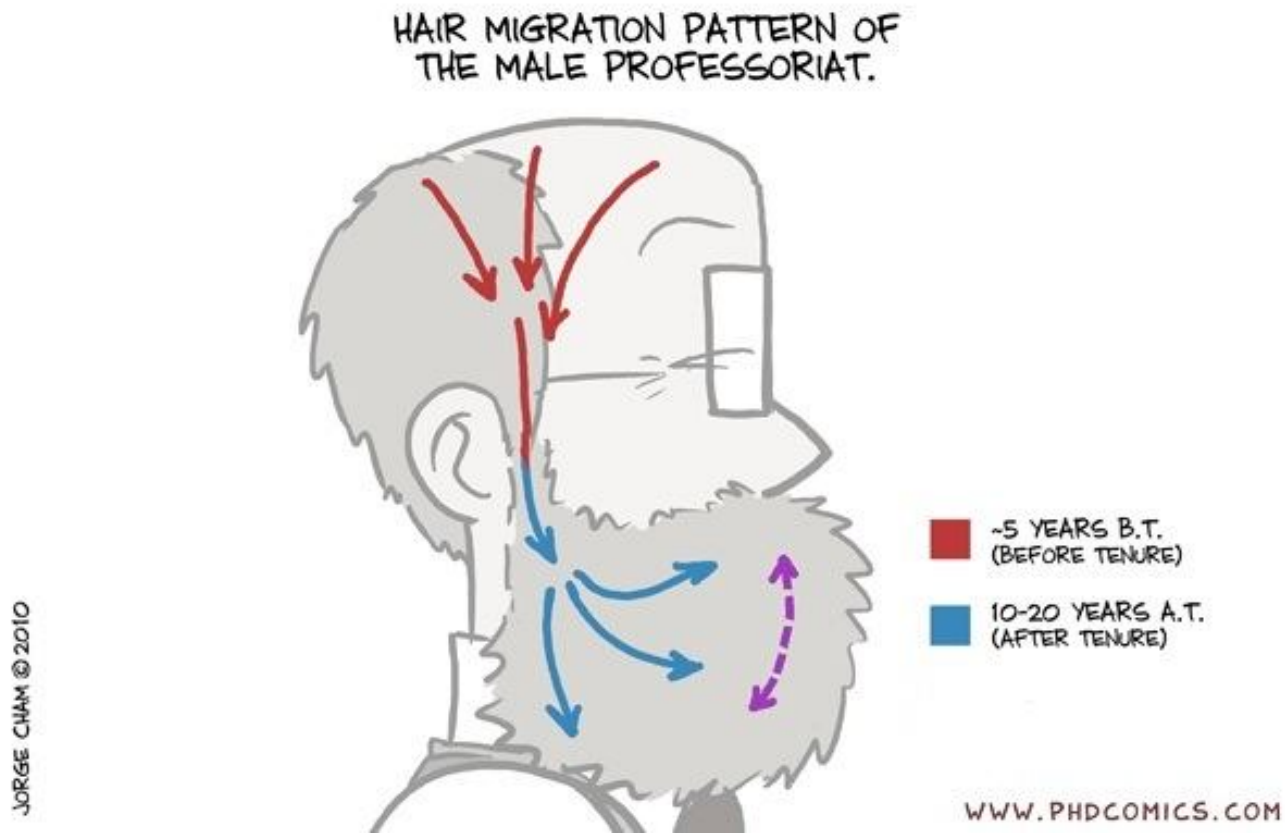
Visual Basic - Chapter 4



Mohammad Shokoohi

* Adopted from An Introduction to Programming Using Visual Basic 2010, Schneider

Chapter 4 – Decisions

4.1 Relational and Logical Operators

4.2 If Blocks

4.3 Select Case Blocks

4.4 Input via User Selection

4.1 Relational and Logical Operators

- ANSI Values
- Relational Operators
- Logical Operators
- Boolean Data Type
- Two Boolean-Valued Methods
- A Boolean-Valued Function

Condition

- A **condition** is an expression involving relational and/or logical operators
- The value of the condition is Boolean – that is, True or False

ANSI Character Set

A numeric representation for every key on the keyboard and for other assorted characters.

32 (space)	48 0	66 B	122 z
33 !	49 1	90 Z	123 {
34 “	57 9	97 a	125 }
35 #	65 A	98 b	126 ~

ANSI Character Set (continued)

A numeric representation for every key on the keyboard and for **other assorted characters**.

162 ¢	177 ±	181 µ	190 ¼
169 ©	178 ²	188 ¼	247 ÷
176 °	179 ³	189 ½	248 ø

Chr Function

For *n* between 0 and 255,

```
Chr(n)
```

is the string consisting of the character with ANSI value *n*.

Examples:

```
Chr(65) is A
Chr(162) is ¢
```

Asc Function

For a string *str*,

`Asc(str)`

is ANSI value of the first character of *str*.

Examples: **`Asc("A")`** is **`65`**

`Asc("¢25")` is **`162`**

Relational Operators

<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
=	equal to
<>	not equal to

ANSI values are used to decide order for strings.

Condition

- A **condition** is an expression involving relational and/or logical operators.
- Result of the condition is *True* or *False*.

Example

When a = 3, b = 4

(a + b) < 2 * a

3 + 4 = 7

2 * 3 = 6

7 is NOT less than 6 and so the value of the expression is False

Another Example

a = 4 b = 3 c = "hello" d = "bye"

(c.Length – b) = (a / 2)

5 – 3 = 2

4 / 2 = 2

True because 2 equals 2

Relational Operator Notes

- Relational operators are binary – they require an operand on both sides of the operator
- Value of a relational expression will always be True or False

Logical Operators

Used with Boolean expressions

- *Not* – makes a False expression True and vice versa
- *And* – will yield a True if and only if both expressions are True
- *Or* – will yield a True if at least one of both expressions are True

Example 4.3

n = 4, *answ* = “Y” Are the following expressions true or false?

Not (n < 6)

(answ = "Y") Or (answ = "y")

(answ = "Y") And (answ = "y")

Not(answ = "y")

Boolean Expression

- An expression that evaluates to either True or False is said to have Boolean data type.
- Example*:*

  The statement

```
txtBox.Text = CStr((2 + 3) < 6)
```

  displays True in the text box.

Boolean Variable

A variable declared with a statement of the form

```
Dim var As Boolean
```

Has Boolean data type. It can assume just the two values True and False.

Example:

```
Dim boolVar As Boolean
boolVar = 2 < 6
txtBox.Text = CStr(boolVar)
```

displays True in the text box.

Syntax Error

The following is NOT a valid way to test whether *n* falls between 2 and 5:

$$2 < n < 5$$

Correction to Syntax Error

To test if *n* falls between 2 and 5 use:

(2 < n) And (n < 5)

A complete relational expression must be on either side of the logical operators And and Or.

Common Error in Boolean Expressions

- A common error is to replace the condition **Not (2 < 3)** with the condition **(2 > 3)**.
- The correct replacement is **(2 >= 3)**

  because >= is the opposite of <, just as <= is the opposite of >.

Two Boolean-Valued Methods

- The expression **`strVar1.EndsWith(strVar2)`**

  is true if the value of the first variable ends with the value of the second variable

- The expression **`strVar1.StartsWith(strVar2)`**

  is true if the value of the first variable begins with the value of the second variable

- **Note:** String literals can be used instead of string variables

Examples

After the following code is executed each text box will contain the word True.

```
Dim firstName As String = "William"
txtBox1.Text = firstName.EndsWith("am")
txtBox2.Text = firstName.StartsWith("Will")
```

A Boolean-Valued Function

- The expression **`IsNumeric(strVar)`**

  is true if the value of *strVar* can be converted to a number with CInt or CDbl.
  Note: The string variable can be replaced with a string literal.

- Examples: **`IsNumeric("123")`** is true

  `IsNumeric("$123")` is true

  `IsNumeric("3 - 2")` is false

4.2 If Blocks

- If Block
- Nested If Blocks
- ElseIf Clauses
- Input Validation with If Blocks

If Block

The following program will take a course of action based on whether a condition is true.

```
If condition Then
   action 1
Else
   action 2
End If
```

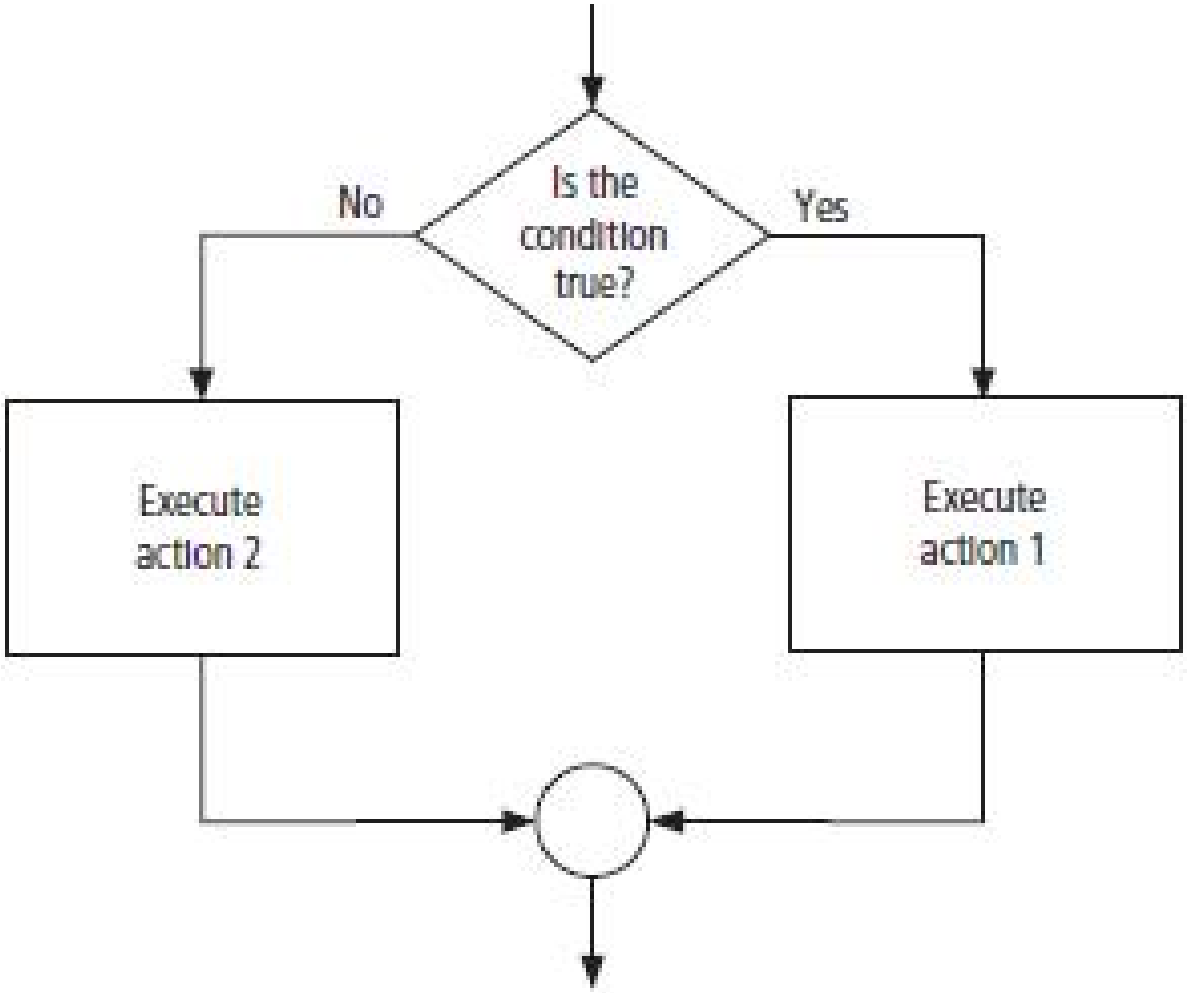
Another example If Block

```
If condition Then
   action 1
End If
Statement 2
Statement 3
```

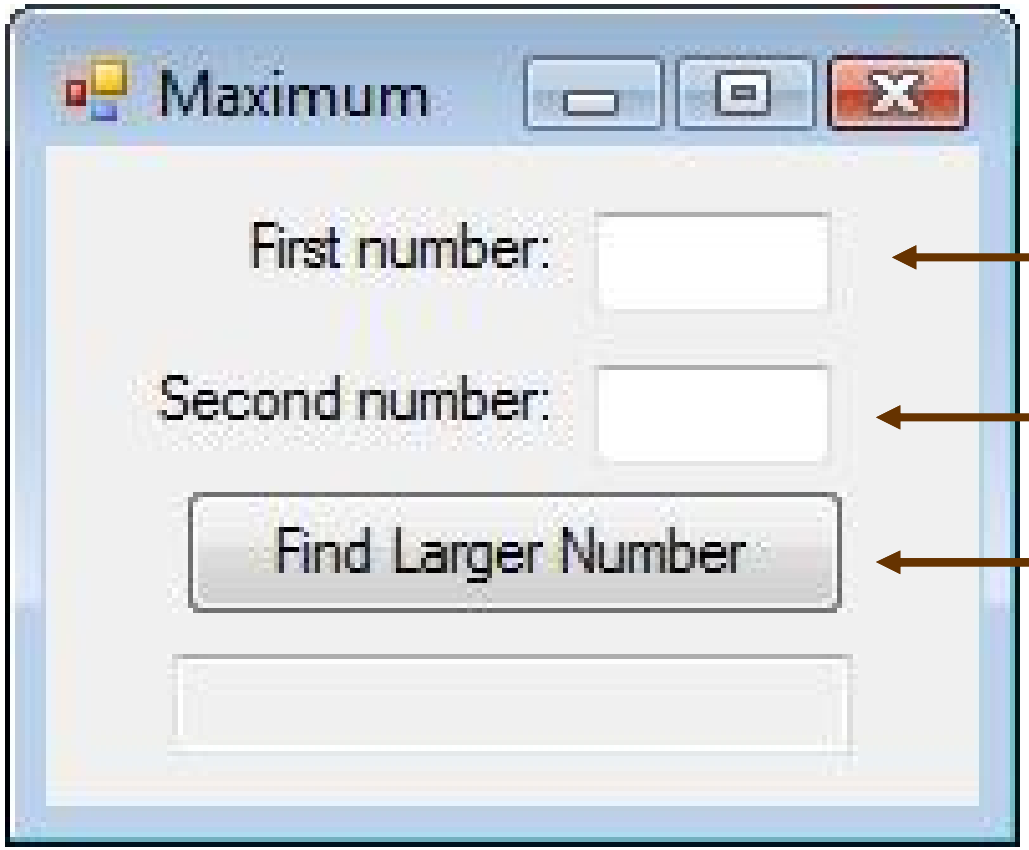
Regardless of whether the condition in the If statement is true or false, these statements will be executed

Pseudocode and Flowchart for an If Block

```
If condition is true Then
  Execute action 1
Else
  Execute action 2
End If
```



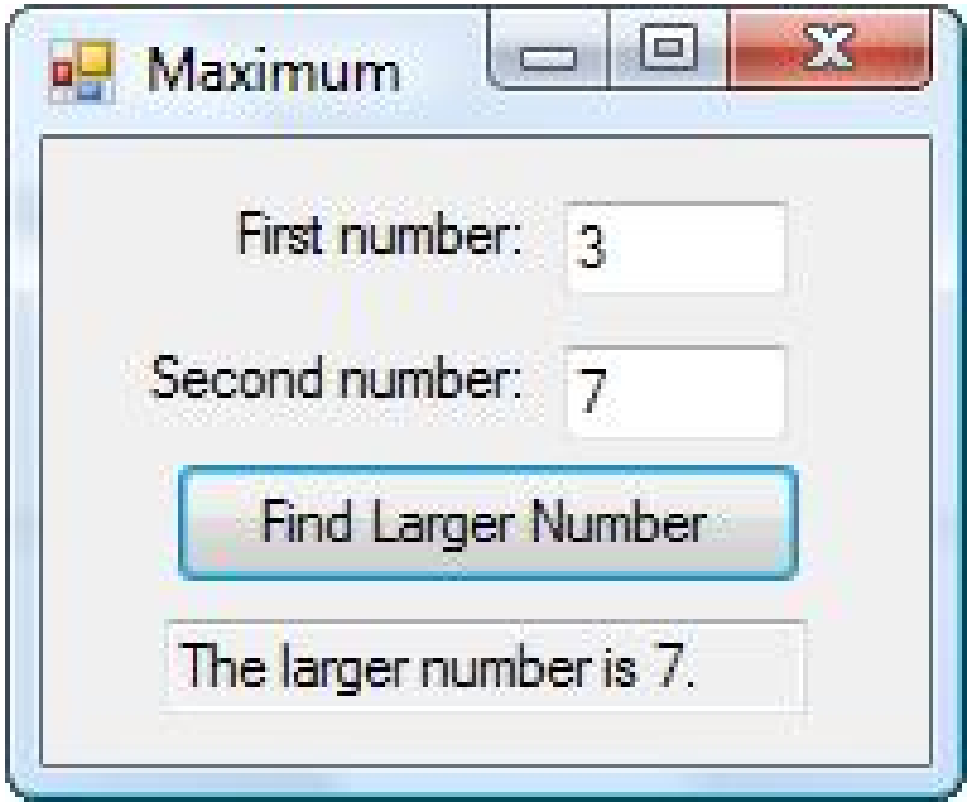
Example 1: Form



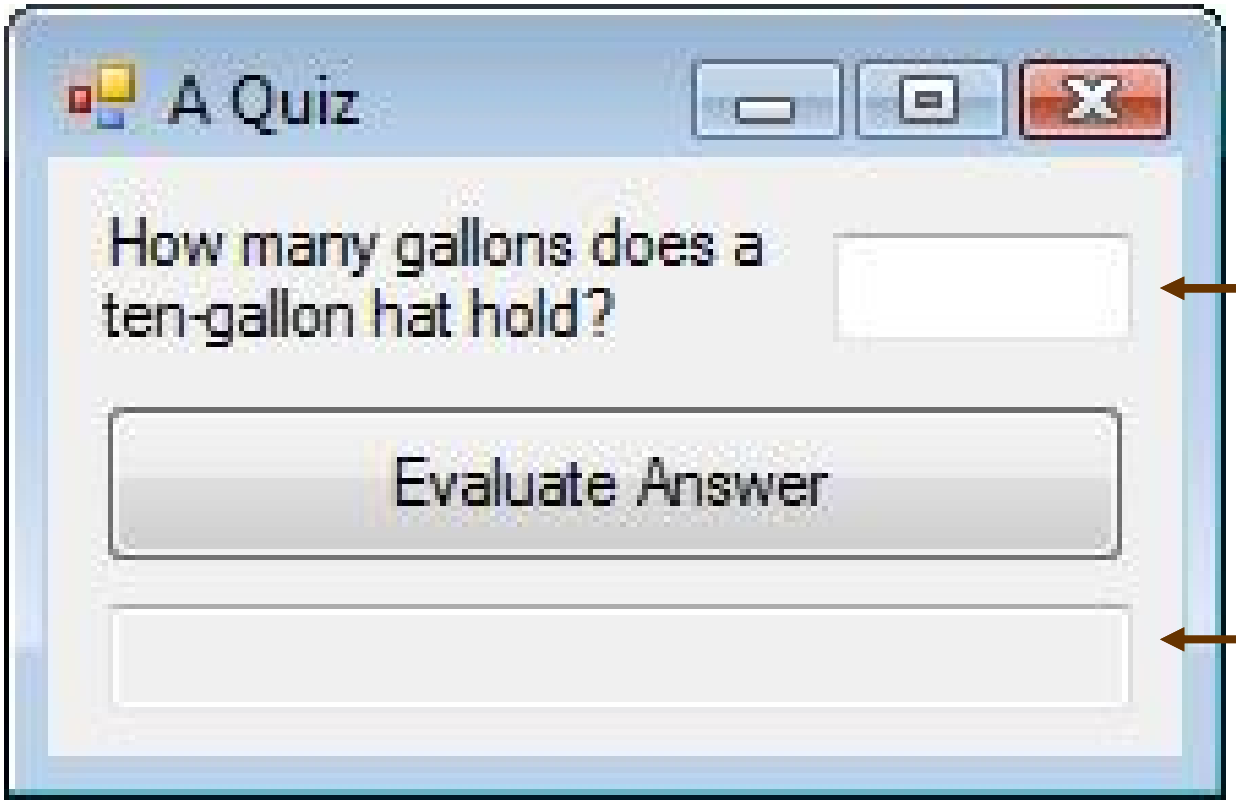
Example 1: Code

```
Private Sub btnFindLarger_Click(...) _
                        Handles btnFindLarger.Click
  Dim num1, num2, largerNum As Double
  num1 = CDbl(txtFirstNum.Text)
  num2 = CDbl(txtSecondNum.Text)
  If num1 > num2 Then
    largerNum = num1
  Else
    largerNum = num2
  End If
  txtResult.Text = "Larger number: " & largerNum
End Sub
```

Example 1: Output



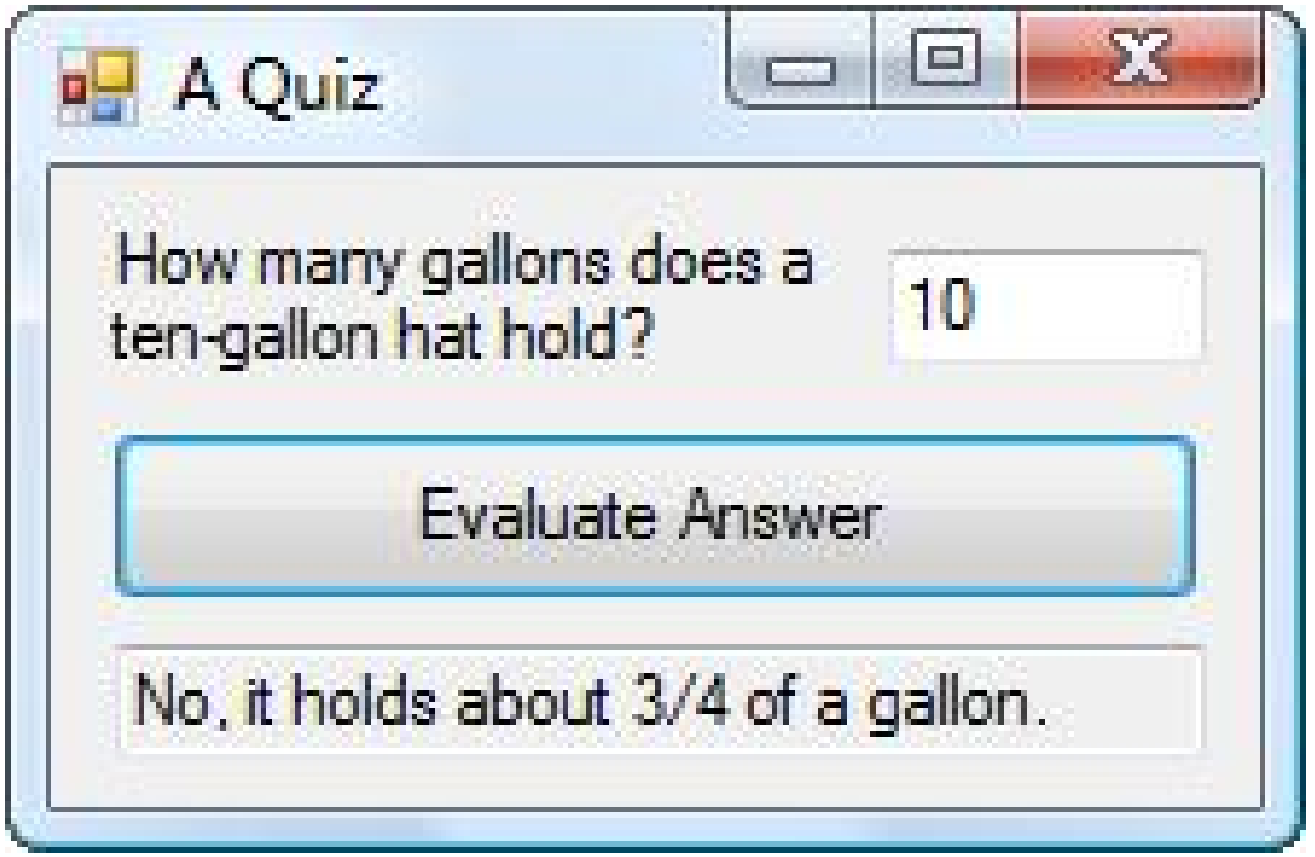
Example 2: Form



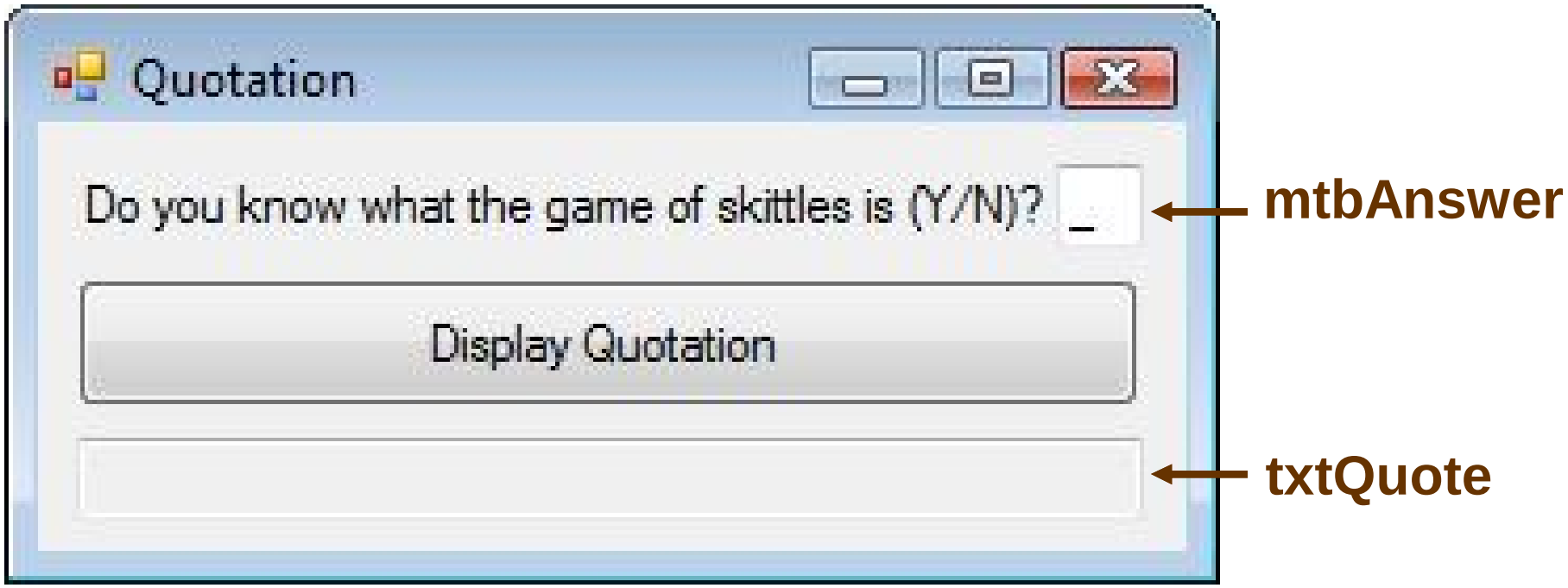
Example 2: Code

```
Private Sub btnEvaluate_Click(...) _
                        Handles btnEvaluate.Click
  Dim answer As Double
  answer = CDbl(txtAnswer.Text)
  If (answer >= 0.5) And (answer <= 1) Then
    txtSolution.Text = "Good, "
  Else
    txtSolution.Text = "No, "
  End If
  txtSolution.Text &= "it holds about 3/4 gals."
End Sub
```

Example 2: Output



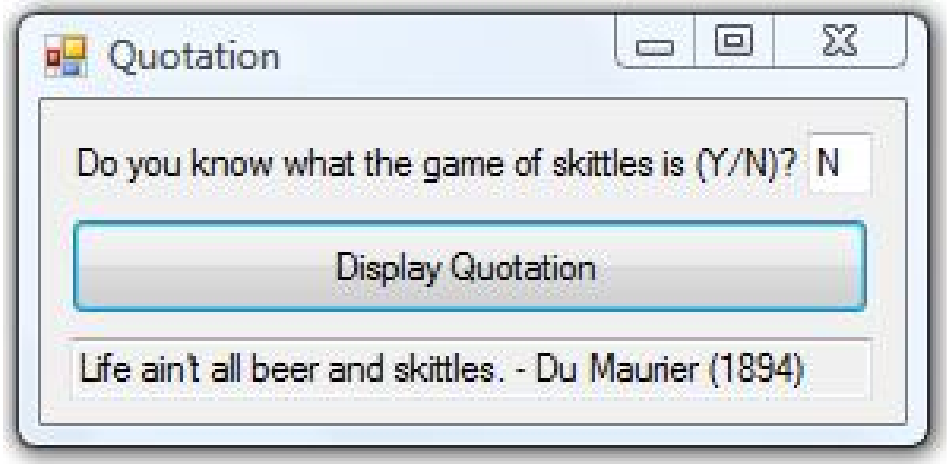
Example 3: Form

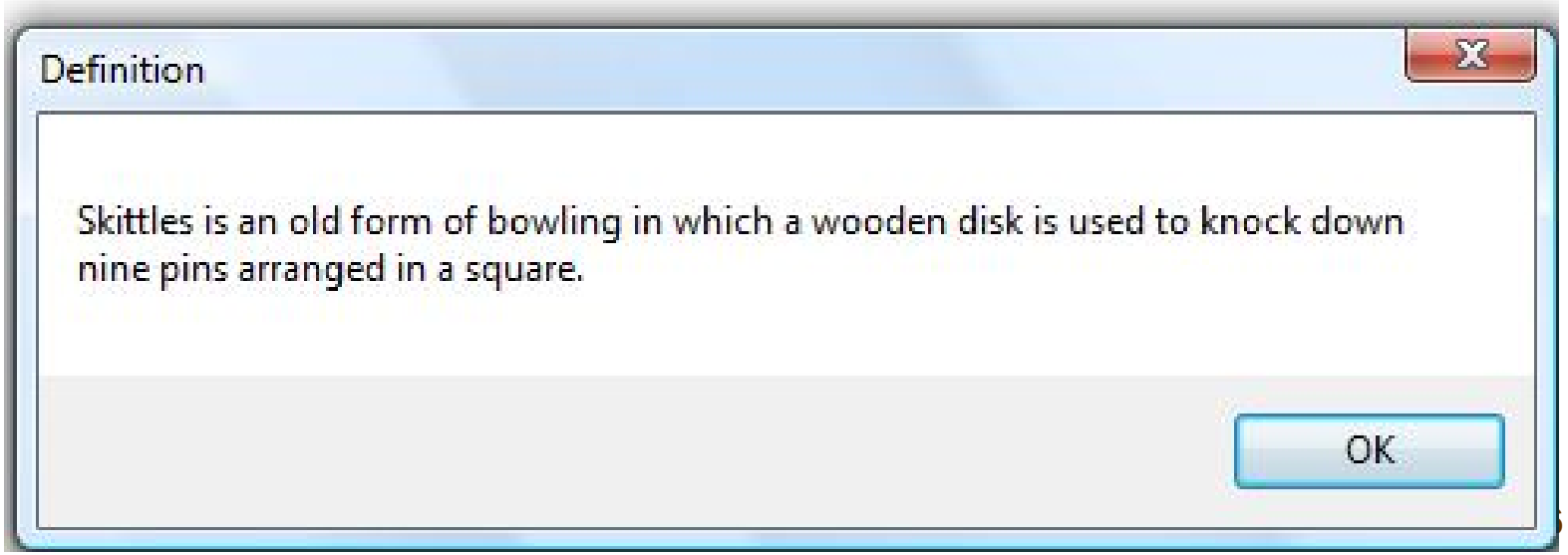


Example 3

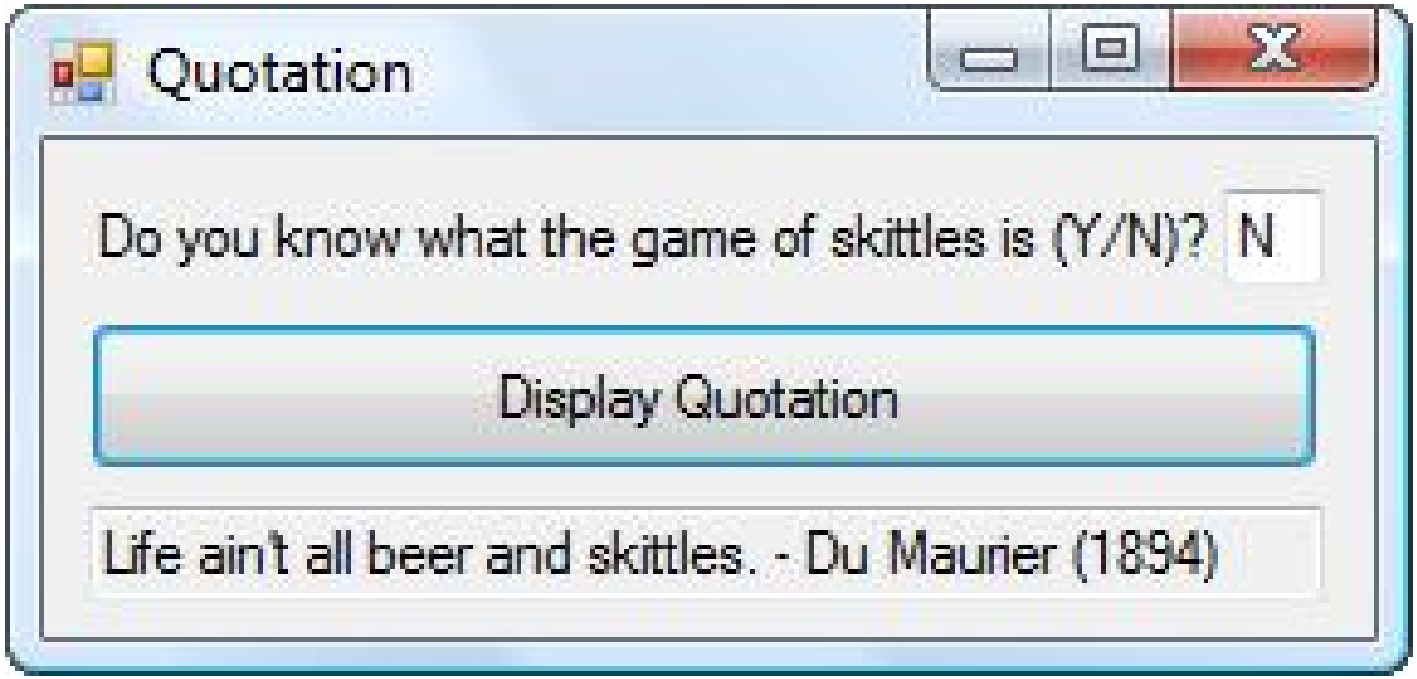
```
Private Sub btnDisplay_Click(...) _
                            Handles btnDisplay.Click
  Dim msg As String
  msg = "Skittles is an old form of bowling " &
   "in which a wooden disk is used to knock " &
   "down nine pins arranged in a square. "
  If txtAnswer.Text.ToUpper = "N" Then
    MessageBox.Show(msg, "")
  End If
  txtQuote.Text = "Life ain't all beer " &
            and skittles. – Du Maurier (1894)."
End Sub
```

Example 3: Output





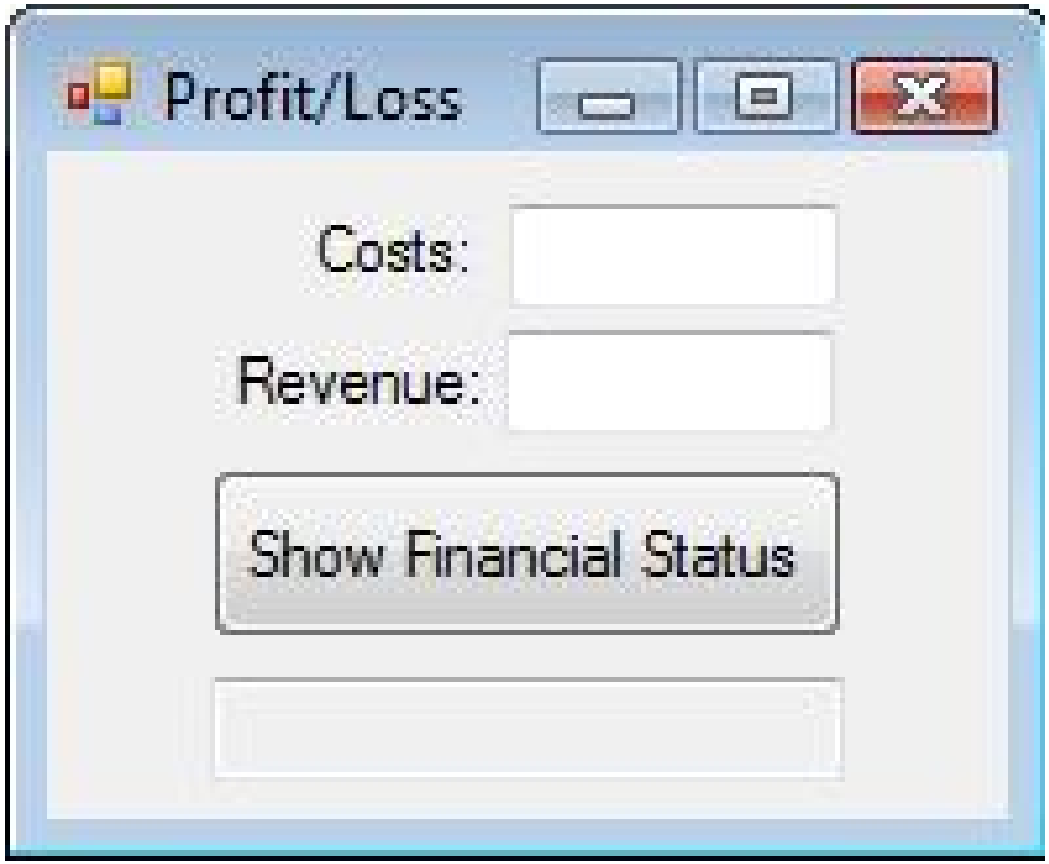
Example 3: Output (continued)



Nested If Blocks

When one If block is contained inside another If block, the structure is referred to as **nested If blocks**.

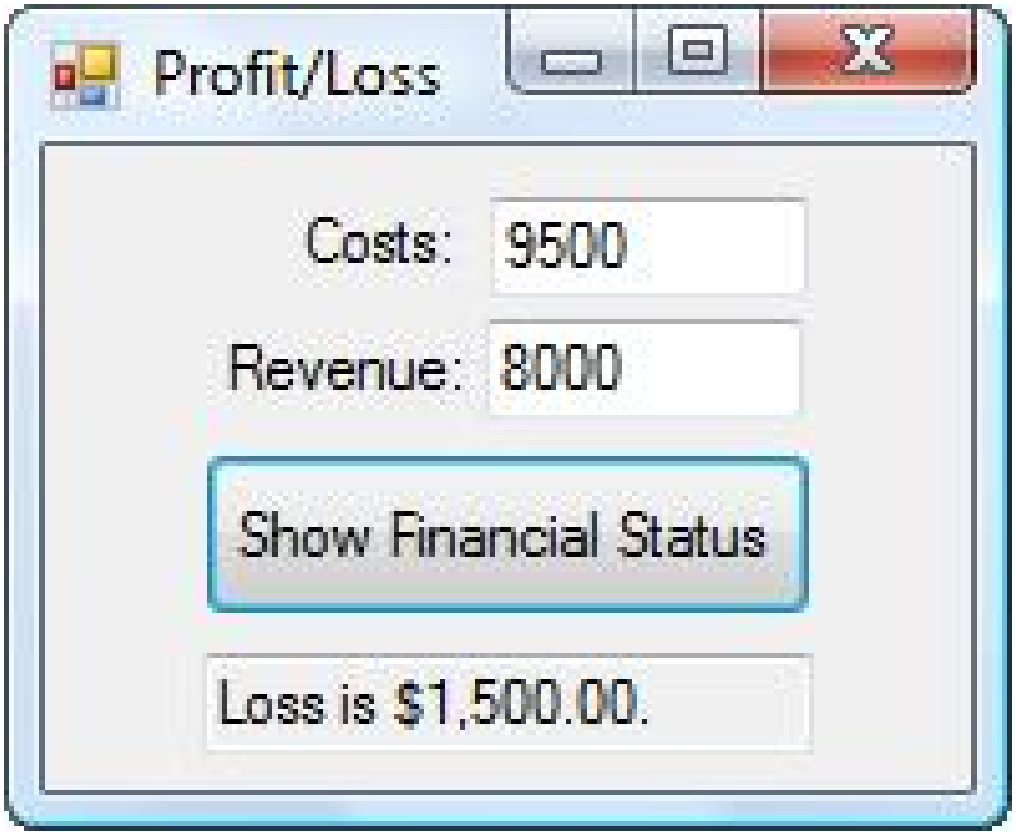
Example 5: Form



Example 5: Partial Code

```
If costs = revenue Then
  txtResult.Text = "Break even"
Else
  If costs < revenue Then
    profit = revenue - costs
    txtResult.Text = "Profit is " &
                     FormatCurrency(profit) & "."
  Else
    loss = costs - revenue
    txtResult.Text = "Loss is " &
                     FormatCurrency(loss) & "."
  End If
End If
```

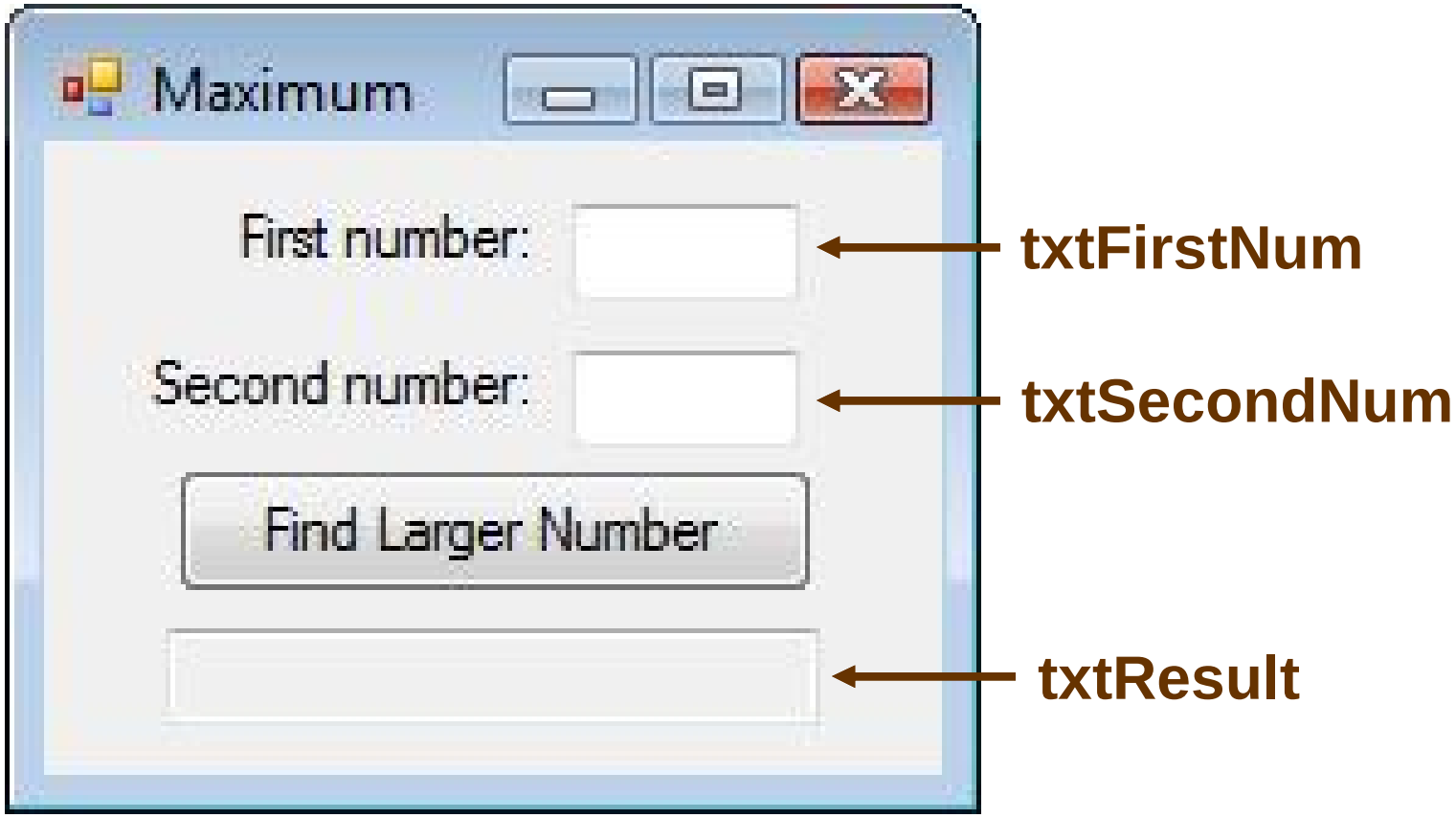
Example 5: Output



ElseIf Clause

```
If condition 1 Then
  action 1
ElseIf condition 2 Then
  action 2
ElseIf condition 3 Then
  action 3
Else
  action 4
End If
```

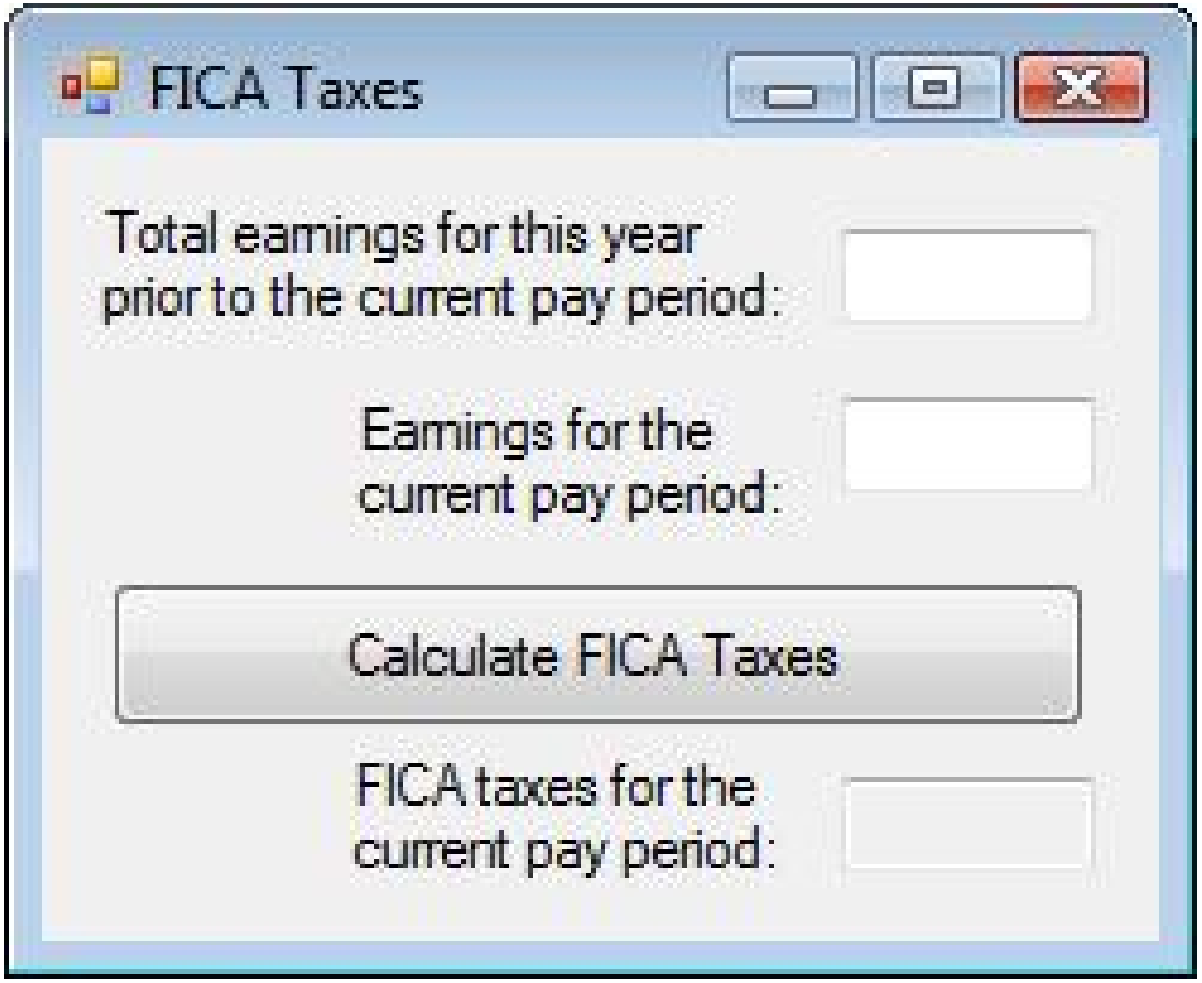
Example 6: Form



Example 6: Code

```
Private Sub btnFindLarger_Click(...) _
                        Handles btnFindLarger.Click
  Dim num1, num2 As Double
  num1 = CDbl(txtFirstNum.Text)
  num2 = CDbl(txtSecondNum.Text)
  If (num1 > num2) Then
    txtResult.Text = "Larger number is " & num1
  ElseIf (num2 > num1) Then
    txtResult.Text = "Larger number is " & num2
  Else
    txtResult.Text = "The two are equal."
  End If
End Sub
```

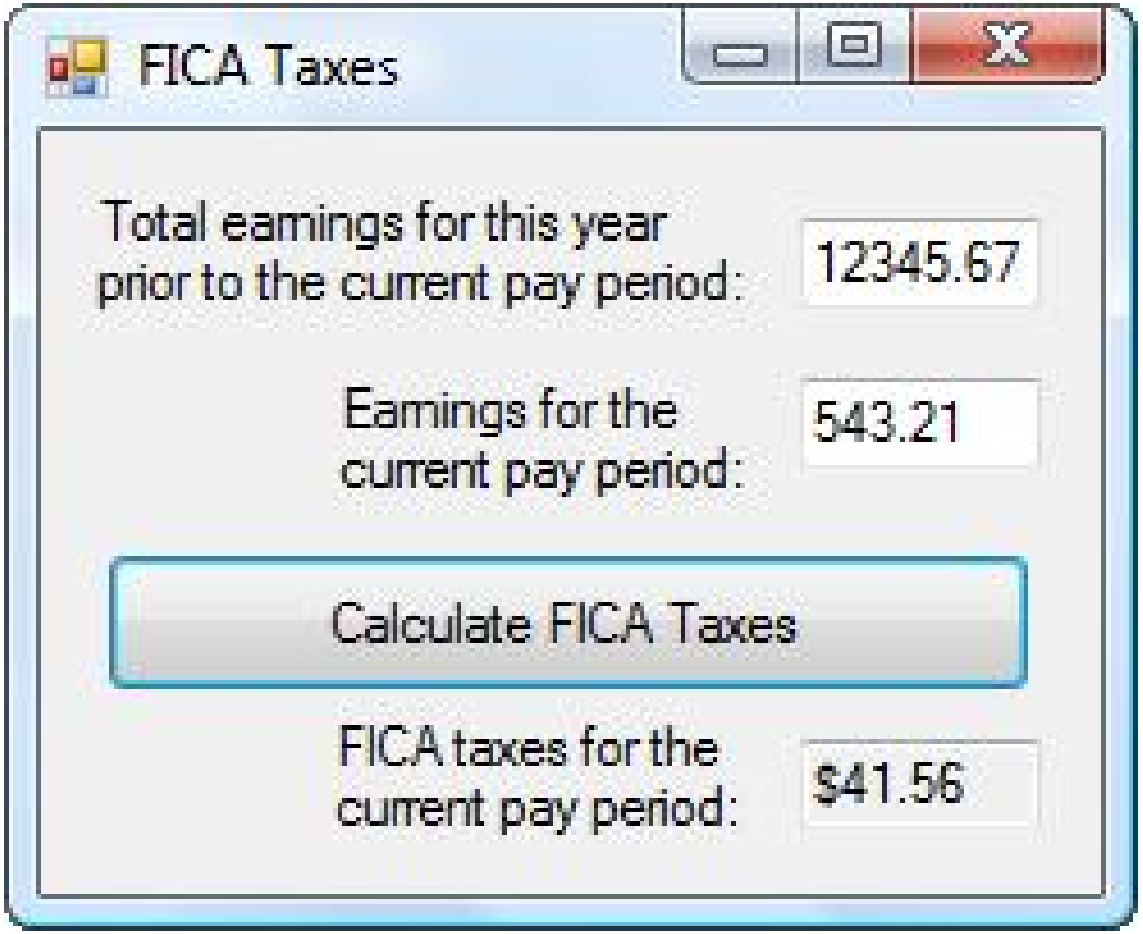
Example 7: Form



Example 7: Partial Code

```
Const WAGE_BASE As Double = 106800
Dim ytdEarnings, curEarnings As Double
Dim socSecBenTax, medicareTax,
    ficaTaxes As Double
ytdEarnings = CDbl(txtToDate.Text)
curEarnings = CDbl(txtCurrent.Text)
If (ytdEarnings + curEarnings) <= WAGE_BASE Then
 socSecBenTax = 0.062 * curEarnings
ElseIf ytdEarnings < WAGE_BASE Then
 socSecBenTax = 0.062 * (WAGE_BASE -
  ytdEarnings)
End If
medicareTax = 0.0145 * curEarnings
ficaTaxes = socSecBenTax + medicareTax
```

Example 7: Output



Input Validation

The statement

```
If (IsNumeric(txtBox.Text) = True) Then
```

is commonly used to validate that input is numeric. It can be condensed to

```
If IsNumeric(txtBox.Text) Then
```

Simplified Nested If Statement

Care should be taken to make If blocks easy to understand.

```
If cond1 Then
  Then
  If cond2 Then
    action
  End If
End If
```

Confusing

```
If cond1 And cond2
  action
End If
```

Clear

Comment

Some programs call for selecting among many possibilities. Although such tasks can be accomplished with complicated nested If blocks, the Select Case block (discussed in Section 4.3) is often a better alternative.

4.3 Select Case Block

- A decision-making structure that simplifies choosing among several actions.
- Avoids complex nested If constructs.
- If blocks make decisions based on the truth value of a condition. Select Case choices are determined by the value of an expression called a **selector**.

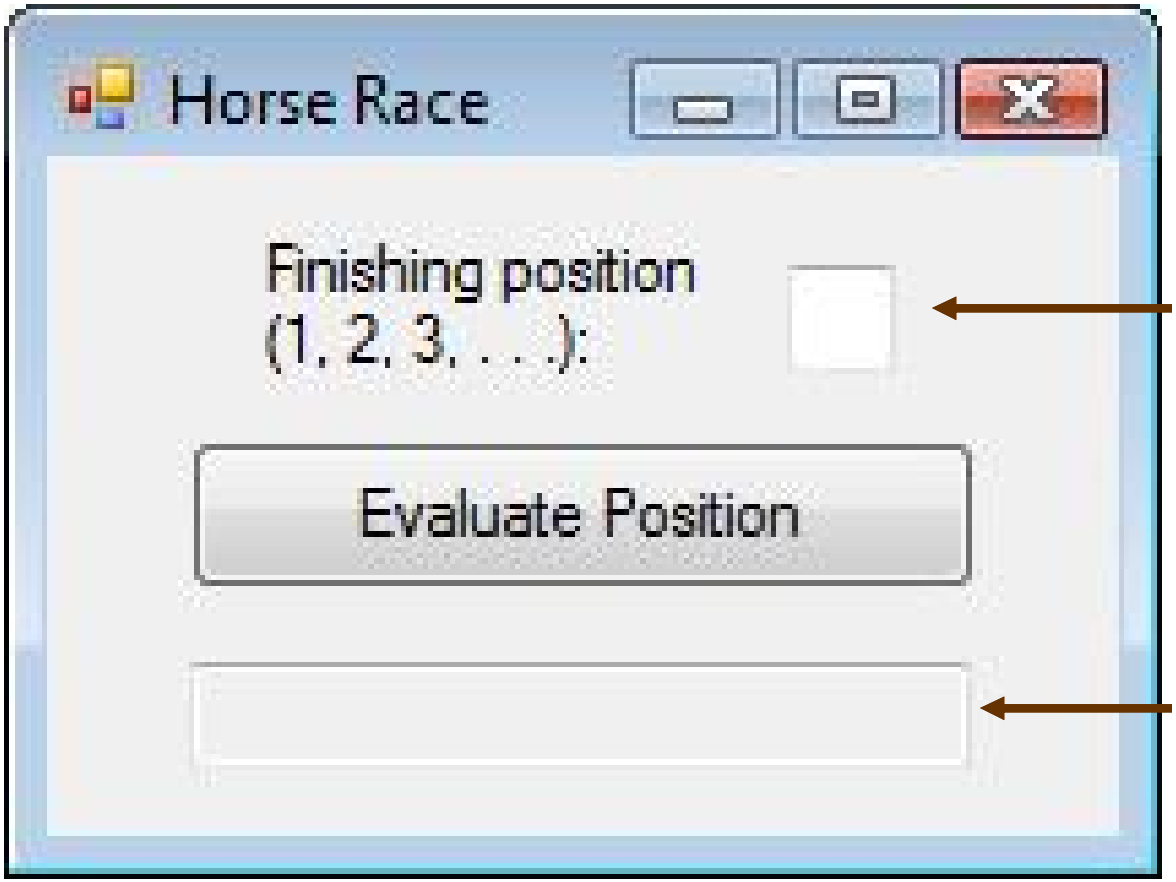
Select Case Terminology

Each of the possible actions is preceded by a clause of the form

```
Case valueList
```

where *valueList* itemizes the values of the **selector** for which the action should be taken.

Example 1: Form



txtPosition

txtOutcome

Example 1: Code

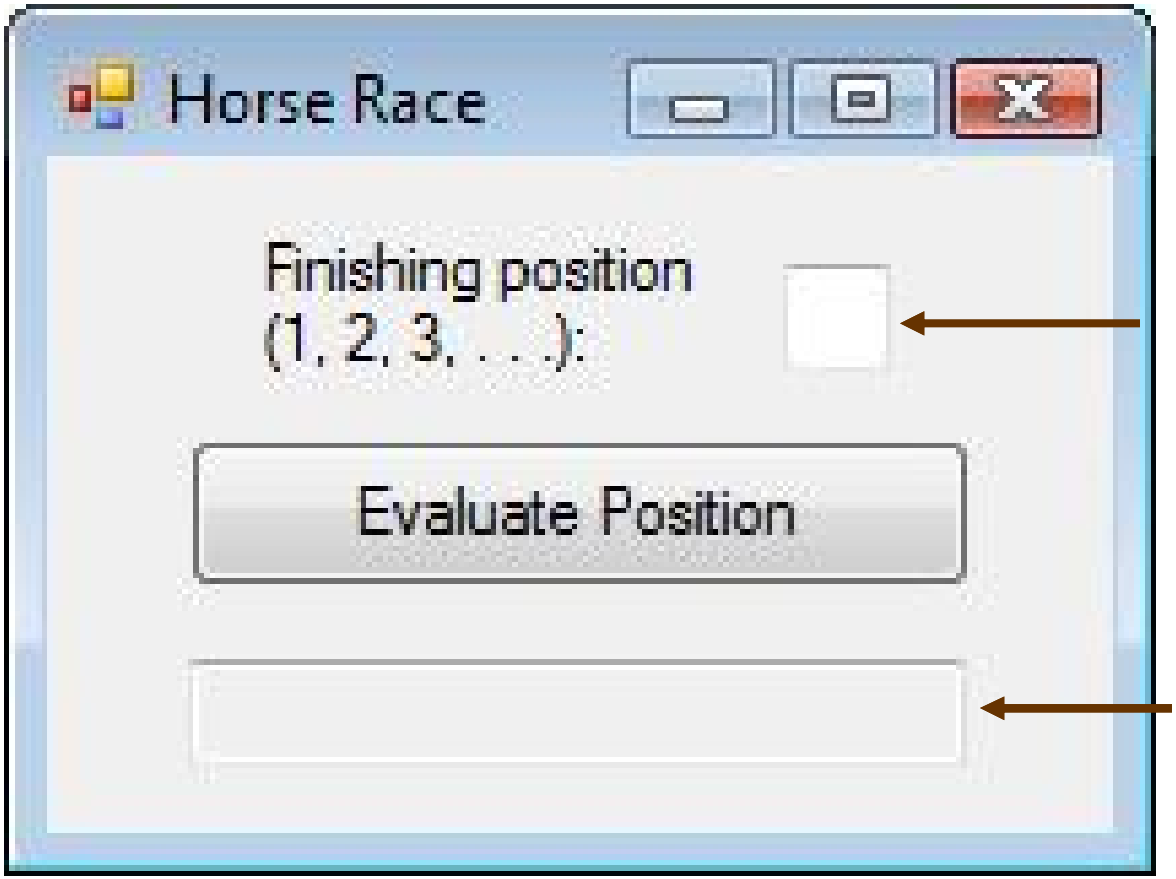
```
Private Sub btnEvaluate_Click(...) _
                          Handles btnEvaluate.Click
  Dim position As Integer = CInt(txtPosition.Text)
  Select Case position
    Case 1
      txtOutcome.Text = "Win"
    Case 2
      txtOutcome.Text = "Place"
    Case 3
      txtOutcome.Text = "Show"
    Case 4, 5
      txtOutcome.Text = "Almost in the money."
    Case Else
      txtBox.Text = "Out of the money."
  End Select
End Sub
```

Selector

Value Lists

Example 1: Output

Example 2: Form



Example 2: Code

```
Private Sub btnEvaluate_Click(...) _
                        Handles btnEvaluate.Click
  Dim position As Integer =
                            CInt(txtPosition.Text)
  Select Case position
    Case 1 To 3
      txtOutcome.Text =
                  "In the money. Congratulations"
    Case Is >= 4
      txtOutcome.Text = "Not in the money."
  End Select
End Sub
```

Example 2: Output

Select Case Syntax

The general form of the Select Case block is

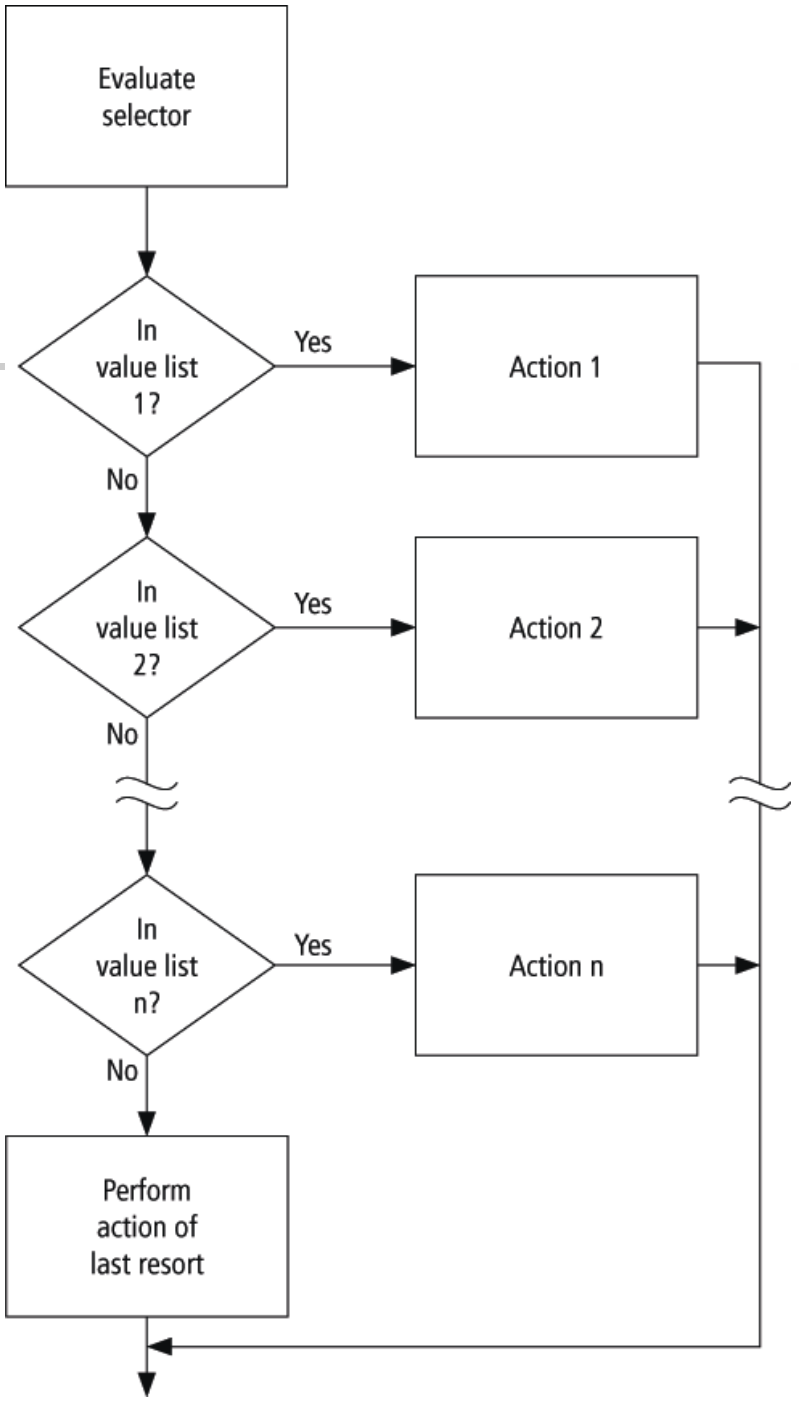
```
Select Case selector
  Case valueList1
    action1
  Case valueList2
    action2
  Case Else
    action of last resort
End Select
```

Rules for Select Case

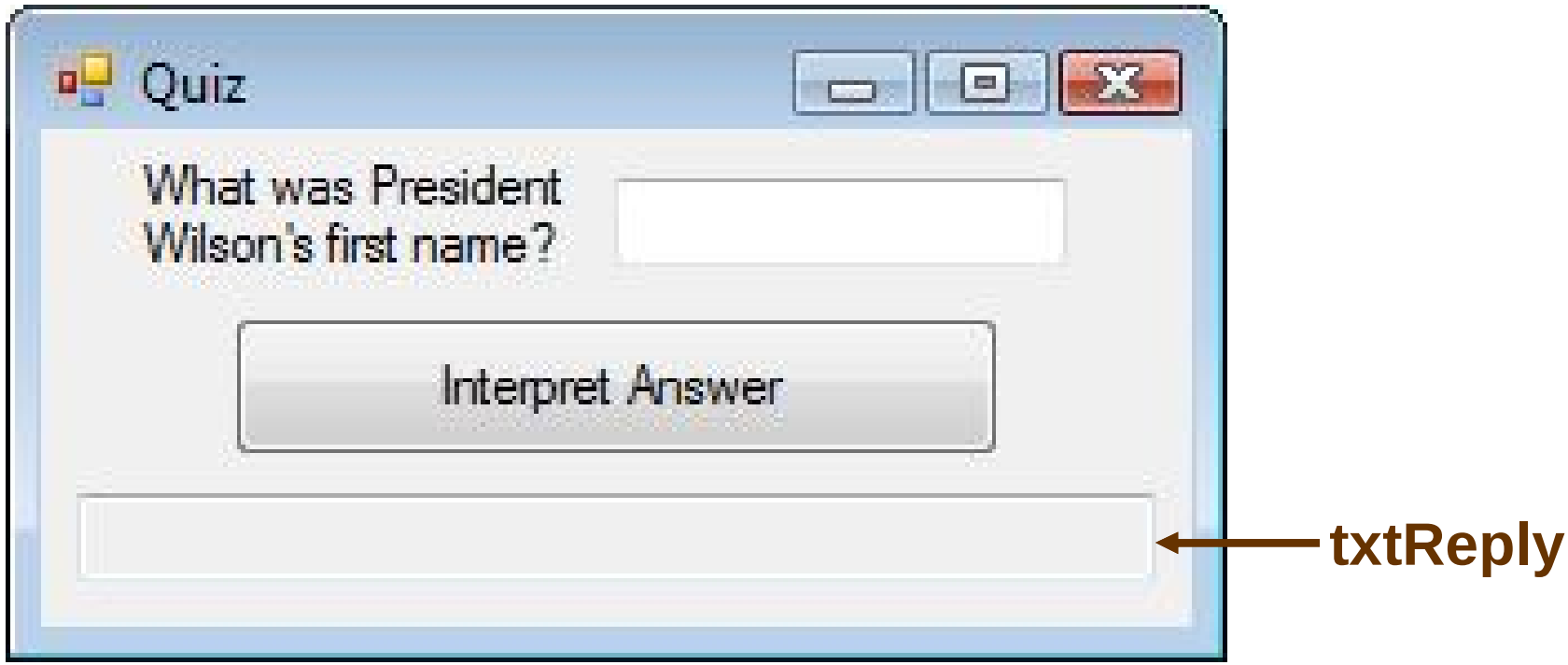
Each value list contains one or more of the following types of items separated by commas.

1. a literal
2. a variable
3. an expression
4. an inequality sign preceded by Is and followed by a literal, variable, or expression
5. a range given in the form *a* To *b*, where *a* and *b* are literals, variables, or expressions.

Flowchart for Select Case



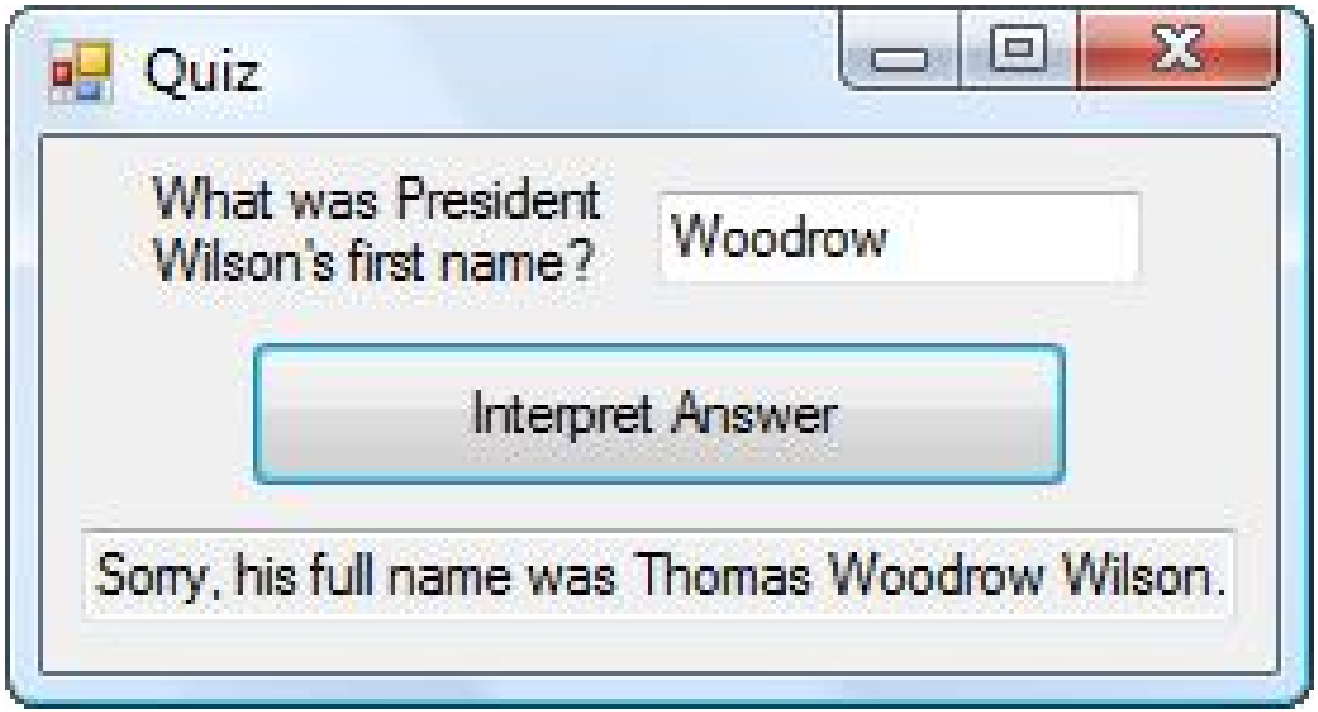
Example 4: Form



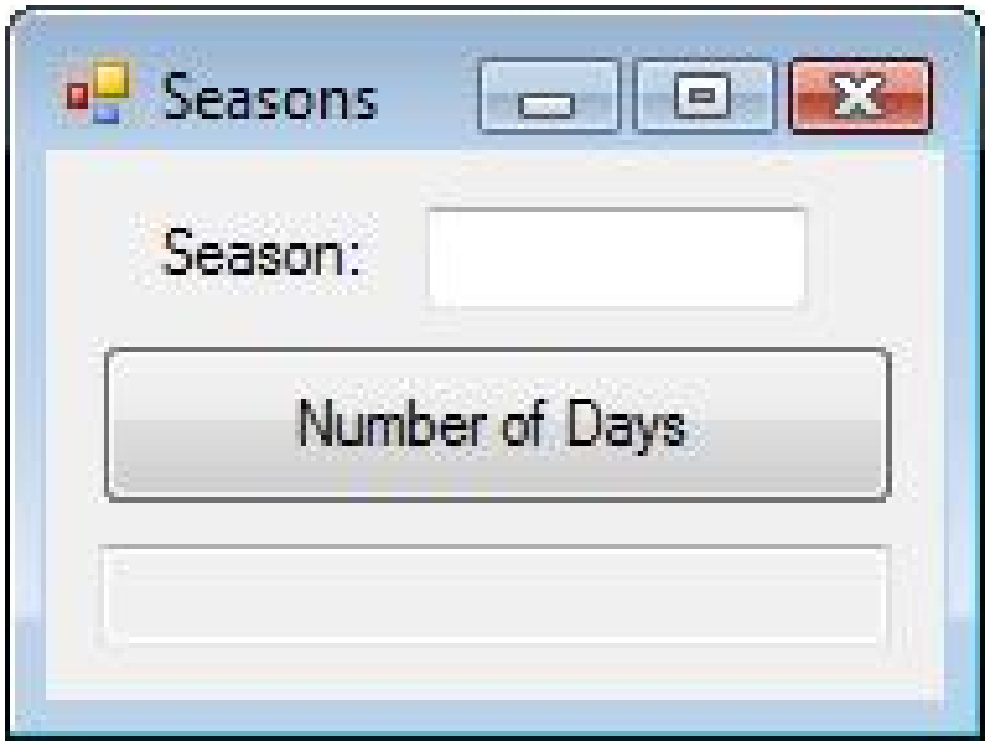
Example 4: Partial Code

```
Select Case firstName
  Case "THOMAS"
    txtReply.Text = "Correct."
  Case "WOODROW"
    txtReply.Text = "Sorry, his name" &
          "  was Thomas Woodrow Wilson."
  Case "PRESIDENT"
    txtReply.Text = "Are you for real?"
  Case Else
    txtReply.Text = "Nice try."
End Select
```

Example 4: Output



Example: Form



Example: Partial Code

```
Dim numDays As Integer
Dim season = txtSeason.Text
Select Case season.ToUpper
  Case "WINTER"
    numDays = 87
  Case "SPRING"
    numDays = 92
  Case "SUMMER", "AUTUMN", "FALL"
    numDays = 93
End Select
```

Example: Output

Comments

- In a Case clause of the form *Case b To c*, the value of *b* should be less than or equal to the value of *c*.
- The word Is should precede an inequality sign in a value list.
- If the word Is is accidentally omitted where required, the editor will automatically insert it when checking the line.

Data Type Comment

- The items in the value list must evaluate to a literal of the same data type as the selector.
- For instance, if the selector evaluated to a string value, as in

```
Dim firstName As String = txtBox.Text
Select Case firstName
```

then the clause

```
Case firstName.Length
```

would be meaningless.

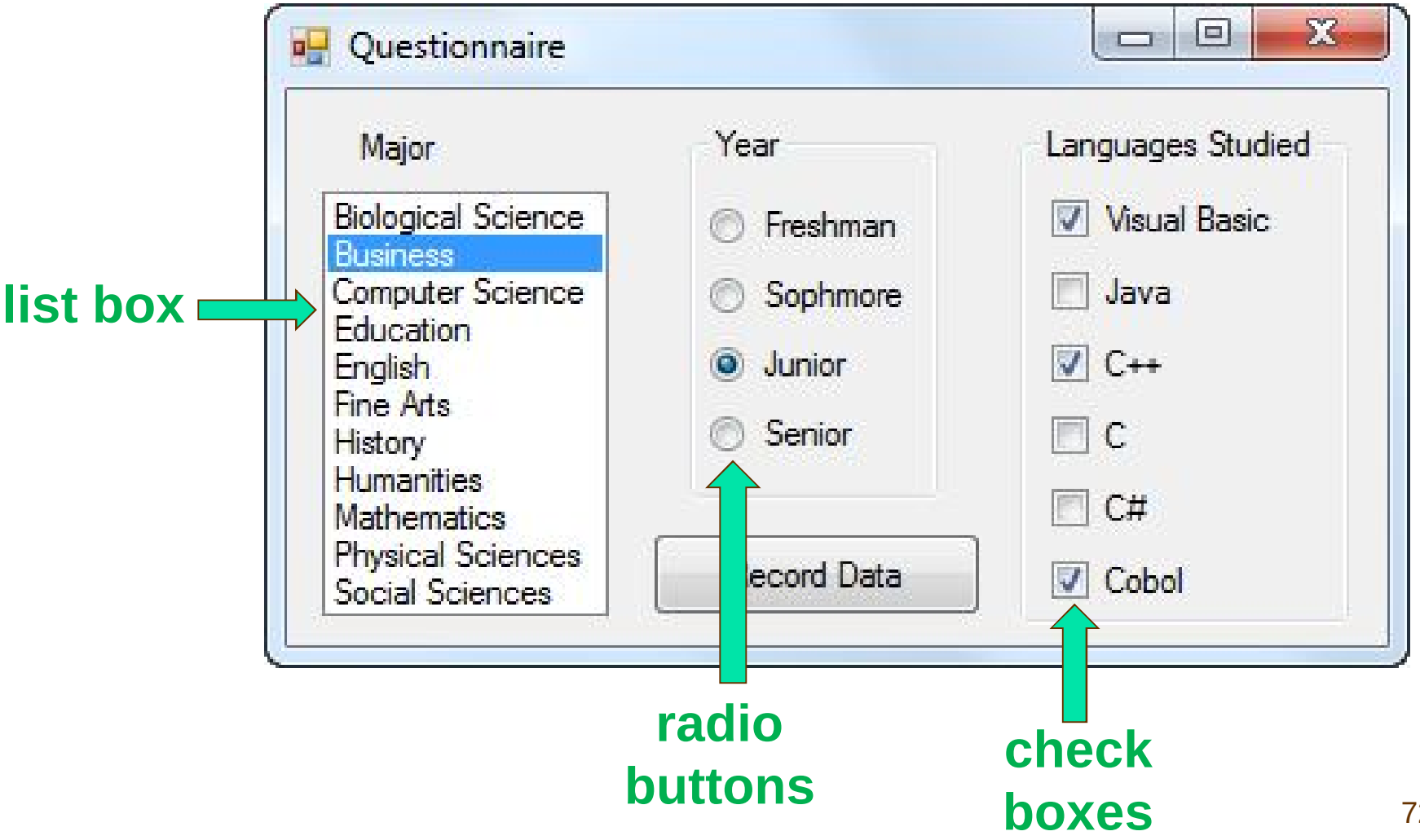
Block-Level Scope

- A variable declared inside an If or Select Case block has **block-level scope**.
- The variable cannot be referred to outside of the block.

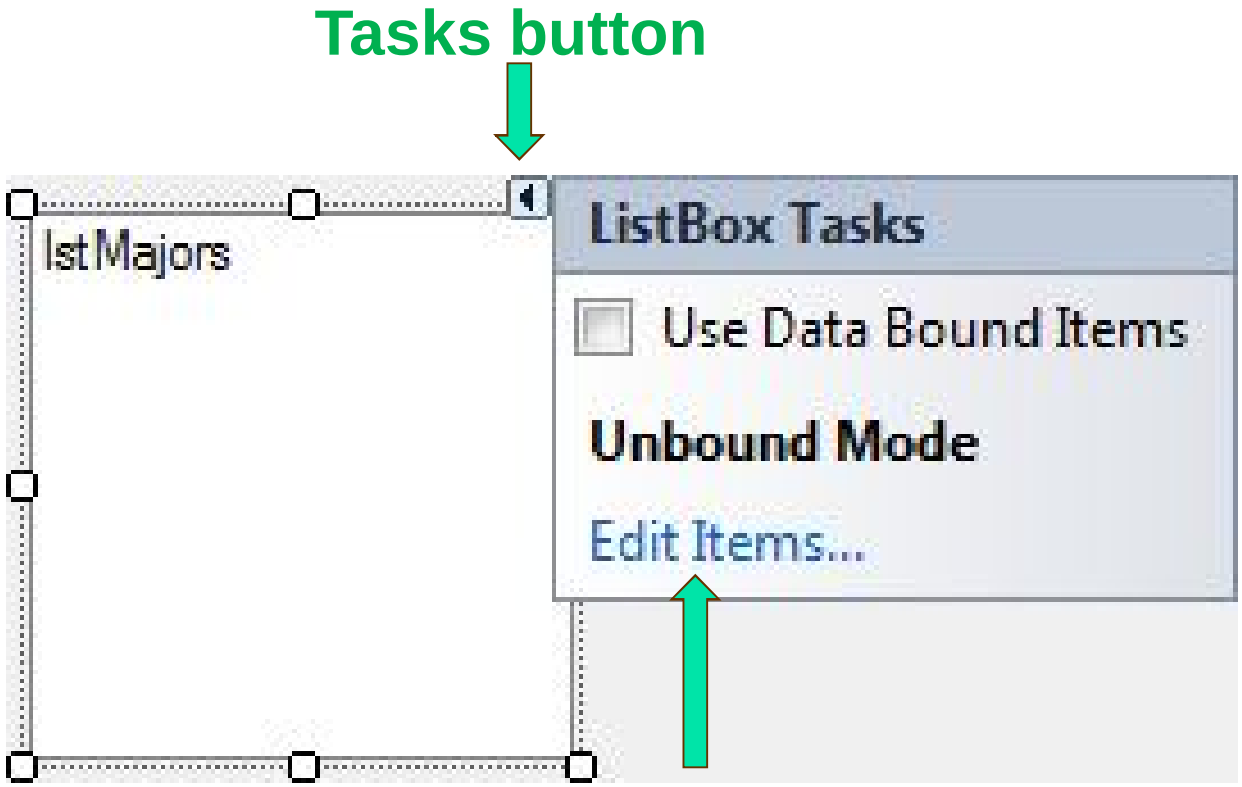
4.4 Input via User Selection

- Using a List Box for Input
- Using Radio Buttons for Input
- Using Check Boxes for Input
- Events Raised by Selections

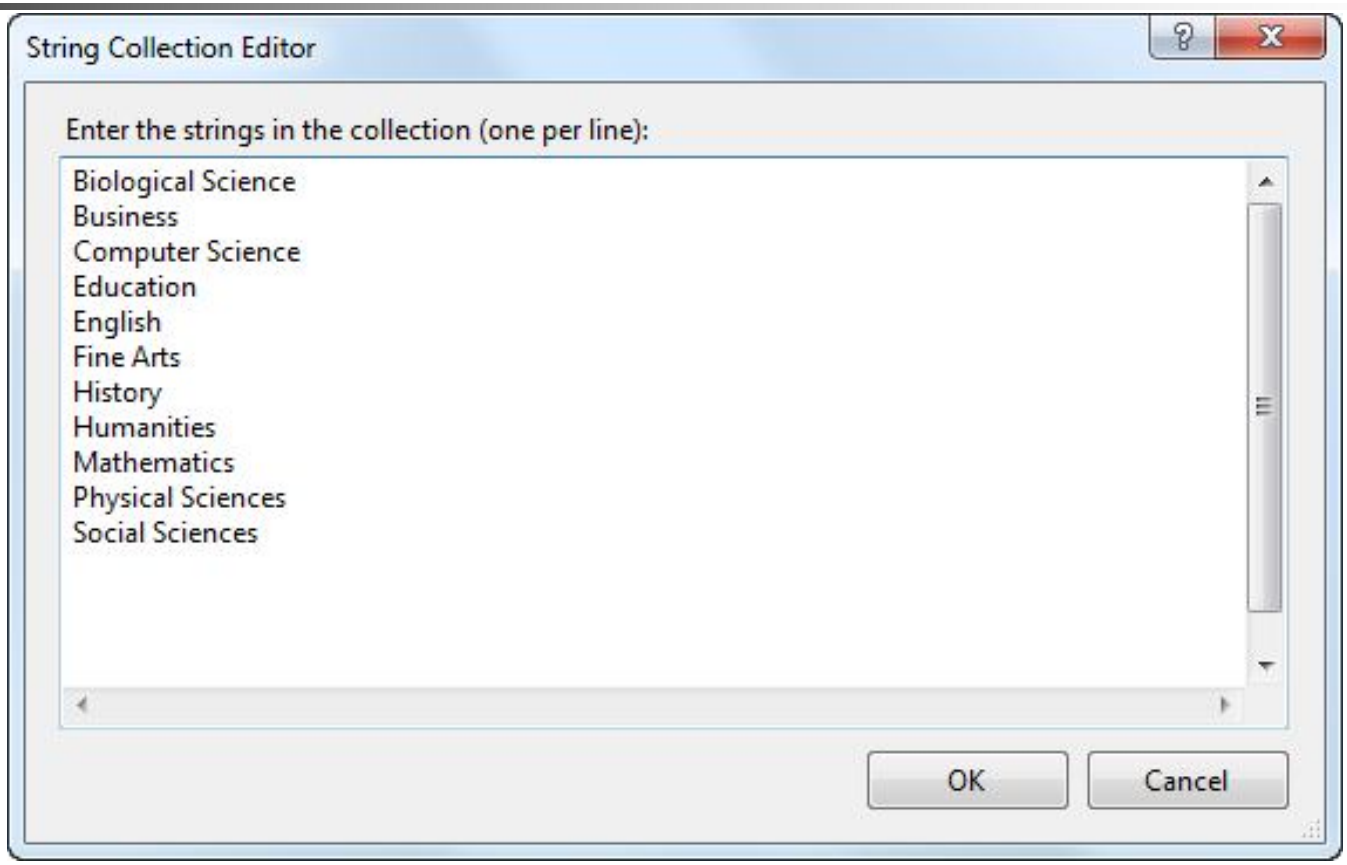
The Three Types of Controls Used for Selection



Fill a List Box at Design Time via its String Collection Editor

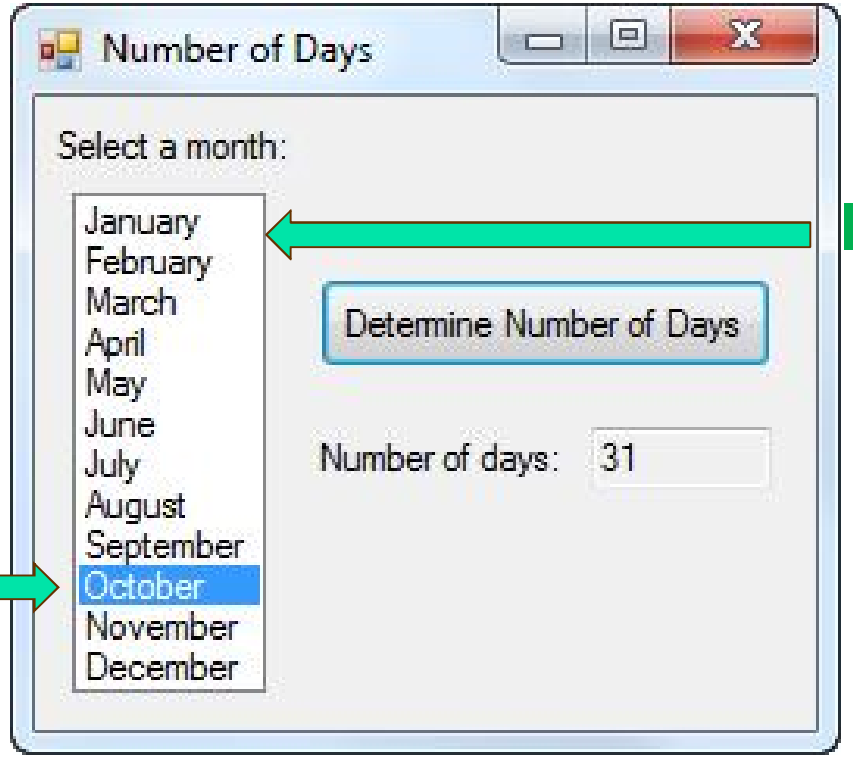


String Collection Editor



Fill by direct typing or by copying and pasting from a text editor or a spreadsheet.

List Box at Run Time



The value of lstMonths.Text is the string consisting of the selected item.

Example 1: Code for btnDetermine.Click

```
Dim daysInMonth As String
Select Case lstMonths.Text
  Case "September", "April", "June", "November"
    daysInMonth = "30"
  Case "February"
    daysInMonth = "28 or 29"
  Case Else
    daysInMonth = "31"
End Select
txtDays.Text = daysInMonth
```

The Group Box Control

- Group boxes are passive objects used to group other objects together.
- When you drag a group box, the attached controls follow as a unit.
- To attach controls to a group box, create the group box and then place the controls into the group box.

Group Box Example

Three attached controls:
Button1
Button2
Button3

Radio Button Properties

- To determine if the button is on or off

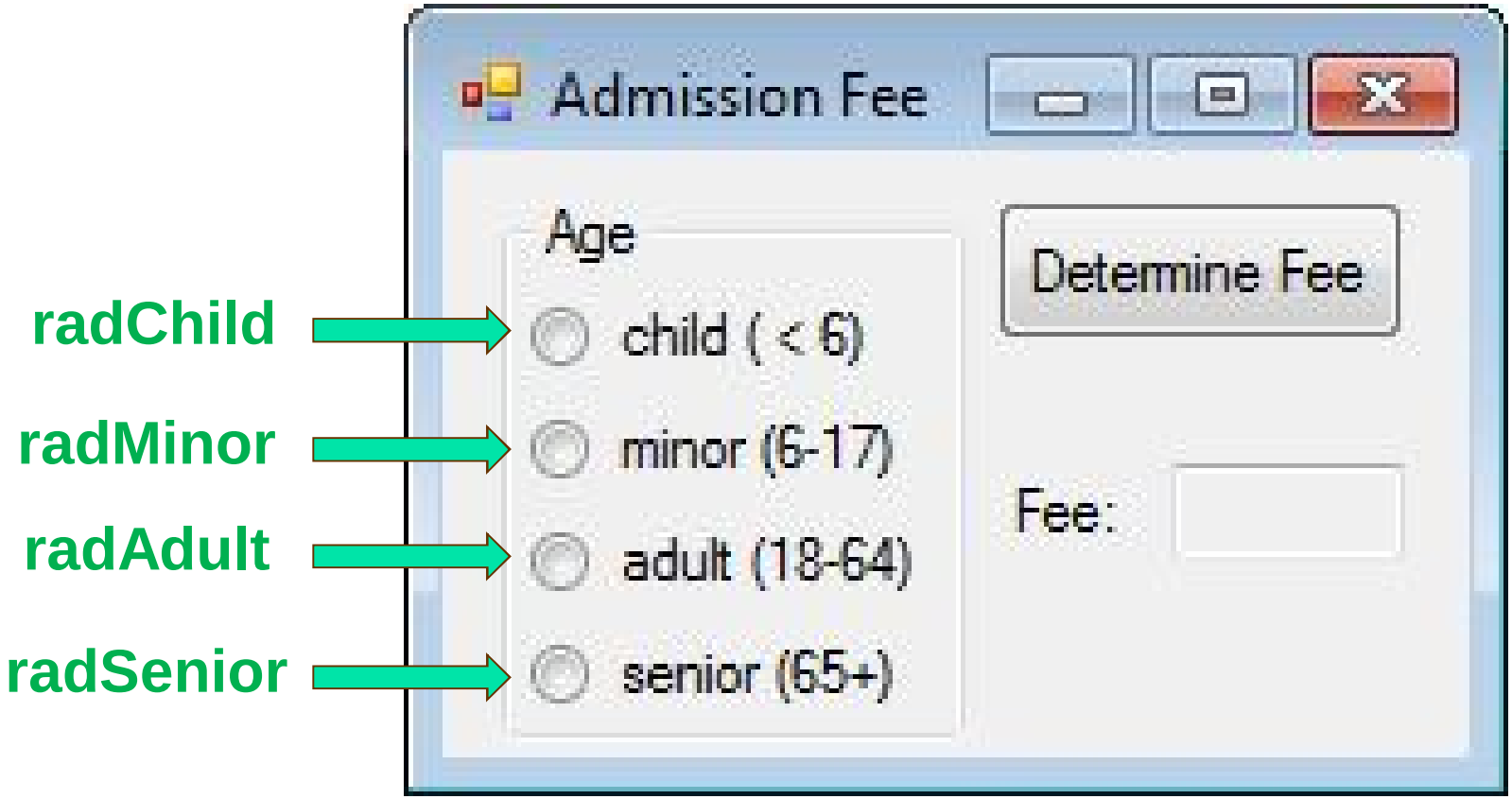
```
radButton.Checked
```

has value True if button is on.

- To turn a radio button on

```
radButton.Checked = True
```

Example 3: Form



Example 3: Code for Button

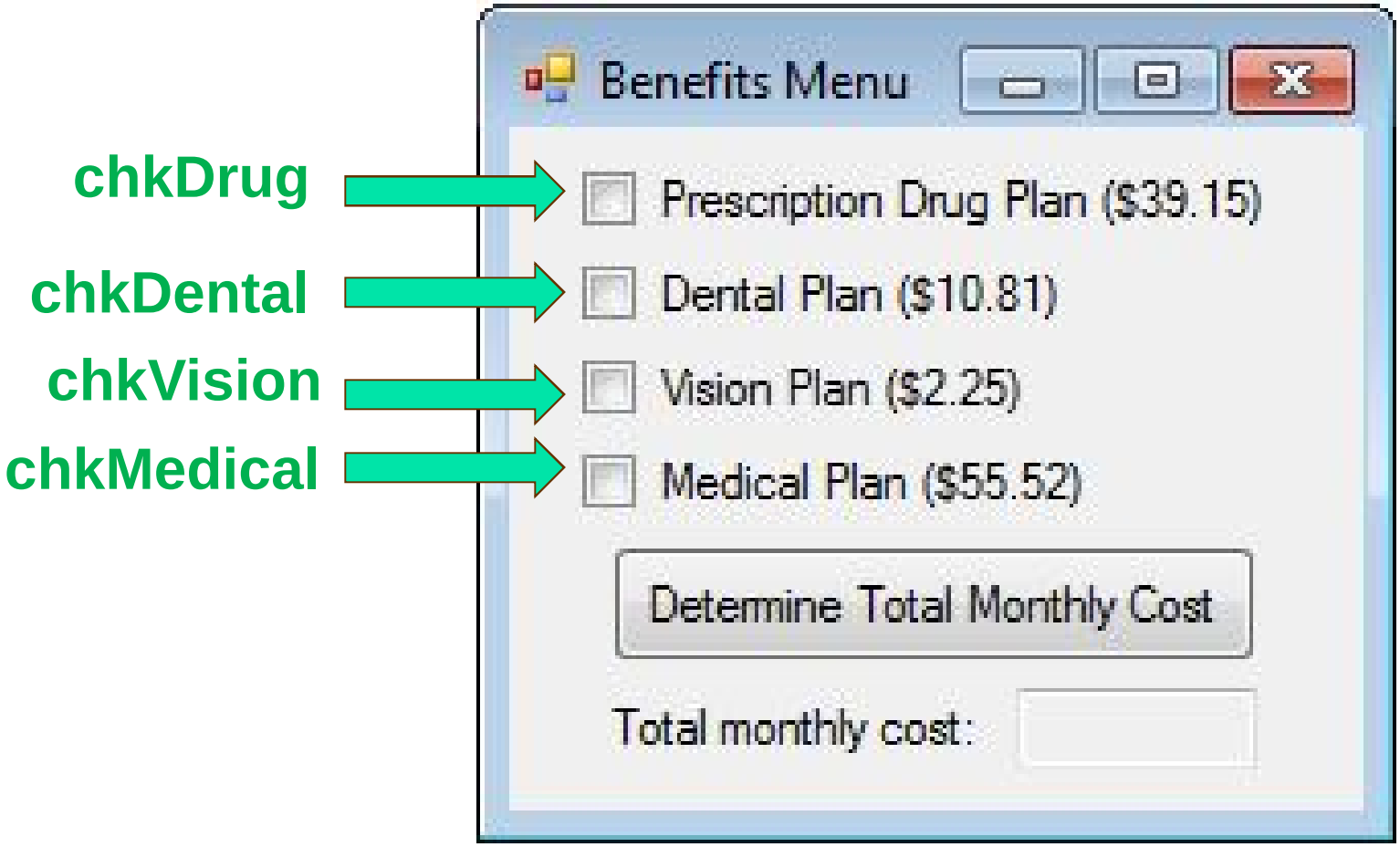
```
If radChild.Checked Then
  txtFee.Text = FormatCurrency(0)
ElseIf radMinor.Checked Then
  txtFee.Text = FormatCurrency(5)
ElseIf radAdult.Checked Then
  txtFee.Text = FormatCurrency(10)
ElseIf radSenior.Checked Then
  txtFee.Text = FormatCurrency(7.5)
Else
  MessageBox.Show("Must make a selection.")
End If
```

Example 3: Output

The Check Box Control

- Consists of a small square and a caption
- Presents the user with a Yes/No choice
- During run time, clicking on the check box toggles the appearance of a check mark.
- Checked property has value True when check box is checked and False when not
- CheckedChanged event is raised when the user clicks on the check box

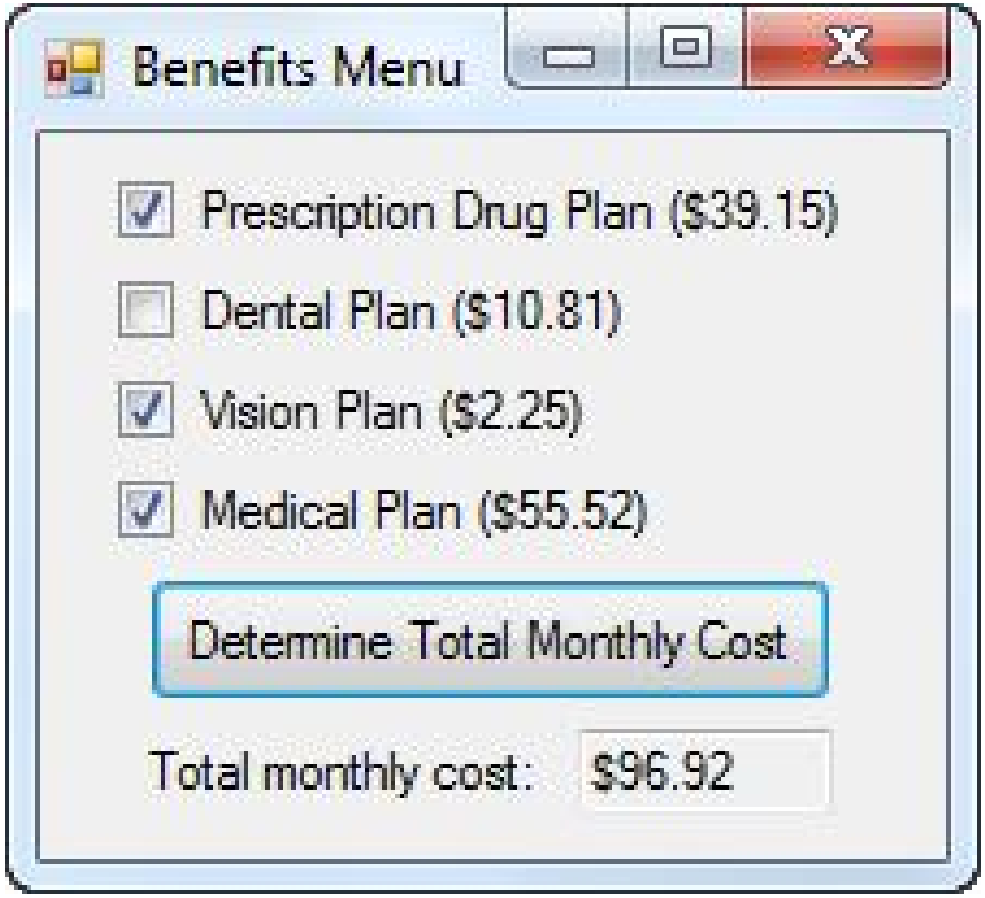
Example 4: Form



Example 4: Code for Button

```
Dim sum As Double = 0
If chkDrugs.Checked Then
  sum += 39.15
End If
If chkDental.Checked Then
  sum += 10.81
End If
If chkVision.Checked Then
  sum += 2.25
End If
If chkMedical.Checked Then
  sum += 55.52
End If
txtTotal.Text = FormatCurrency(sum)
```

Example 4: Output



Events Raised by a Selection

- SelectedIndexChanged – raised when a new item of a list box is selected
- CheckedChanged - raised when the user clicks on an unchecked radio button or a check box; that is, when the value of the Checked property is changed.

Example 5: Code

```
Private Sub checkBox_Changed(...) Handles _
                chkDrugs.CheckedChanged,
                chkDental.CheckedChanged,
                chkVision.CheckedChanged,
                chkMedical.CheckChanged
  Dim sum As Double = 0
  If chkDrugs.Checked Then
    sum += 39.15
  End If
```

(*continued on next slide*)

Example 5: Code (continued)

```
If chkDental.Checked Then
    sum += 10.81
  End If
  If chkVision.Checked Then
    sum += 2.25
  End If
  If chkMedical.Checked Then
    sum += 55.52
  End If
  txtTotal.Text = FormatCurrency(sum)
End Sub
```

Example 5: Output