Chapter 3:
Fundamentals of Programming in VB.NET

- VB.NET Controls (3.1)
- VB.NET Events (3.2)
- Numbers (3.3)
- Strings (3.4)
- Input and Output (3.5)

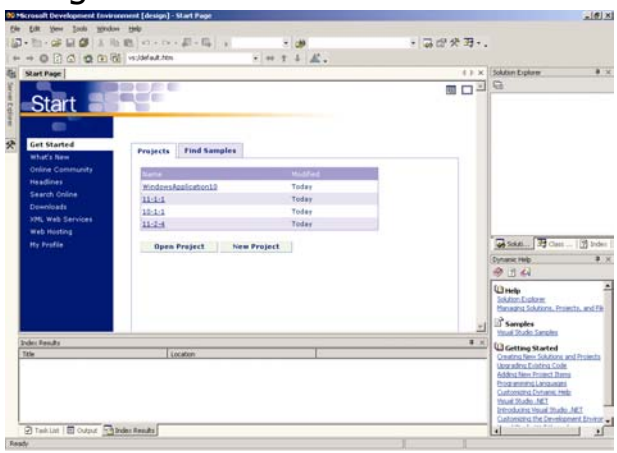
David I. Schneider, *An Introduction to Programming using Visual Basic.NET, 5th Edition*, Prentice Hall, 2002.

VB.NET Controls (3.1)

- Invoking VB.NET
- A Text Box Walkthrough
- A Button Walkthrough
- A Label Walkthrough
- A List Box Walkthrough
- The Name Property
- A Help Walkthrough
- Fonts / Auto Hide

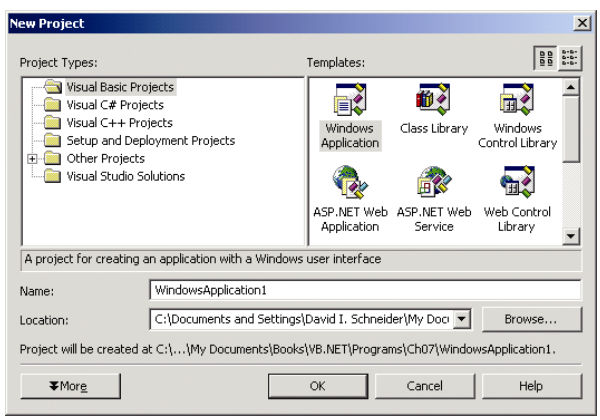
VB.NET Controls (3.1) (cont.)

- Invoking VB.NET



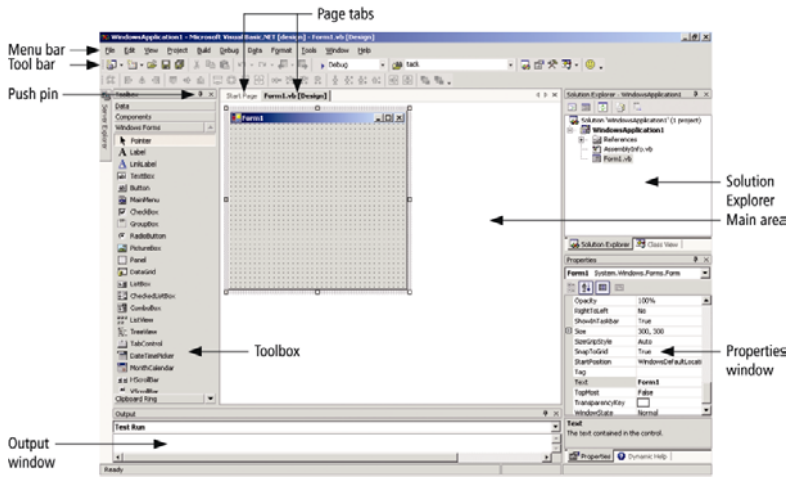
VB.NET Controls (3.1) (cont.)

- Create a New Project



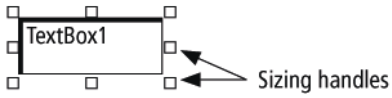
VB.NET Controls (3.1) (cont.)

- Initial VB.NET Screen



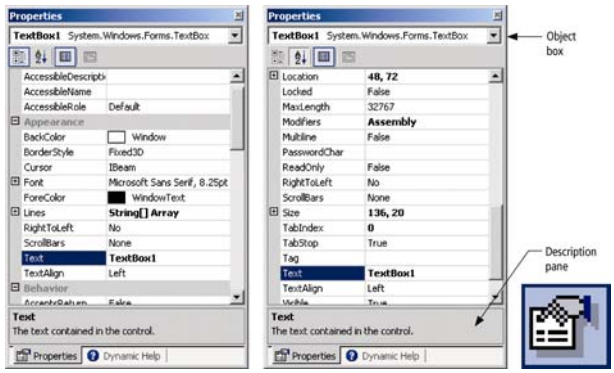
VB.NET Controls (3.1) (cont.)

- A Text Box Walkthrough
  - In the ToolBox, double click the Text Box icon
  - The control is selected when you see the sizing handles
  - Press the Del key to delete



VB.NET Controls (3.1) (cont.)

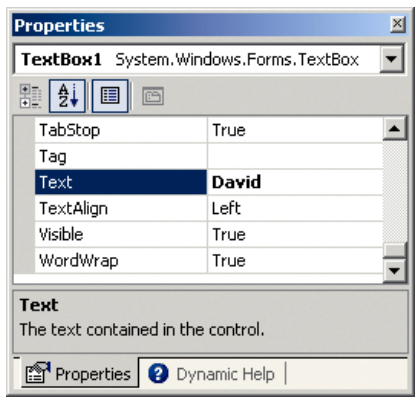
- Text Box Properties



Categorized view Alphabetical view

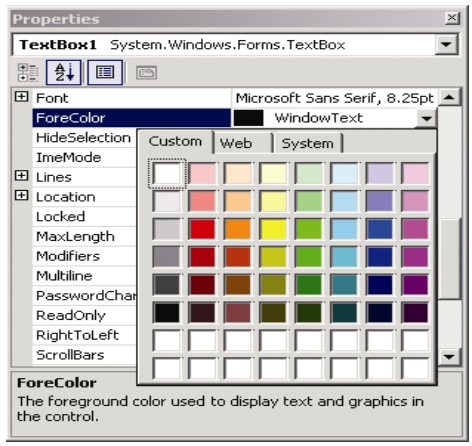
VB.NET Controls (3.1) (cont.)

- Changing Properties



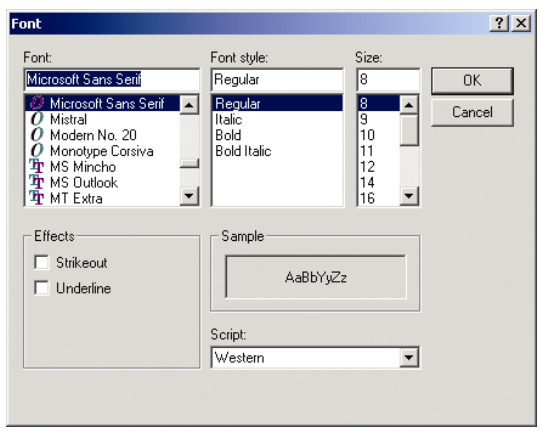
VB.NET Controls (3.1) (cont.)

- ForeColor Property



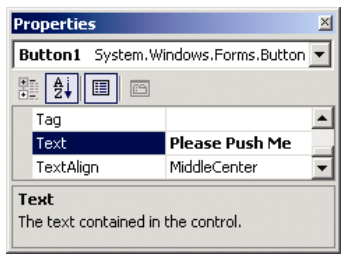
VB.NET Controls (3.1) (cont.)

- Font Property



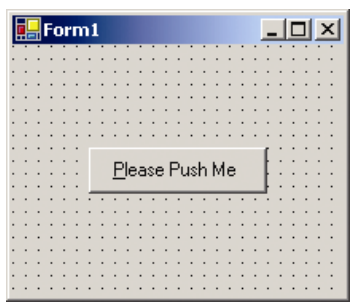
VB.NET Controls (3.1) (cont.)

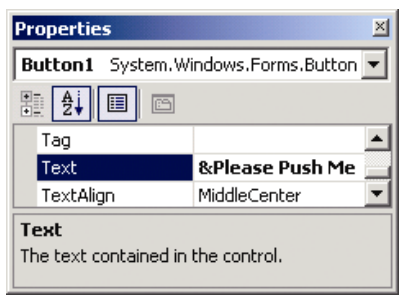
- A Button Walkthrough
  - Add the button
  - Change the Text property



VB.NET Controls (3.1) (cont.)

- Add an "access key"





VB.NET Controls (3.1) (cont.)

- A Label Walkthrough
  - Add the Label
  - Change the Text property
  - Resize the control

VB.NET Controls (3.1) (cont.)

- A List Box Walkthrough
  - Add the List Box
  - Change the Text property
  - Resize the control

VB.NET Controls (3.1) (cont.)

- The Name Property
  - How the programmer refers to a control in code
  - Name must begin with a letter
  - Must be less than 215 characters long
  - May include numbers and the underscore
  - Use appropriate 3 character naming prefix

VB.NET Controls (3.1) (cont.)

- Control Name Prefixes

Control	Prefix	Example
Button	Btn	BtnCompute Total
Label	Lbl	LblInstructions
List box	Lst	LstOutput
Text box	txt	TxtAddress

VB.NET Controls (3.1) (cont.)

- Fonts
  - Proportional width fonts take up less space for "I" than for "W" – like Microsoft Sans Serif
  - Fixed-width fonts take up the same amount of space for each character – like Courier New
  - Fixed-width fonts are good for tables

VB.NET Controls (3.1) (cont.)

- Auto Hide
  - Hides tool windows when not in use
  - Vertical push pin icon indicates auto hide is disabled
  - Click the push pin to make it horizontal and enable auto hide

VB.NET Events (3.2)

- An Event Procedure Walkthrough
- Properties and Event Procedures of the Form
- The Declaration Statement of an Event Procedure

VB.NET Events (3.2) (cont.)

- An Event Procedure Walkthrough
  - An event is an action, such as the user clicking on a button
  - Usually, nothing happens until the user does something and generates an event

VB.NET Events (3.2) (cont.)

- The three steps in creating a VB.NET program:
  1. Create the interface; that is, generate, position, and size the objects.
  2. Set properties; that is, configure the appearance of the objects.
  3. Write the code that executes when events occur.

VB.NET Events (3.2) (cont.)

- Changing Properties
  - Properties are changed in code with the following:

```
controlName.property = setting
```

  - This is an assignment statement

```
txtBox.ForeColor = Color.Red
```

VB.NET Events (3.2) (cont.)

- Event Procedures

```
Private Sub objectName_event(ByVal sender
  As System.Object, ByVal e As
  System.EventArgs) Handles
  objectName.event
```

Shown in the book as:

```
Private Sub objectName_event(…) Handles
  objectName.event
```

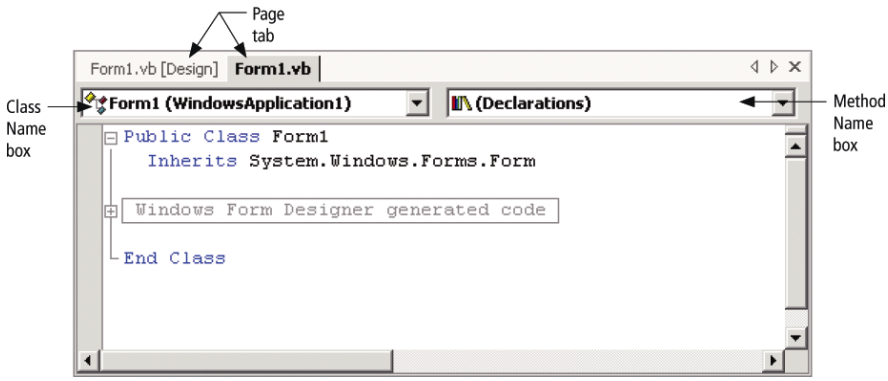
VB.NET Events (3.2) (cont.)

- Structure of an Event Procedure

```
Private Sub objectName_event(...)
    Handles objectName.event
  statements
End Sub
```

VB.NET Events (3.2) (cont.)

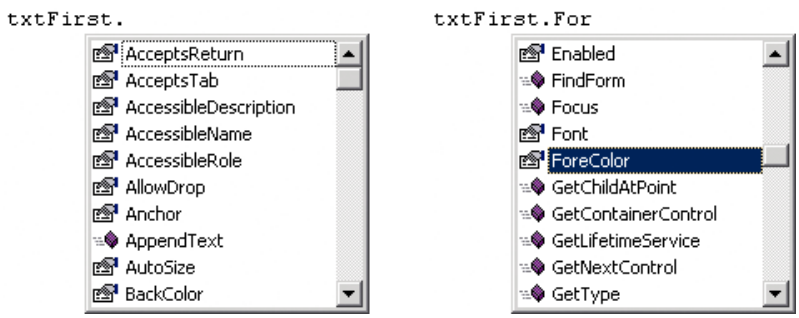
- Program Region



VB.NET Events (3.2) (cont.)

- IntelliSense

Automatically pops up to give the programmer help.



VB.NET Events (3.2) (cont.)

- Code for Walkthrough

```
Private Sub txtFirst_TextChanged(...)
    Handles txtFirst.TextChanged
  txtFirst.ForeColor = Color.Blue
End Sub

Private Sub btnRed_Click(...)
   Handles  btnRed.Click
  txtFirst.ForeColor = Color.Red
End Sub

Private Sub txtFirst_Leave(...)
    Handles txtFirst.Leave
  txtFirst.ForeColor = Color.Black
End Sub
```

VB.NET Events (3.2) (cont.)

- Assigning properties in code
  - The following won't work:

```
Form1.Text = "Demonstration"
```

  - The form is referred to by the keyword *Me*.

```
Me.Text = "Demonstration"
```

VB.NET Events (3.2) (cont.)

- The Declaration Statement of an Event Procedure
  - A declaration statement for an event procedure:

```
Private Sub btnOne_Click(...) Handles
  btnOne.Click
```

  - The name can be changed at will. For example

```
Private Sub ButtonPushed(...) Handles
  btnOne.Click
```

  - Handling more than one event:

```
Private Sub ButtonPushed(...) Handles
  btnOne.Click, btnTwo.Click
```

Numbers (3.3)

- Arithmetic Operations
- Variables
- Incrementing the Value of a Variable
- Built-In Functions:
  - Math.Sqrt
  - Int
  - Math.Round

Numbers (3.3) (cont.)

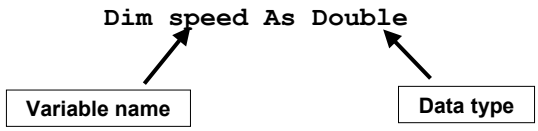
- The Integer Data Type
- Multiple Declarations
- Parentheses
- Three Types of Errors

Numbers (3.3) (cont.)

- Arithmetic Operations
  - Numbers are called *numeric literals*
  - Five arithmetic operations in VB.NET
    - + addition
    - - subtraction
    - * multiplication
    - / division
    - ^ exponentiation

Numbers (3.3) (cont.)

- Variables
  - Declaration:



- Assignment:

```
speed = 50
```

Numbers (3.3) (cont.)

- Initialization
  - Numeric variables are automatically initialized to 0:

```
Dim varName As Double
```

  - To specify a nonzero initial value

```
Dim varName As Double = 50
```

Numbers (3.3) (cont.)

- Incrementing
  - To add 1 to the numeric variable *var*

    var = var + 1
  - Or as a shortcut

    var +=1

Numbers (3.3) (cont.)

- Built-in Functions
  - Functions *return* a value

    Math.Sqrt(9) returns 3
    Int(9.7) returns 9
    Math.Round(2.7) is 3

Numbers (3.3) (cont.)

- Integer Data Type
  - An integer is a whole number
  - Declaring an integer variable:

```
Dim varName As Integer
```

Numbers (3.3) (cont.)

- Multiple Declarations

```
Dim a, b As Double
```

Two other types of multiple-declaration statements are

```
Dim a As Double, b As Integer
Dim c As Double = 2, b As Integer = 5
```

Numbers (3.3) (cont.)

- Three Types of Errors
  - Syntax error
  - Run-time error
  - Logic error

Strings (3.4)

- Variables and Strings
- Using Text Boxes for Input and Output
- Concatenation
- ANSI Character Set
- String Properties and Methods:

Length	ToUpper
Trim	ToLower
IndexOf	Substring

Strings (3.4) (cont.)

- The Empty String
- Initial Value of a String
- Option Strict
- Internal Documentation
- Line-Continuation Character

Strings (3.4) (cont.)

- Variables and Strings

```
Private Sub btnDisplay_Click(...) Handles
  btnDisplay.Click
      Dim today As String
      today = "Monday"
      With lstOutput.Items
            .Clear()
            .Add("hello")
            .Add(today)
      End With
End Sub
```

Strings (3.4) (cont.)

- Using Text Boxes for Input and Output
  - The contents of a text box is always a string
  - Input example

```
strVar = txtBox.Text
```

  - Output example

```
txtBox.Text = strVar
```

Strings (3.4) (cont.)

- Data Conversion
  - Because the contents of a text box is always a string, sometimes you must convert the input or output

```
numVar = CDbl(txtBox.Text)
```

Converts a String to a Double

```
txtBox.Text = CStr(numVar)
```

Converts a number to a string

Strings (3.4) (cont.)

- Concatenation
  - Combining two strings to make a new string

```
quote1 = "The ballgame isn't over, "
quote2 = "until it's over."
quote = quote1 & quote2
txtOutput.Text = quote & " Yogi Berra"
```

  - Displays

```
The ball game isn't over until it's over. Yogi
  Berra
```

Strings (3.4) (cont.)

- ANSI Character Set
  - A numeric representation for every key on the keyboard

32 (space)	48 O	66 B	122 z
33 !	49 1	90 Z	123 {
34 "	57 9	97 a	125 }
35 #	65 A	98 b	126 -

Strings (3.4) (cont.)

- String Properties and Methods:

"Visual".Length is 6.

"Visual".ToUpper is VISUAL.

"123 Hike".Length is 8.

"123 Hike".ToLower is 123 hike.

"a" & " bcd ".Trim & "efg" is abcdefg.

Strings (3.4) (cont.)

- More String Properties and Methods:

"fanatic".Substring(0, 3) is "fan".

"fanatic".IndexOf("ati") is 3.

"fanatic".Substring(4, 2) is "ti".

"fanatic".IndexOf("a") is 1.

"fanatic".Substring(4) is "tic".

"fanatic".IndexOf("nt") is –1.

Strings (3.4) (cont.)

- The Empty String
  - The string "", which contains no characters, is called the empty string or the zero-length string.
  - The statement lstBox.Items.Add("") skips a line in the list box.
  - The contents of a text box can be cleared with either the statement

```
txtBox.Clear()
```

  - or the statement

```
txtBox.Text = ""
```

Strings (3.4) (cont.)

- Initial Value of a String
  - By default the initial value is **Nothing**
  - Strings can be given a different initial value as follows:

```
Dim today As String = "Monday"
```

Strings (3.4) (cont.)

- Option Strict
  - VB.NET allows numeric variables to be assigned strings and vice versa, a poor programming practice.
  - To turn this feature off, put the following statement at the very top of the code window

```
Option Strict On
```

Strings (3.4) (cont.)

- Internal Documentation
  1. Other people can easily understand the program.
  2. You can understand the program when you read it later.
  3. Long programs are easier to read because the purposes of individual pieces can be determined at a glance.

Strings (3.4) (cont.)

- Line-Continuation Character
  - A long line of code can be continued on another line by using underscore (_) preceded by a space

```
msg = "640K ought to be enough " & _
"for anybody. (Bill Gates, 1981)"
```

Input and Output (3.5)

- Formatting Output with Format Functions
- Formatting Output with Zones
- Reading Data from Files
- Getting Input from an Input Dialog Box
- Using a Message Dialog Box for Output

Input and Output (3.5) (cont.)

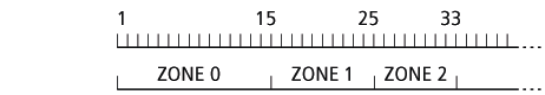
- Formatting Output with Format Functions

Function	String Value
FormatNumber(12345.628,1)	12,345.6
FormatCurrency(12345.628,2)	$12,345.63
FormatPercent(0.185,2)	18.50%

Input and Output (3.5) (cont.)

- Formatting Output with Zones
  - Use a fixed-width font such as Courier New
  - Divide the characters into zones with a format string.

```
Dim fmtStr As String = "{0, 15}{1, 10}{2, 8}"
lstOutput.Items.Add(String.Format(fmtStr, data0,
   data1, data2))
```



Input and Output (3.5) (cont.)

- Inputting Data
  - Data can be stored in files and accessed with a StreamReader object or supplied by the user with an input dialog box.

Input and Output (3.5) (cont.)

Steps to Use StreamReader

1. Execute a statement of the form

```
Dim readerVar As IO.StreamReader  = _
      IO.File.OpenText(filespec)
```

or the pair of statements

```
Dim readerVar As IO.StreamReader
readerVar = IO.File.OpenText(filespec)
```

2. Assume the file contains one item of data per line.
   Read items of data in order, one at a time, from the file with the ReadLine method.

```
strVar = readerVar.ReadLine
```

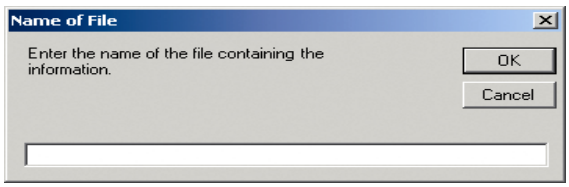
3. After the desired items have been read from the file, terminate the communications link

```
readerVar.Close()
```

Input and Output (3.5) (cont.)

- Getting Input from an Input Dialog Box

```
stringVar = InputBox(prompt, title)
fileName = InputBox("Enter the name " _
      & "of the file containing the " & _
       "information.", "Name of File")
```



Input and Output (3.5) (cont.)

- Using a Message Dialog Box for Output
  - `MsgBox(prompt, , title)`

```
MsgBox("Nice try, but no cigar.", ,
  "Consolation")
```