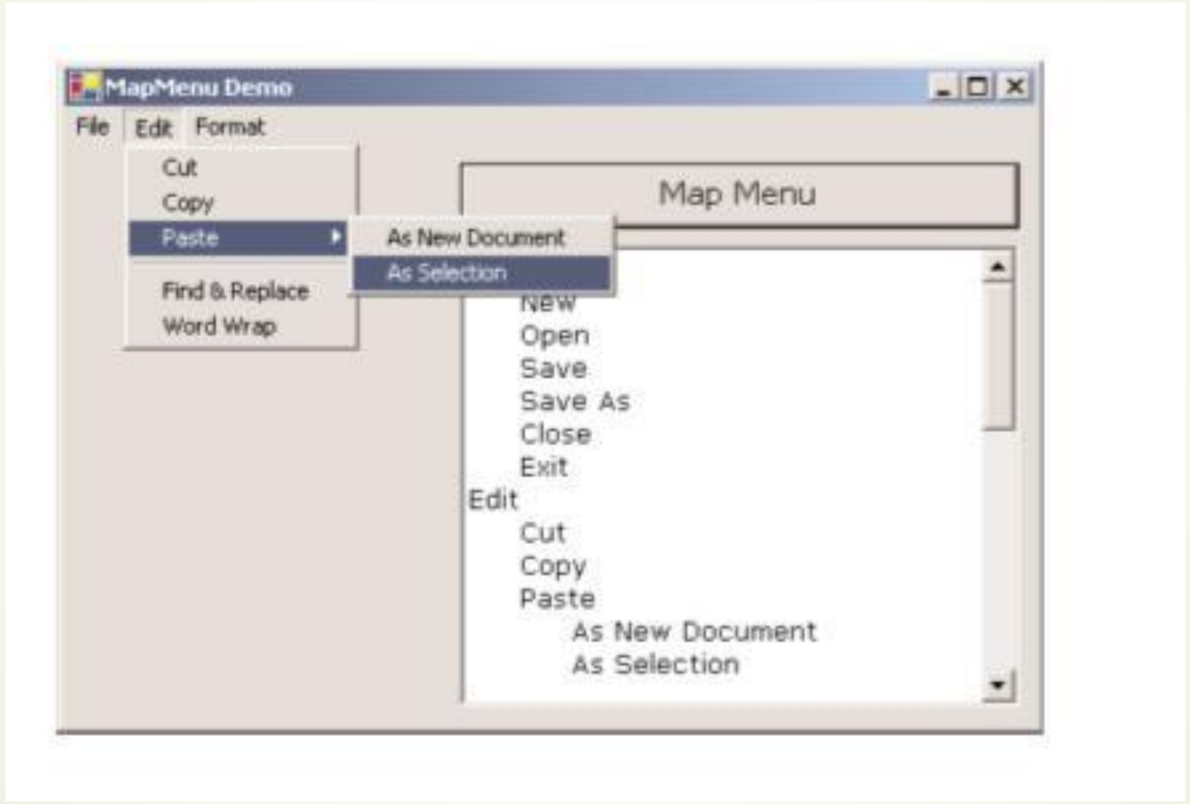
Iterating a Menu’s Items

- The first command in the File menu can be accessed by the expression
  - Me.Menu.MenuItems(0).MenuItems(0)
- The second command on the same level as the File command (typically, the Edit menu).
  - Me.Menu.MenuItems(1)
- The same items can be accessed by name as well

Exercise: The MapMenu Project

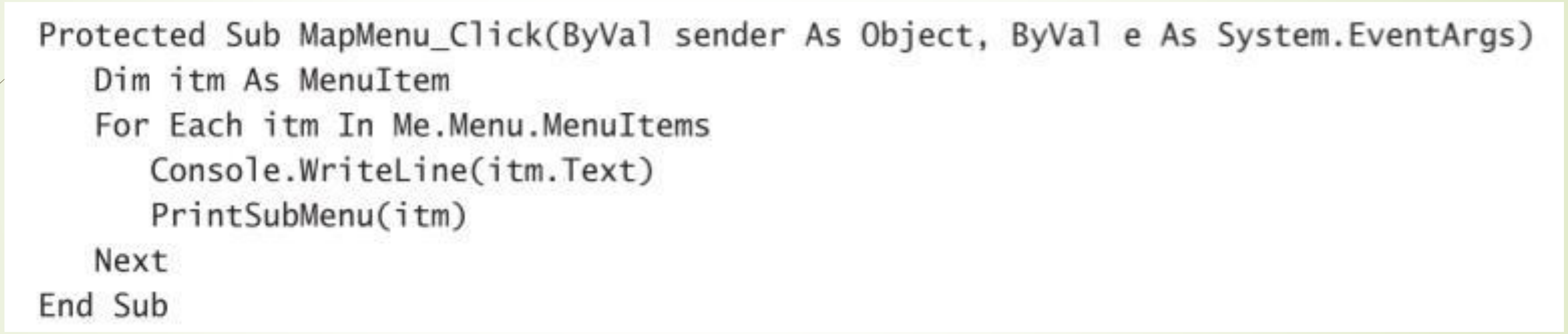
- Access the items of a menu from within your application’s code.
- A menu, a TextBox control, and a Button
  - Prints the menu’s structure on the TextBox.

The MapMenu application



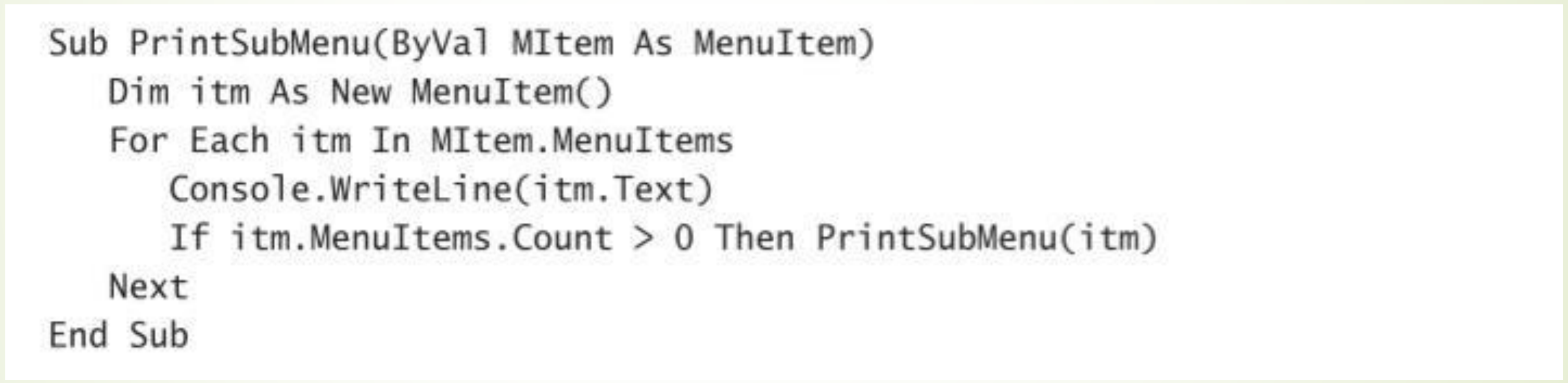
```
    New
    Open
    Save
    Save As
    Close
    Exit
Edit
    Cut
    Copy
    Paste
        As New Document
        As Selection
```

Printing the Top-Level Commands of a Menu



```
Protected Sub MapMenu_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim itm As MenuItem
    For Each itm In Me.Menu.MenuItems
        Console.WriteLine(itm.Text)
        PrintSubMenu(itm)
    Next
End Sub
```

Printing Submenu Items

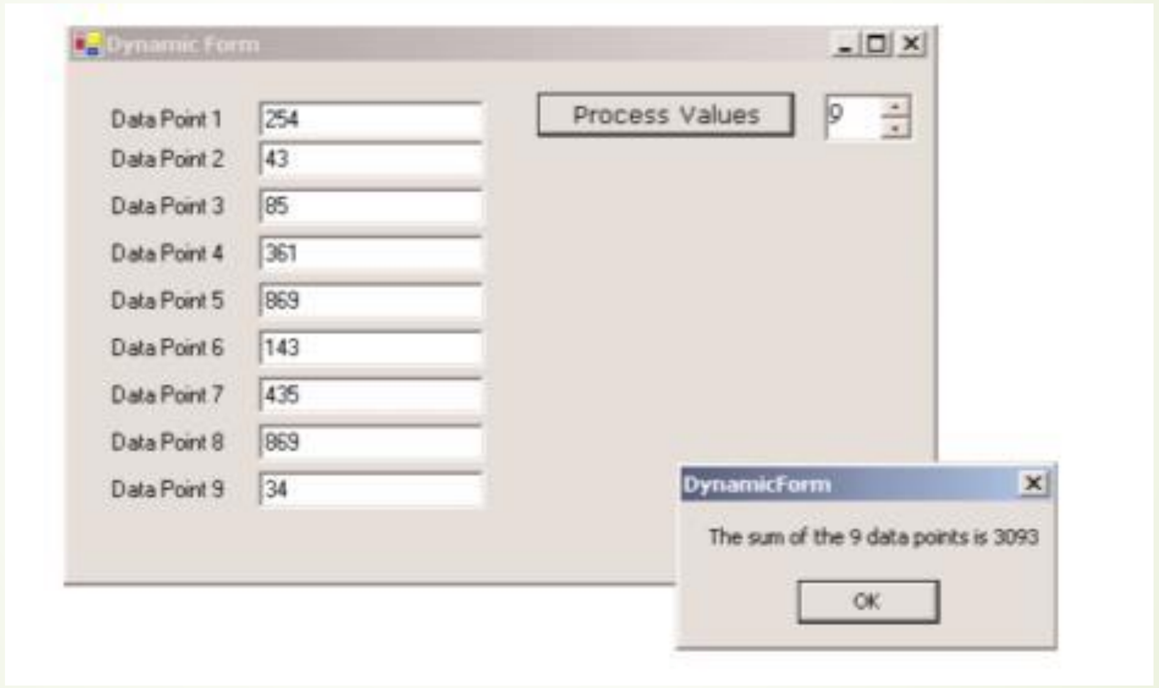


```
Sub PrintSubMenu(ByVal MItem As MenuItem)
   Dim itm As New MenuItem()
   For Each itm In MItem.MenuItems
      Console.WriteLine(itm.Text)
      If itm.MenuItems.Count > 0 Then PrintSubMenu(itm)
   Next
End Sub
```

Building Dynamic Forms at Runtime

- The Form.Controls Collection
  - Add method
  - Remove method
  - Count property
  - All method
  - Clear method

Exercise: The DynamicForm Project

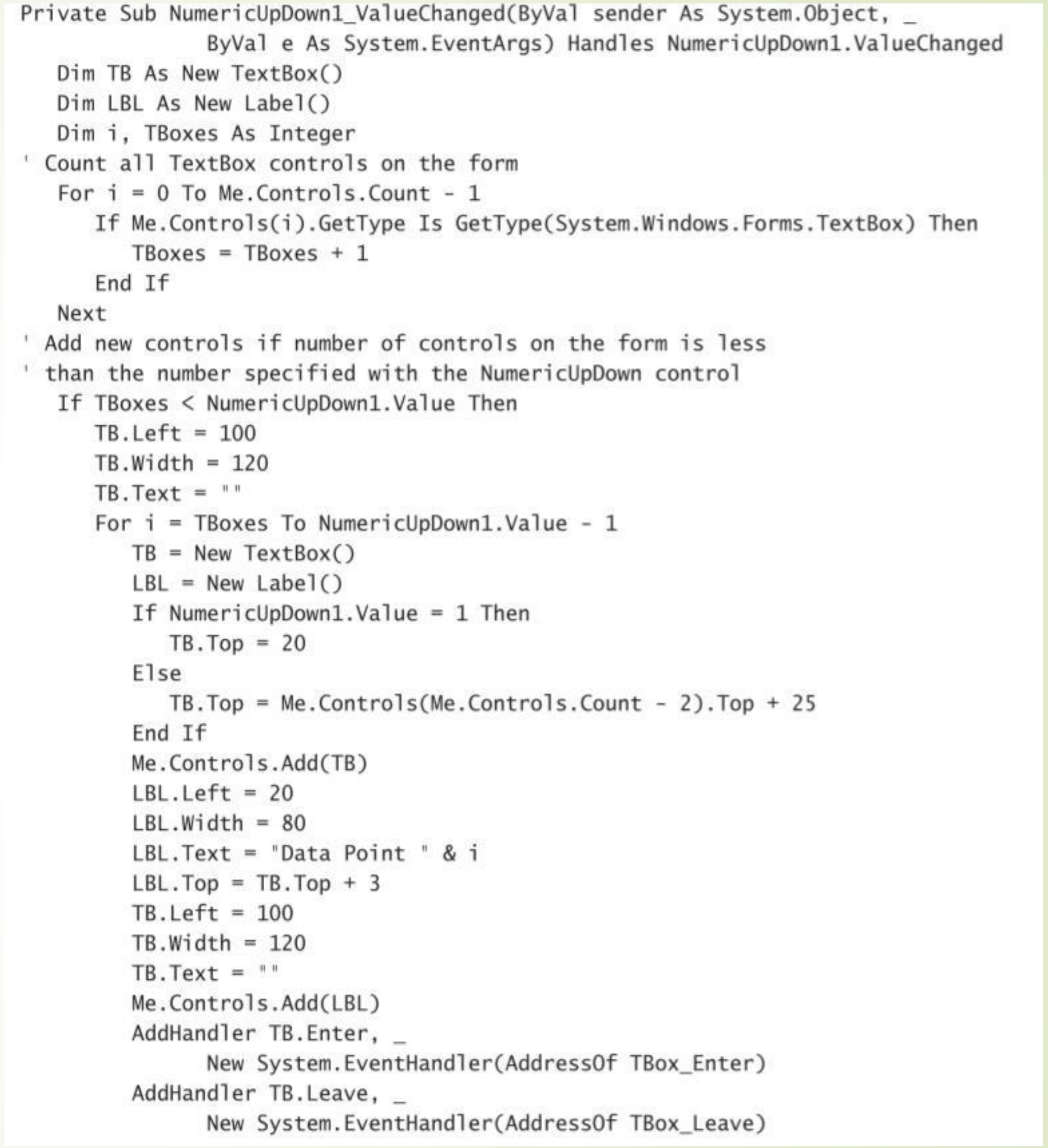


Data Point 1	254
Data Point 2	43
Data Point 3	85
Data Point 4	361
Data Point 5	869
Data Point 6	143
Data Point 7	435
Data Point 8	869
Data Point 9	34

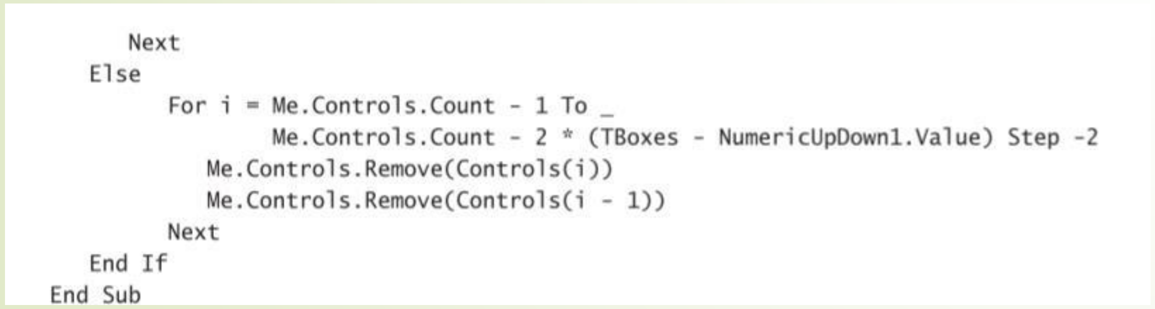
9

The sum of the 9 data points is 3093

Adding and Removing Controls at Runtime



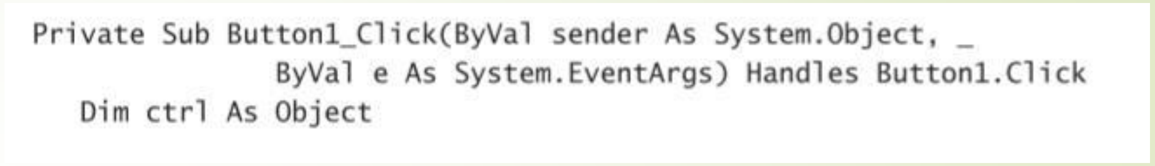
```
Private Sub NumericUpDown1_ValueChanged(ByVal sender As System.Object, _
                ByVal e As System.EventArgs) Handles NumericUpDown1.ValueChanged
   Dim TB As New TextBox()
   Dim LBL As New Label()
   Dim i, TBoxes As Integer
' Count all TextBox controls on the form
   For i = 0 To Me.Controls.Count - 1
      If Me.Controls(i).GetType Is GetType(System.Windows.Forms.TextBox) Then
         TBoxes = TBoxes + 1
      End If
   Next
' Add new controls if number of controls on the form is less
' than the number specified with the NumericUpDown control
   If TBoxes < NumericUpDown1.Value Then
      TB.Left = 100
      TB.Width = 120
      TB.Text = ""
      For i = TBoxes To NumericUpDown1.Value - 1
         TB = New TextBox()
         LBL = New Label()
         If NumericUpDown1.Value = 1 Then
            TB.Top = 20
         Else
            TB.Top = Me.Controls(Me.Controls.Count - 2).Top + 25
         End If
         Me.Controls.Add(TB)
         LBL.Left = 20
         LBL.Width = 80
         LBL.Text = "Data Point " & i
         LBL.Top = TB.Top + 3
         TB.Left = 100
         TB.Width = 120
         TB.Text = ""
         Me.Controls.Add(LBL)
         AddHandler TB.Enter, _
               New System.EventHandler(AddressOf TBox_Enter)
         AddHandler TB.Leave, _
               New System.EventHandler(AddressOf TBox_Leave)
```



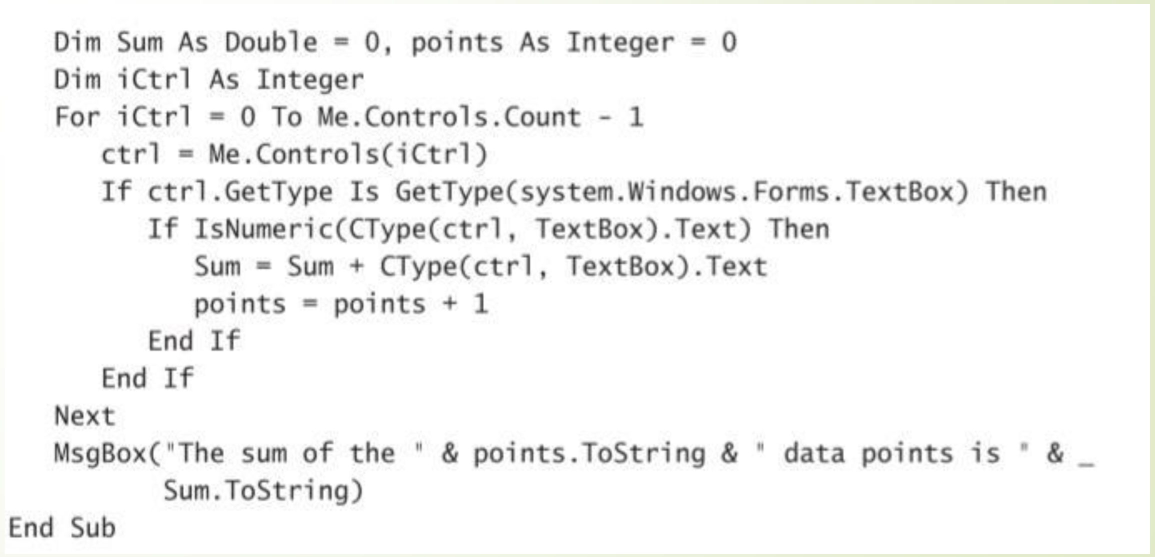
```
      Next
   Else
         For i = Me.Controls.Count - 1 To _
                 Me.Controls.Count - 2 * (TBoxes - NumericUpDown1.Value) Step -2
            Me.Controls.Remove(Controls(i))
            Me.Controls.Remove(Controls(i - 1))
         Next
   End If
End Sub
```

Reading the Controls on the Form

- These three properties return a True/False value indicating whether one or more of the control keys were down when the key was pressed.



```
Private Sub Button1_Click(ByVal sender As System.Object, _
                ByVal e As System.EventArgs) Handles Button1.Click
   Dim ctrl As Object
```



```
   Dim Sum As Double = 0, points As Integer = 0
   Dim iCtrl As Integer
   For iCtrl = 0 To Me.Controls.Count - 1
      ctrl = Me.Controls(iCtrl)
      If ctrl.GetType Is GetType(system.Windows.Forms.TextBox) Then
         If IsNumeric(CType(ctrl, TextBox).Text) Then
            Sum = Sum + CType(ctrl, TextBox).Text
            points = points + 1
         End If
      End If
   Next
   MsgBox("The sum of the " & points.ToString & " data points is " & _
          Sum.ToString)
End Sub
```

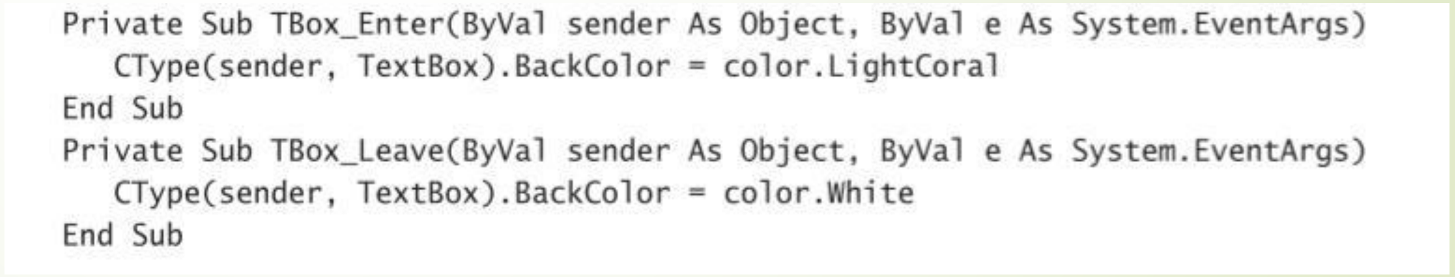
Reading the Controls with a For Each…Next Loop

```
Private Sub bttnProcess2_Click(ByVal sender As System.Object, _
               ByVal e As System.EventArgs) Handles bttnProcess2.Click
   Dim TB As Control
   Dim Sum As Double = 0, points As Integer = 0
   For Each TB In Me.Controls
      If TB.GetType Is GetType(Windows.Forms.TextBox) Then
         If IsNumeric(CType(TB, TextBox).Text) Then
            Sum = Sum + CType(TB, TextBox).Text
            points = points + 1
         End If
      End If
   Next
   MsgBox("The sum of the " & points.ToString & " data points is " & _
          Sum.ToString)
End Sub
```

Creating Event Handlers at Runtime

- The statement that connects a control's event to a specific event handler, is the AddHandler statement, whose syntax is:
  - AddHandler control.event, New System.EventHandler(AddressOf subName)

- Event Handlers Added at Runtime



```
Private Sub TBox_Enter(ByVal sender As Object, ByVal e As System.EventArgs)
   CType(sender, TextBox).BackColor = color.LightCoral
End Sub
Private Sub TBox_Leave(ByVal sender As Object, ByVal e As System.EventArgs)
   CType(sender, TextBox).BackColor = color.White
End Sub
```