Chapter 4 – Decision Making

A useful computer requires the ability to make decisions.

The key to programming the computer to make correct decisions is making sure you understand how to represent and evaluate the expression representing the decision properly.

Chapter 4 – Decision Making

4.1 If Statements

Visual Basic .NET offers more than one way for decisions to be made.

The **If** statement matches the idea of a single decision to a single result.

You can program an `If` statement by using the following code illustrating its syntax:

```
If (Expression) Then
    Program statements to execute if expression evaluates to True
End If
```

An If statement consists of an expression that determines whether a program statement or statements execute.

An expression can be the comparison of two values.

To compare values you may use any of the following operators:

<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
=	Equal to
<>	Not equal to

Chapter 4 – Decision Making

With either a variable or a constant value placed on both sides of the operator, an expression can be evaluated to either **`True`** or **`False`**.

When the condition in an `If` statement evaluates to `True`, the statements immediately following it are executed until an `End If` statement is reached.

If the `If` statement evaluates to `False`, the statements immediately after it, until the `End If`, are not executed.

Here are some expressions that evaluate to `True`:

`1 = 1`
`2 >= 1`
`2 >= 2`
`1 <= 2`
`1 < 2`
`1 <> 2`
`"a" <> "c"`
`"A" <> "a"`
`"D" = "D"`

Here are some expressions that evaluate to `False`:

`1 = 2`
`2 <= 1`
`2 < 2`
`3 > 4`
`1 >= 2`
`1 <> 1`
`"A" = "a"`
`"D" <> "D"`

Chapter 4 – Decision Making

Drill 4.1

Indicate whether each expression evaluates to `True` or `False`

```
1 (5 >= 4)
```
Answer: True

```
2 (-3 < -4)
```
Answer: False

```
3 (5 = 4)
```
Answer: False

```
4 (5 <> 4)
```
Answer: True

```
5 (4 >= 4)
```
Answer: True

```
6 (4 <= 4)
```
Answer: True

Chapter 4 – Decision Making

Simple If Statement

Write a program that will output a message if the user enters the word “Yes” in a text box.

By using an `If` statement, you can determine if the value that is entered in a text box is equal to the value you desire.

The following code is an example that compares the text box contents to the String “Yes”.

The code assumes that a `btnIf` button has been created to place the code and a `txtInput` text box and a `lblOutput` label have been created to hold the input and output.

```
Private Sub btnIf_Click(...
  If (txtInput.Text = “Yes”) Then
    lblOutput.Text = “This will output, because the user entered Yes”
  End If
End Sub
```

Chapter 4 – Decision Making

Simple If Statement Continued

What do you think would be contained in `lblOutput`:

1 If the user enters "`Yes`" in the `txtInput` text box?

Answer: The condition will evaluate to `True` and the text "This out output, because the user entered Yes" is placed in the `lblOutput` label.

2 If the user enters "No" in the `txtInput` text box?

Answer: The condition will evaluate to `False` and the `txtOutput` text box remain empty.

Chapter 4 – Decision Making

Simple If Statement with Code Following It

Some statements execute based on a decision and some regardless of the evaluation of the condition.

The following program is modified from the previous one.

```
Private Sub btnIf_Click(...
  If (txtInput.Text = "Yes") Then
    lblOutput.Text = "This will output, because the user entered Yes"
  End If

  lblOutput.Text &= " and this is here as well"
End Sub
```

Chapter 4 – Decision Making

Simple If Statement with Code Following It Continued

What do you think would be contained in `lblOutput`:

1 If the user enters "`Yes`" in the `txtInput` text box?

Answer: The output will be "This will output, because the user entered Yes and this is here as well"

2 If the user enters "No" in the `txtInput` text box?

Answer: The output will be " and this is here as well"

Chapter 4 – Decision Making

Drill 4.2

Given the following code, what do you think would be contained in `lblOutput`?

```
Private Sub btnIf_Click(...
  Dim intUserValue As Integer

  intUserValue = Val(txtInput.Text)

  If (intUserValue > 2) Then
    lblOutput.Text = "The first statement prints"
  End If
  lblOutput.Text = lblOutput.Text & " and the second statement prints"
End Sub
```

1 If the user enters 1 in the txtInput text box?

Answer: The output will be " and the second statement prints"

2 If the user enters 2 in the txtInput text box?

Answer: The output will be " and the second statement prints"

3 If the user enters 3 in the txtInput text box?

Answer: The output will be "The first statement prints and the second statement prints"

Chapter 4 – Decision Making

Drill 4.3

Given the following code, what do you think would be contained in `lblOutput`?

```
Private Sub btnIf_Click(...
  Dim intUserValue As Integer

  intUserValue = Val(txtInput.Text)

  If (intUserValue < 2) Then
    lblOutput.Text = "The first statement prints"
  End If
  lblOutput.Text = lblOutput.Text & " and the second statement prints"
End Sub
```

1 If the user enters 1 in the txtInput text box?

Answer: The output will be "The first statement prints and the second statement prints"

2 If the user enters 2 in the txtInput text box?

Answer: The output will be " and the second statement prints"

3 If the user enters 3 in the txtInput text box?

Answer: The output will be " and the second statement prints"

Chapter 4 – Decision Making

Drill 4.4

Given the following code, what do you think would be contained in `lblOutput`?

```
Private Sub btnIf_Click(...
  Dim intUserValue As Integer

  intUserValue = Val(txtInput.Text)

  If (intUserValue >= 2) Then
    lblOutput.Text = "The first statement prints"
  End If
  lblOutput.Text = lblOutput.Text & " and the second statement prints"
End Sub
```

1 If the user enters 1 in the txtInput text box?

Answer: The output will be " and the second statement prints"

2 If the user enters 2 in the txtInput text box?

Answer: The output will be "The first statement prints and the second statement prints"

3 If the user enters 3 in the txtInput text box?

Answer: The output will be "The first statement prints and the second statement prints"

Chapter 4 – Decision Making

Drill 4.5

Given the following code, what do you think would be contained in `lblOutput`?

```
Private Sub btnIf_Click(...
  Dim intUserValue As Integer

  intUserValue = Val(txtInput.Text)

  If (intUserValue <= 2) Then
    lblOutput.Text = "The first statement prints"
  End If
  lblOutput.Text = lblOutput.Text & " and the second statement prints"
End Sub
```

1 If the user enters 1 in the txtInput text box?

Answer: The output will be "The first statement prints and the second statement prints"

2 If the user enters 2 in the txtInput text box?

Answer: The output will be "The first statement prints and the second statement prints"

3 If the user enters 3 in the txtInput text box?

Answer: The output will be " and the second statement prints"

Chapter 4 – Decision Making

Example: In Stock?

Problem Description

The application will ask the user to enter the amount of a product a company has on hand.

If the number is greater than 0, then the program outputs that the “Product is in Stock”.

Otherwise, it outputs that the “Product is Sold Out”.

Problem Discussion

It will require creating a form with a `txtStockAmount` text box to store the amount of a product a company has in stock, a `lblAmount` label with the `Text` property set to “`Amount in Stock`”, another label, `lblInStock`, to hold a message, and a button with the `Text` property set to “`Calculate`”.

The code of the program compares the number entered by the user to 0.

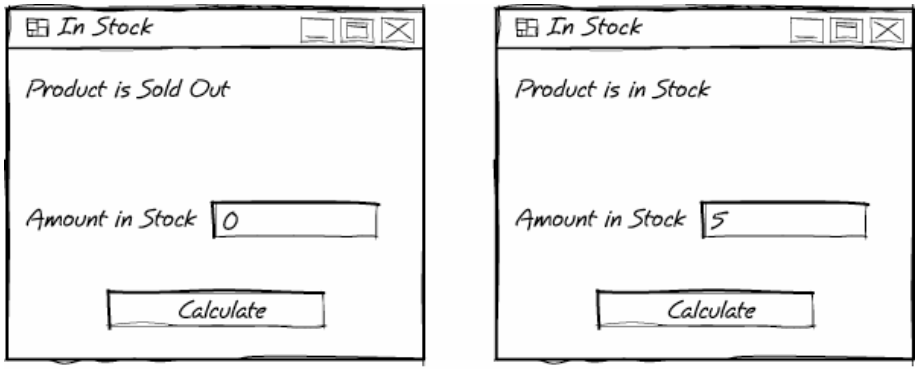


Fig 04-02

Chapter 4 – Decision Making

Problem Solution

Create Project and Form

Step 1: From the Start window, click on New Project. The New Project window will appear.

Step 2: Specify the name of the application as `InStock`.

Step 3: Specify the location as "`C:\VB Net Coach\Chapter 4\Code\` ".

Step 4: Click on the OK button.

Step 5: Rename the form to `frmInStock`.

Step 6: Rename the file by right-clicking on the file name in the Solution Explorer and setting the name to `frmInStock.vb`.

Step 7: Set the `Text` property of the form to `In Stock`.

Chapter 4 – Decision Making

Add the In Stock Label

Step 1: Place a label control across the top of the form.

Step 2: Set the `Name` property to `lblInStock`.

Step 3: Clear the `Text` property.

Add the Amount In Stock Label

Step 1: Place a label control to the right and about halfway down the form.

Step 2: Set the `Name` property to `lblAmount`.

Step 3: Set the `Text` property to `Amount in Stock`.

Step 4: Set the `Font Bold` property to `True`.

Add the Stock Amount Text Box

Step 1: Place a text box control below the In Stock label.

Step 2: Set the `Name` property to `txtStockAmount`.

Step 3: Clear out the default value from the `Text` property.

Chapter 4 – Decision Making

Add the Calculate Button

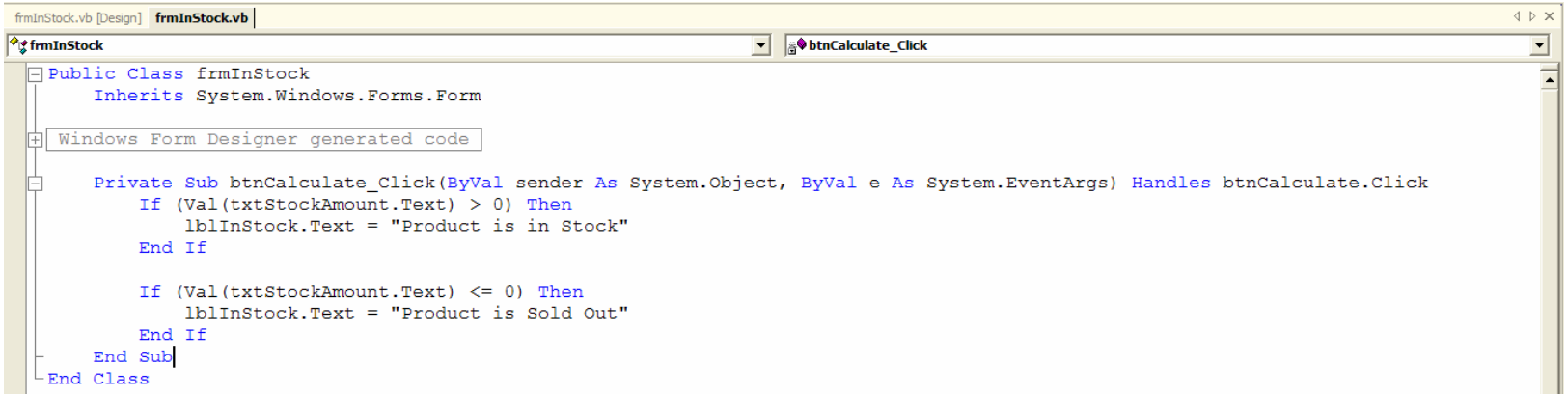
Step 1: Place a button control in the left side of the form, below the text box.

Step 2: Set the `Name` property to `btnCalculate`.

Step 3: Set the `Text` property to `Calculate`.

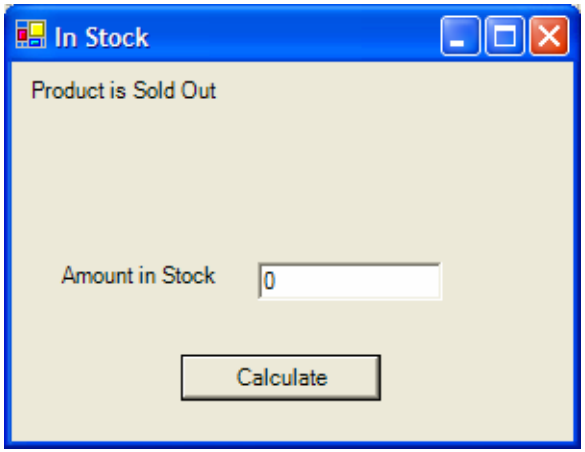
Step 4: Double-click on the button.

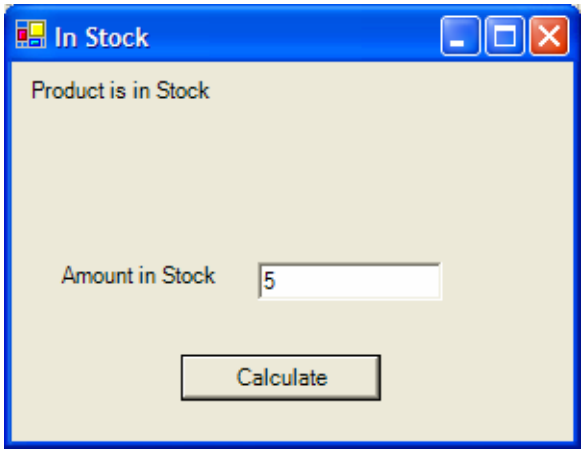
Step 5: Attach the code to output a message as to whether an item is in stock.



Chapter 4 – Decision Making

Here are two possible outputs.





Chapter 4 – Decision Making

Example: Expenses?

Problem Description

Write a program that outputs the difference between the amount of your income versus the amount of your expenses, as well as printing a message that indicates whether you are spending more than you are making.

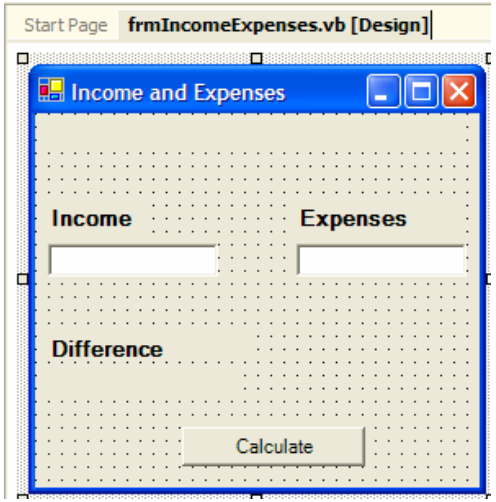
Problem Discussion

First, you must create a form that has two text boxes: `txtIncome` and `txtExpenses`.

Each should have a label above it indicating what is stored in the text box: income & expenses.

Additionally, you need two labels to store the difference between the income and expenses and one to hold your output message.

Finally, you need a button to calculate the difference and output the message.



Chapter 4 – Decision Making

Problem Solution

Create Project and Form

Step 1: From the Start window, click on New Project. The New Project window will appear.

Step 2: Specify the name of the application as `IncomeAndExpense`.

Step 3: Specify the location as "`C:\VB Net Coach\Chapter 4\Code\` ".

Step 4: Click on the OK button.

Step 5: Rename the form to `frmIncomeExpenses`.

Step 6: Rename the file by right-clicking on the file name in the Solution Explorer and setting the name to `frmIncomeExpenses.vb`.

Chapter 4 – Decision Making

Add the Result Label

Step 1: Place a label control across the top of the form.

Step 2: Set the `Name` property to `lblResult`.

Step 3: Remove the default value from the `Text` property.

Add the Income Label

Step 1: Place a label control a little below the `lblResult` label.

Step 2: Set the `Name` property to `lblIncome`.

Step 3: Set the `Text` property to `Income`.

Step 4: Set the `Font Bold` property to `True`.

Add the Income Text Box

Step 1: Place a text box control below the income label.

Step 2: Set the `Name` property to `txtIncome`.

Step 3: Clear out the default value from the `Text` property.

Chapter 4 – Decision Making

Add the Expense Label

Step 1: Place a label control to the right of the income label.

Step 2: Set the `Name` property to `lblExpenses`.

Step 3: Set the `Text` property to `Expenses`.

Step 4: Set the `Font Bold` property to `True`.

Add the Expense Text Box

Step 1: Place a text box control below the expenses label.

Step 2: Set the `Name` property to `txtExpenses`.

Step 3: Remove the default value from the `Text` property.

Chapter 4 – Decision Making

Add the Difference Title Label

Step 1: Place a label control below the income text box.

Step 2: Set the `Name` property to `lblDifferenceTitle`.

Step 3: Set the `Text` property to `Difference`.

Step 4: Set the `Font Bold` property to `True`.

Add the Difference Label

Step 1: Place a label control below the difference title label.

Step 2: Set the `Name` property to `lblDifference`.

Step 3: Remove the default value from the `Text` property.

Chapter 4 – Decision Making

Add the Calculate Button

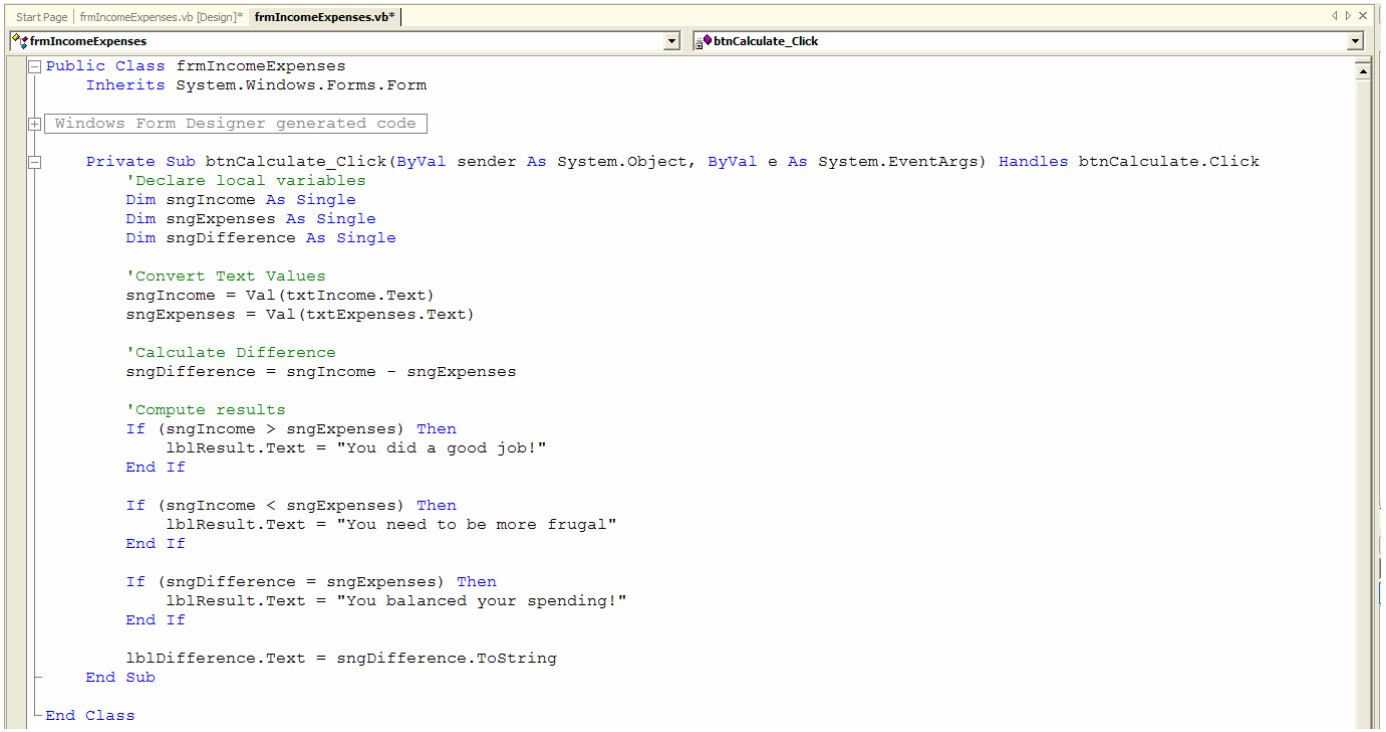
Step 1: Place a button control in the left side of the form, below the text box.

Step 2: Set the `Name` property to `btnCalculate`.

Step 3: Set the `Text` property to `Calculate`.

Step 4: Double-click on the button.

Step 5: Attach the code to output a message as to whether an item is in stock.



Chapter 4 – Decision Making

Add the Calculate Button Continued

The code for the button could also have been written by comparing the difference of the income and expenses to 0. This illustrates the fact that there are many different ways to solve the same problem.

Start Page | frmIncomeExpenses.vb [Design] | frmIncomeExpenses.vb

frmIncomeExpenses | btnCalculate_Click

```
Public Class frmIncomeExpenses
    Inherits System.Windows.Forms.Form

    Windows Form Designer generated code

    Private Sub btnCalculate_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnCalculate.Click
        'Declare local variables
        Dim sngIncome As Single
        Dim sngExpenses As Single
        Dim sngDifference As Single

        'Convert Text Values
        sngIncome = Val(txtIncome.Text)
        sngExpenses = Val(txtExpenses.Text)

        'Calculate Difference
        sngDifference = sngIncome - sngExpenses

        'Compute results
        If (sngDifference > 0) Then
            lblResult.Text = "You did a good job!"
        End If

        If (sngDifference < 0) Then
            lblResult.Text = "You need to be more frugal"
        End If

        If (sngDifference = 0) Then
            lblResult.Text = "You balanced your spending!"
        End If

        lblDifference.Text = sngDifference.ToString
    End Sub

End Class
```

Chapter 4 – Decision Making

Example: Voting Booth Application

Problem Description

With all the commotion surrounding the 2000 presidential election, a better voting booth is needed.

Throughout the next few chapters you will develop a number of Voting Booth applications.

You will see how, as you learn more commands and controls in the Visual Basic .NET language, you will be able to improve the accuracy of the voting booth.

Maybe you can sell it in Florida!

Chapter 4 – Decision Making

Problem Discussion

Your first application will allow voters to enter the name of the person they wish to vote for, thereby adding 1 for each vote to that person's counter.

You will have one counter for Bush, Gore, and Nader.

You will create a text box that will accept the name of the person to vote for and a button to process the actual vote.

You will add a results button that will display the final results of the election.

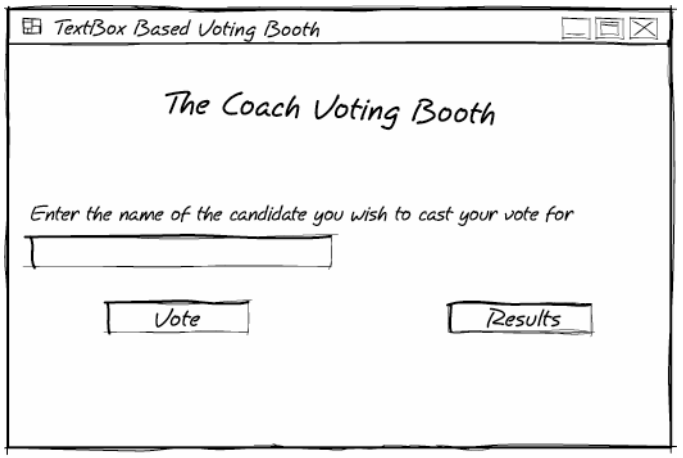


Fig 04-09

Chapter 4 – Decision Making

Problem Discussion Continued

In order to store the number of votes for each candidate, you will require a variable for each candidate.

Since the number of votes a candidate can have is a whole number, an `Integer` data type variable will be used.

These variables will need to be accessed from both the Vote and Results buttons' Click events.

Therefore, the variables will need to be declared in the `Declarations` section of the form.

One other issue you will have to deal with is to initialize these variables.

Technically, you do not have to because the default value for an `Integer` is `0`, but it is always a good habit to initialize them.

Forms, like all objects, have a special routine called a **constructor**.

A constructor is called before the actual object is completely created.

This is the appropriate place for initialization of variable in a form.

Chapter 4 – Decision Making

Problem Solution

Create Project and Form

Step 1: From the Start window, click on New Project. The New Project window will appear.

Step 2: Specify the name of the application as `Voting Booth 1`.

Step 3: Specify the location as "`C:\VB Net Coach\Chapter 4\Code\` ".

Step 4: Click on the OK button.

Step 5: Rename the form to `frmVotingBooth`.

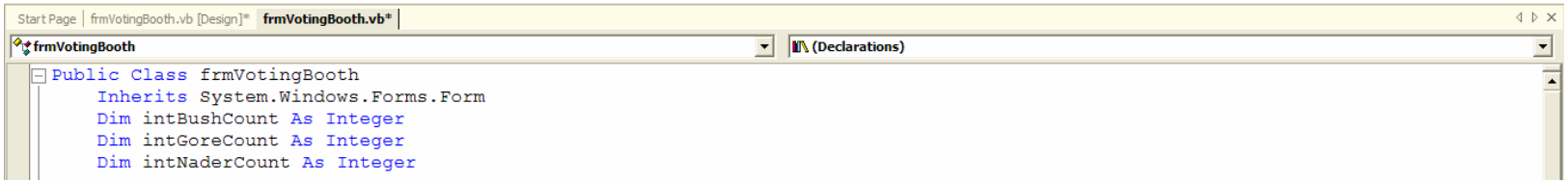
Step 6: Rename the file by right-clicking on the file name in the Solution Explorer and setting the name to `frmVotingBooth.vb`.

Step 7: Set the `Text` property of the form to `TextBox Based Voting Booth`.

Chapter 4 – Decision Making

Add Variable Declarations and Initialization

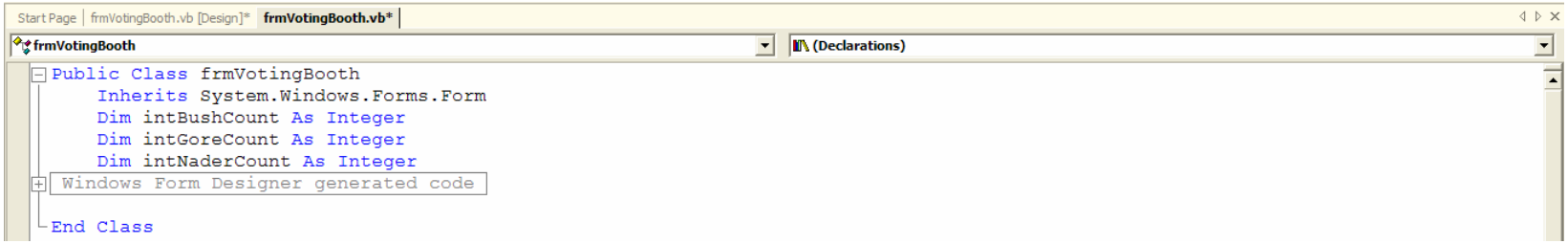
Step 1: Insert the code shown into the `Declarations` section of the form.



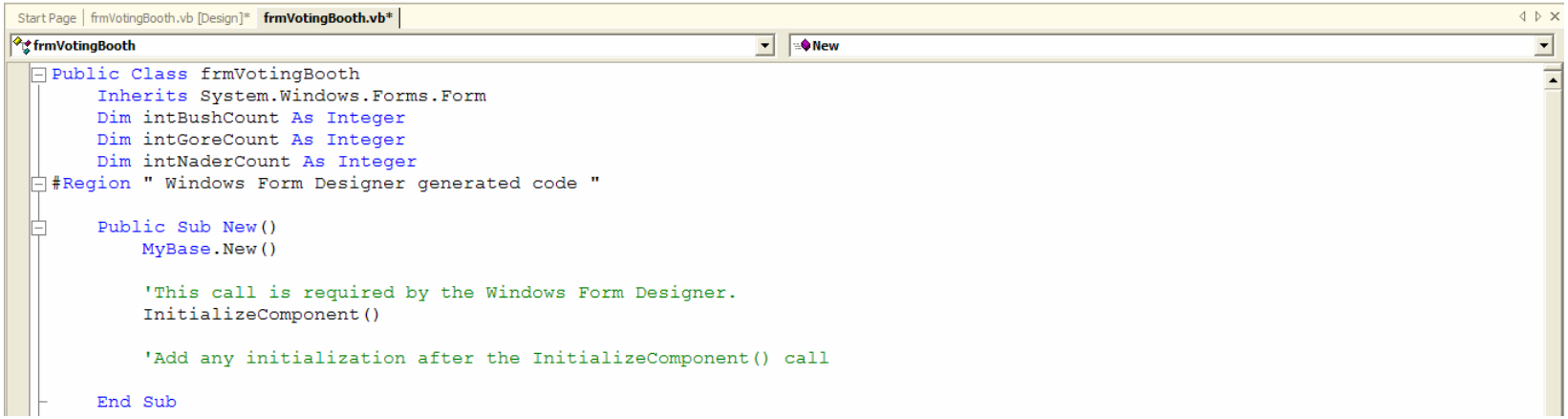
Chapter 4 – Decision Making

Add Variable Declarations and Initialization Continued

Step 2: Add the code to the form's constructor so that the variables are initialized. In order to add code to the constructor, make it visible. If you view the forms code before you have added any other code than the variable declarations, it will look like this:



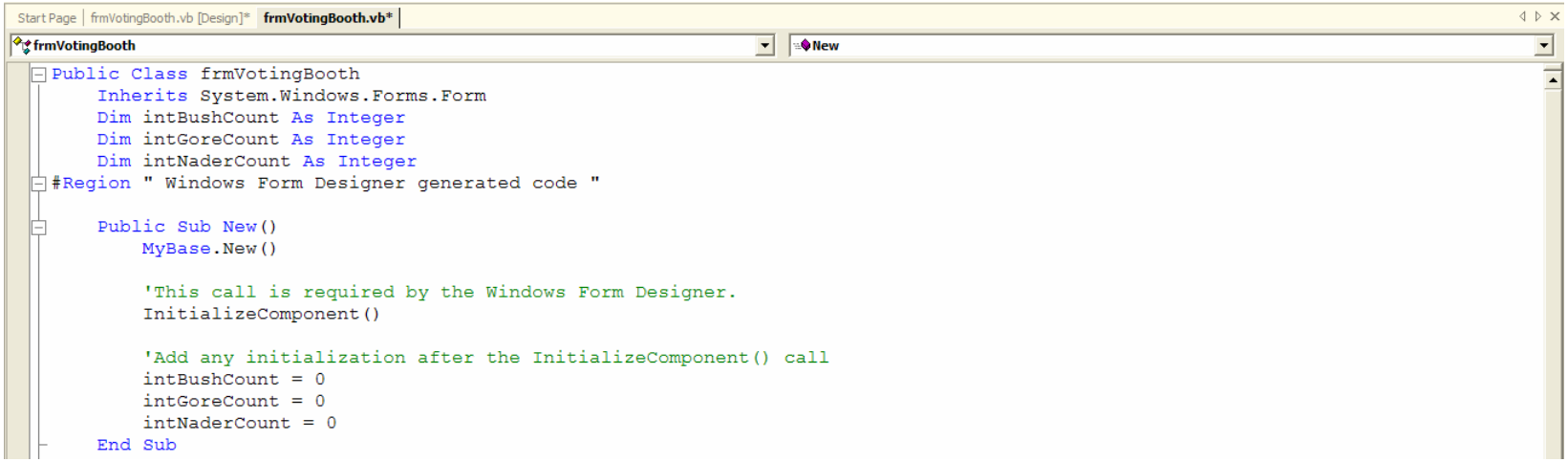
If you click on the + next to the Windows Form Designer generated code box, the code will expand:



Chapter 4 – Decision Making

Add Variable Declarations and Initialization Continued

Your code to initialize the variables should go directly after the comment '`Add any initialization after the InitializeComponent() call`':



Chapter 4 – Decision Making

Add Title Label

Step 1: Place a label control across the top of the form.

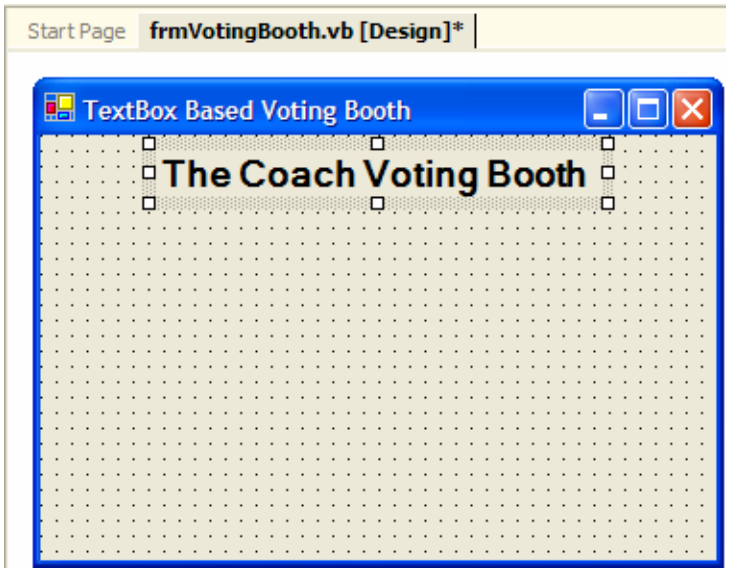
Step 2: Set the `Name` property to `lblTitle`.

Step 3: Set the `Text` property to `The Coach Voting Booth`.

Step 4: Set the `Font Bold` property to `True`.

Step 5: Set the `Font Size` property to `18`.

Step 6: Set the `TextAlign` property to `MiddleCenter`.



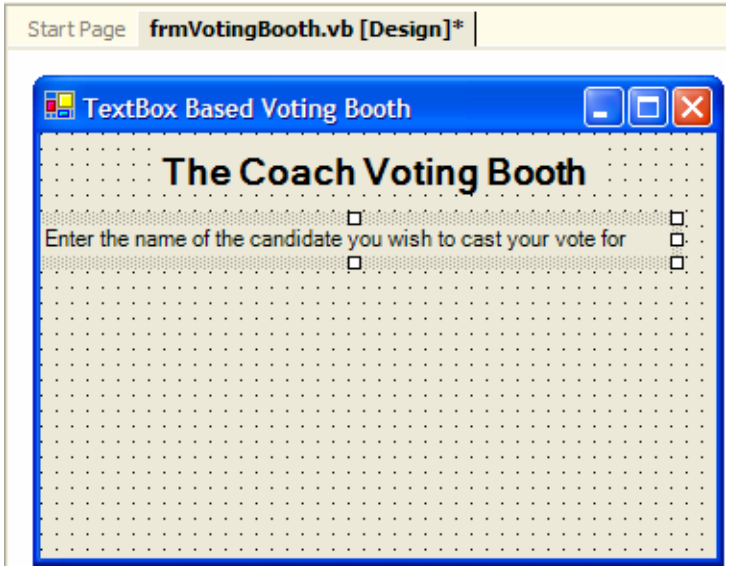
Chapter 4 – Decision Making

Add Instructions Label

Step 1: Place a label control below the previous one.

Step 2: Set the `Name` property to `lblDirections`.

Step 3: Set the `Text` property to `"Enter the name of the candidate you wish to cast your vote for"`.



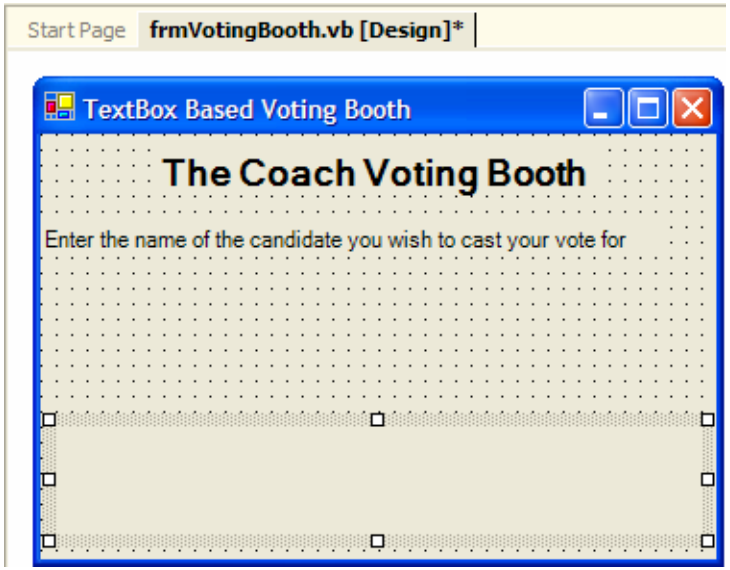
Chapter 4 – Decision Making

Add Results Label

Step 1: Place a label control at the bottom of the form. Make sure it is large enough to display the election results.

Step 2: Set the `Name` property to `lblResults`.

Step 3: Remove the default value from the `Text` property.



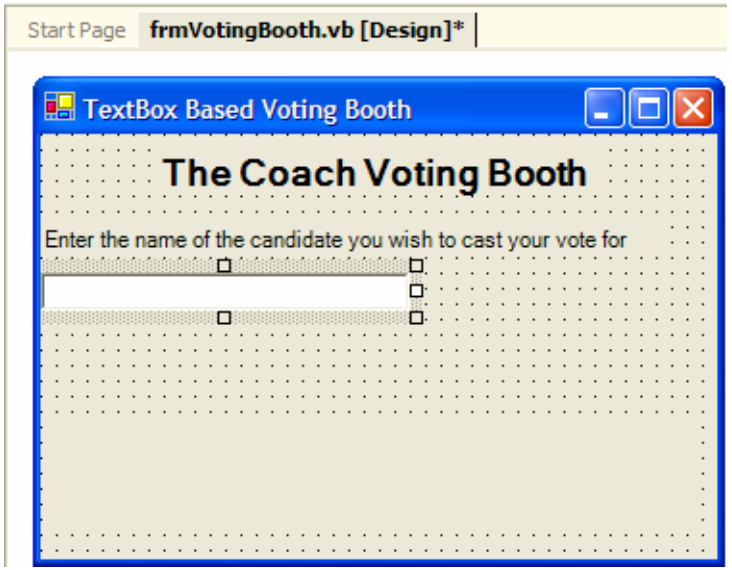
Chapter 4 – Decision Making

Add Voting Text Box

Step 1: Place a text box control below the instructions label.

Step 2: Set the `Name` property to `txtVote`.

Step 3: Clear out the default value from the `Text` property.



Chapter 4 – Decision Making

Add Vote Button

Step 1: Place a button control in the left side of the form, below the text box.

Step 2: Set the `Name` property to `btnVote`.

Step 3: Set the `Text` property to `Vote`.

Chapter 4 – Decision Making

Add Vote Button Continued

Step 4: Double-click on the button.

Step 5: Attach the code to process the vote. It must add 1 to the appropriate variable that stores the number of votes for each person.

```
Private Sub btnVote_Click(...
   If (txtVote.Text = "Bush") Then
      intBushCount = intBushCount + 1
   End If
   If (txtVote.Text = "Gore") Then
      intGoreCount = intGoreCount + 1
   End If
   If (txtVote.Text = "Nader") Then
      intNaderCount = intNaderCount + 1
   End If
   'Erase the vote
   txtVote.Text = ""
End Sub
```

Chapter 4 – Decision Making

Add Results Button

Step 1: Place a button control to the right of the other button.

Step 2: Set the `Name` property to `btnResults`.

Step 3: Set the `Text` property to `Results`.

Chapter 4 – Decision Making

Add Results Button Continued

Step 4: Double-click on the button.

Step 5: Attach the code to display the results of the election in the `lblResults` label control.

```
Private Sub btnResults_Click(...
   lblResults.Text = "Bush had " & Str(intBushCount) & _
   " Votes, Gore had " & intGoreCount & _
   " Votes, and Nader had " & intNaderCount & " Votes"
End Sub
```

Chapter 4 – Decision Making

What's Wrong with Your Application?

The voting system you have developed is problematic for a number of reasons:

1. It allows only three options to vote for. No way exists to enter choices other than the three.
2. If the name is entered in any variation of a proper spelling of the name other than the one in the If statement, then it will be ignored.
3. Finally, the program is inefficient because if the vote is for Bush, it still checks the other options.

Chapter 4 – Decision Making

4.2 Else and ElseIf Statements

Previous examples did not require something to be performed when the condition in the `If` statement evaluated to `False`.

Visual Basic .NET provides the **`Else`** and **`ElseIf`** keywords to handle these cases.

When an `If` statement's expression evaluates to `False`, the next `ElseIf` condition is evaluated.

If it evaluates to `True`, then the statements directly after it are executed.

Any additional `ElseIf` statements are evaluated in the same fashion.

After all `ElseIf` statements are evaluated, if they all evaluate to `False` and an `Else` statement is included, then the statements directly following the `Else` keyword will be executed.

```
If (Condition) Then
    Do Something
ElseIf (Condition 2) Then
    Do Something Else
ElseIf (Condition 3) Then
    Do Something Else
...
Else
    Do Something Else
End If
```

Chapter 4 – Decision Making

Simple If/Else Statement

Write a program similar to the earlier one that outputs a message if the user enters “Yes”.

If the user enters anything but “Yes”, then you will output a message indicating that “Yes” was not entered.

The code assumes that a `btnIfElse` button has been created to place the code and that `txtInput` text boxes and a `lblOutput` label were created to hold the input and output.

```
Private Sub btnIfElse_Click(...
  If (txtInput.Text = "Yes") Then
    lblOutput.Text = "The user answered the question with a Yes"
  Else
    lblOutput.Text = "The user did not answer the question with a Yes"
  End If
End Sub
```

If the user enters “Yes”, the text “The user answered the question with a Yes” is placed in the label. Otherwise, “The user did not answer the question with a Yes” is placed in the label.

Chapter 4 – Decision Making

Another Simple If/Else Statement Example

Write a program that will output a message if a discount will be applied, which is to happen if the purchase price is more than $100.

If the purchase price is more than $100, then the code will place "`DISCOUNT`" in `txtOutput`.

Otherwise, the code will place "`FULL PRICE`" in the text box.

The code assumes that a `btnIfElse` button has been created to place the code and that a text boxes `txtInput` was create to hold the input and label `lblOutput` the output.

```
Private Sub btnIfElse_Click(...
  Dim sngPurchasePrice As Single

  sngPurchasePrice = Val(txtInput.Text)

  If (sngPurchasePrice > 100) Then
    lblOutput.Text = "DISCOUNT"
  Else
    lblOutput.Text = "FULL PRICE"
  End If
End Sub
```

Another Simple If/Else Statement Continued

What do you think would be contained in `lblOutput`:

1 If the user enters `199.95` in the `txtInput` text box?

Answer: The condition will evaluate to `True` and the text "DISCOUNT" is placed in the `lblOutput` label. No other statements after `Else` are executed.

2 If the user enters `99.95` in the `txtInput` text box?

Answer: The condition will evaluate to `False`. All the statements until `Else` are not executed. The text "`FULL PRICE`" is placed in the `lblOutput` text box.

Chapter 4 – Decision Making

Drill 4.6

Using the same application, but changing the code in the button as follows, what do you think the output would be if the value entered by the user is 0, 1, and then 2, respectively?

```
Private Sub btnIfElse_Click(...
  Dim intDrillValue As Integer

  intDrillValue = Val(txtInput.Text)

  If (intDrillValue <= 1) Then
    lblOutput.Text = "This will output, because intDrillValue <= 1"
  Else
    lblOutput.Text = "Instead this outputs, because intDrillValue > 1"
  End If
  lblOutput.Text &= " and this is here as well"
End Sub
```

Answer: If the input is 0, the output is "This will output, because intDrillValue <= 1 and this is here as well".

If the input is 1, the output is "This will output, because intDrillValue <= 1 and this is here as well".

If the input is 2, the output is "Instead this outputs, because intDrillValue > 1 and this is here as well".

Chapter 4 – Decision Making

Drill 4.7

Using the same application, but changing the code in the button as follows, what do you think the output would be if the value entered by the user is 0, 1, and then 2, respectively?

```
Private Sub btnIfElse_Click(...
  Dim intDrillValue As Integer

  intDrillValue = Val(txtInput.Text)

  If (intDrillValue < 1) Then
    lblOutput.Text = "This will output, because intDrillValue < 1"
  Else
    lblOutput.Text = "Instead this outputs, because intDrillValue >= 1"
  End If
  lblOutput.Text &= " and this is here as well"
End Sub
```

Answer: If the input is 0, the output is "This will output, because intDrillValue < 1 and this is here as well".

If the input is 1, the output is "Instead this outputs, because intDrillValue >= 1 and this is here as well".

If the input is 2, the output is "Instead this outputs, because intDrillValue >= 1 and this is here as well".

Chapter 4 – Decision Making

Simple If/ElseIf/Else Statement

Write a program that applies a varied discount based on the total purchase price.

The application should compute how much of a discount should be applied to a purchase.

If the purchase price is more than $100, then the discount should be 5%.

If the purchase price is more than $500, then the discount should be 10%.

The code should place the amount of the discount in the `lblOutput` label. If no discount is applied, then place the String "`NO DISCOUNT`" in the label.

```
Private Sub btnIfElse_Click(...
    Dim sngPurchasePrice As Single

    sngPurchasePrice = Val(txtInput.Text)

    If (sngPurchasePrice > 500) Then
      lblOutput.Text = (sngPurchasePrice * 0.1).ToString()
    ElseIf (sngPurchasePrice > 100) Then
      lblOutput.Text = (sngPurchasePrice * 0.05).ToString()
    Else
      lblOutput.Text = "NO DISCOUNT"
    End If
End Sub
```

Chapter 4 – Decision Making

Simple If/ElseIf/Else Statement

What do you think would be contained in `lblOutput`:

1 If the user enters `600.00` in the `txtInput` text box?

Answer: The first condition will evaluate to `True` and the text “60” is placed in the `lblOutput` label. No other statements after `ElseIf` or `Else` are executed.

2 If the user enters `250.00` in the `txtInput` text box?

Answer: The first condition will evaluate to `False`. All the statements until `ElseIf` are not executed. The second condition will evaluate to `True` and the text “12.5” is placed in the `lblOutput` label. No other statements after `Else` are executed.

3 If the user enters `50.00` in the `txtInput` text box?

Answer: The first and second condition will evaluate to `False`. Only the statements after the `Else` statement and before the `End If` are executed and the text “NO DISCOUNT” is placed in the `lblOutput` label.

Chapter 4 – Decision Making

Drill 4.8

Assume that the code for the previous example was instead coded as follows:

```
Private Sub btnIfElse_Click(...
    Dim sngPurchasePrice As Single

    sngPurchasePrice = Val(txtInput.Text)

    If (sngPurchasePrice > 100) Then
      lblOutput.Text = (sngPurchasePrice * 0.05).ToString()
    ElseIf (sngPurchasePrice > 500) Then
      lblOutput.Text = (sngPurchasePrice * 0.1).ToString()
    Else
      lblOutput.Text = "NO DISCOUNT"
    End If
End Sub
```

Chapter 4 – Decision Making

Drill 4.8 Continued

What do you think would be contained in `lblOutput`:

1 If the user enters `600.00` in the `txtInput` text box?

Answer: The first condition will evaluate to `True` and the text “30” is placed in the `lblOutput` label. No other statements after `ElseIf` or `Else` are executed.

2 If the user enters `250.00` in the `txtInput` text box?

Answer: The first condition will evaluate to `True` and the text “12.5” is placed in the `lblOutput` label. No other statements after `ElseIf` or `Else` are executed.

3 If the user enters `50.00` in the `txtInput` text box?

Answer: The first and second condition will evaluate to `False`. Only the statements after the `Else` statement and before the `End If` are executed and the text “NO DISCOUNT” is placed in the `lblOutput` label.

Chapter 4 – Decision Making

Drill 4.9

The code assumes that a `btnIfElse` button has been created to place the code and that a `txtInput` text box and a `lblOutput` label were created to hold the input and output, respectively. What do you think the output would be if the value entered by the user is -1, 0, and then 1, respectively?

```
Private Sub btnIfElse_Click(...
  Dim intDrillValue As Integer

  intDrillValue = Val(txtInput.Text)

  If (intDrillValue > 0) Then
    lblOutput.Text = "The number if positive"
  ElseIf (intDrillValue < 0) Then
    lblOutput.Text = "The number is negative"
  Else
    lblOutput.Text = " I got a big zero"
End Sub
```

Answer: If the input is -1, the output is "The number is negative"

If the input is 0, the output is "I got a big zero"

If the input is 1, the output is "The number is positive"

Chapter 4 – Decision Making

Example: Letter Grade Program

Problem Description

Write a program that will display a letter grade based on a number grade entered.

The program should assign an A if the grade is greater than or equal to 90, a B if the grade is between an 80 and an 89, a C if the grade is between a 70 and a 79, and a D if the grade is between a 60 and a 69. Otherwise, the program assigns an F.

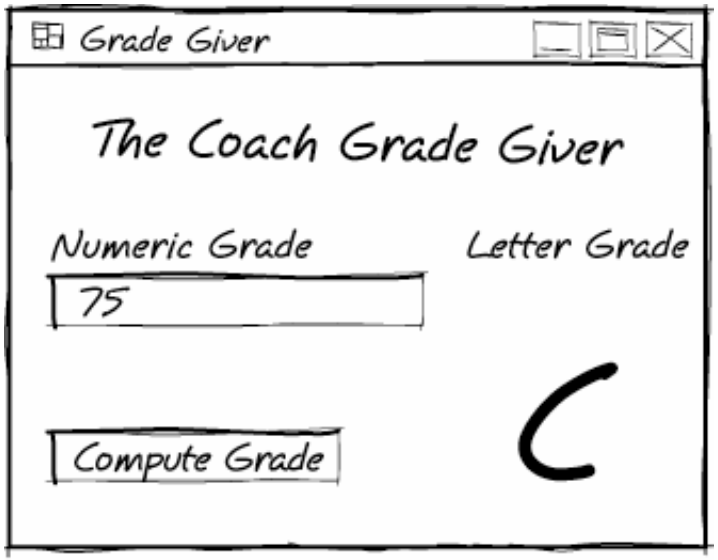


Fig 04-23

Chapter 4 – Decision Making

Problem Discussion

The application will require a text box to accept the numerical grade and a label to output the result.

The actual computation of the letter grade will be performed in the `Click` event of a button.

The letter grade can be determined using an `If` statement with a few `ElseIf` statements checking the range of each possible letter grade.

Using an `If` statement with `ElseIf` statements is preferred over using a series of `If` statements because once a letter grade has been determined, it would be wasteful to check the remaining `If` statements.

Chapter 4 – Decision Making

Problem Solution

Create Project and Form

Step 1: From the Start window, click on New Project. The New Project window will appear.

Step 2: Specify the name of the application as `Grade Giver`.

Step 3: Specify the location as "`C:\VB Net Coach\Chapter 4\Code\` ".

Step 4: Click on the OK button.

Step 5: Rename the form to `frmGradeGiver`.

Step 6: Rename the file by right-clicking on the file name in the Solution Explorer and setting the name to `frmGradeGiver.vb`.

Step 7: Set the `Text` property of the form to `Grade Giver`.

Chapter 4 – Decision Making

Add Title Label

Step 1: Place a label control across the top of the form.

Step 2: Set the `Name` property to `lblTitle`.

Step 3: Set the `Text` property to `The Coach Grade Giver`.

Step 4: Set the `Font Size` property to `18`.

Step 5: Set the `Font Bold` property to `True`.

Step 6: Set the `TextAlign` property to `MiddleCenter`.

Chapter 4 – Decision Making

Add Numeric Grade Label

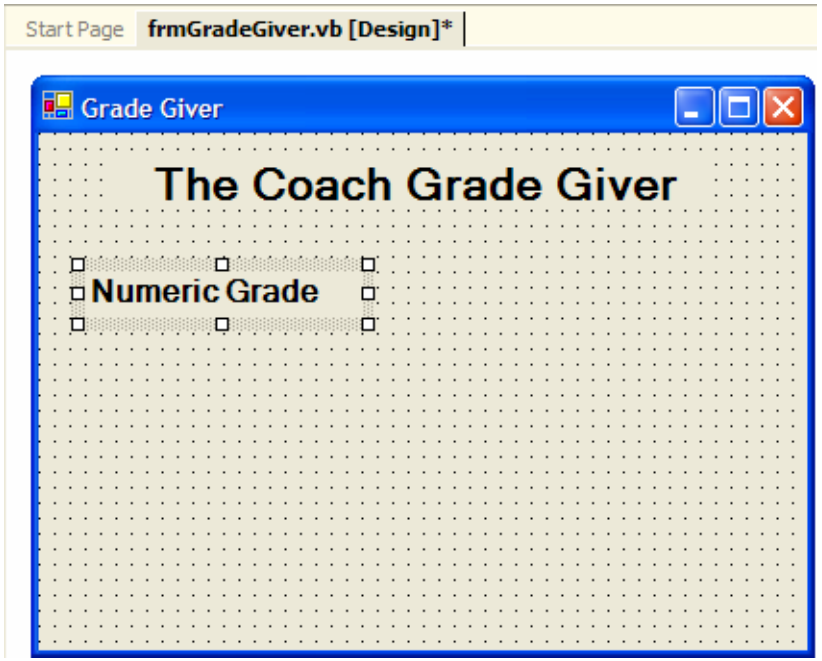
Step 1: Place a label control near the left side of the form.

Step 2: Set the `Name` property to `lblNumericGradeTitle`.

Step 3: Set the `Text` property to `Numeric Grade`.

Step 4: Set the `Font Size` property to `12`.

Step 5: Set the `Font Bold` property to `True`.



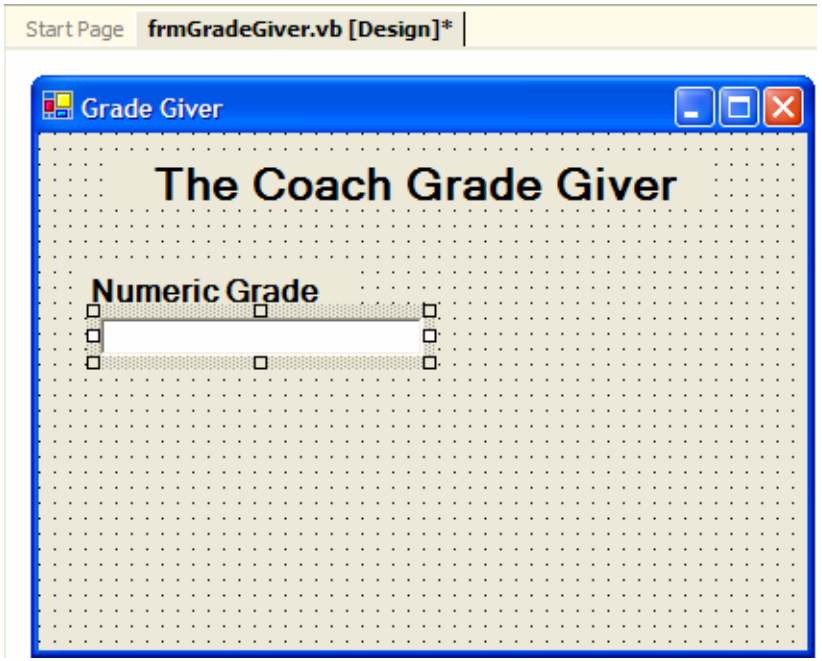
Chapter 4 – Decision Making

Add Numeric Grade Text Box

Step 1: Place a text box control below the numeric grade label.

Step 2: Set the `Name` property to `txtNumericGrade`.

Step 3: Clear out the default value from the `Text` property.



Chapter 4 – Decision Making

Add Letter Grade Label

Step 1: Place a label control near the right side of the form.

Step 2: Set the `Name` property to `lblLetterGradeTitle`.

Step 3: Set the `Text` property to `Letter Grade`.

Step 4: Set the `Font Size` property to `12`.

Step 5: Set the `Font Bold` property to `True`.

Chapter 4 – Decision Making

Add lblGrade Label

Step 1: Place a label control below the letter grade title label.

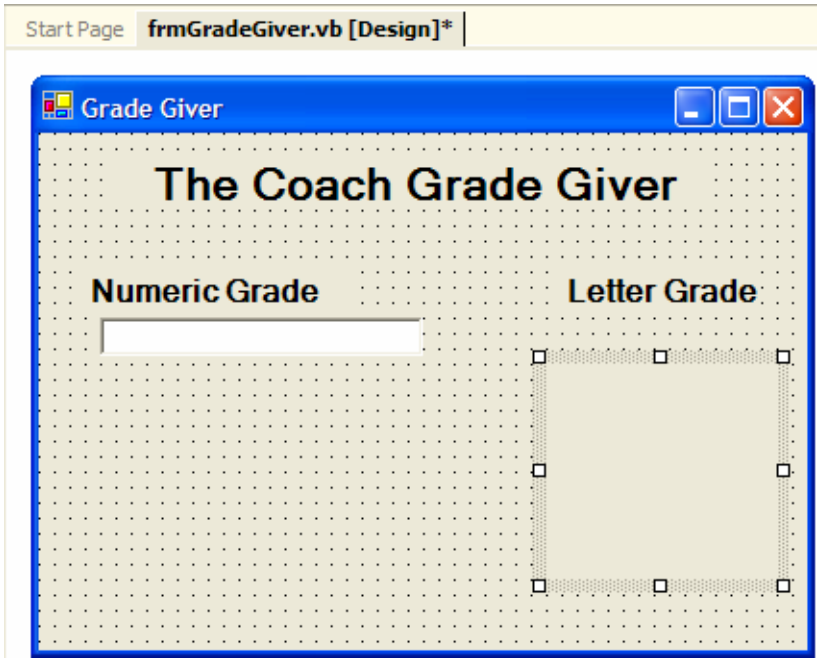
Step 2: Set the `Name` property to `lblLetterGrade`.

Step 3: Clear the `Text` property.

Step 4: Set the `Font Size` property to `48`.

Step 5: Set the `Font Bold` property to `True`.

Step 6: Set the `TextAlign` property to `MiddleCenter`.



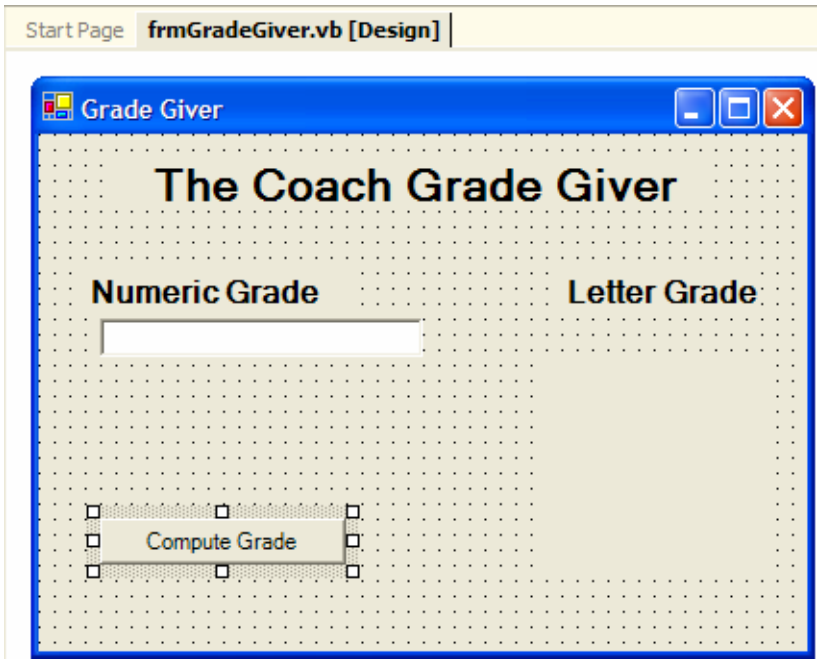
Chapter 4 – Decision Making

Add Compute Grade Button

Step 1: Place a button control in the bottom left side of the form.

Step 2: Set the `Text` property to `Compute Grade`.

Step 3: Set the `Name` property to `btnCompute`.

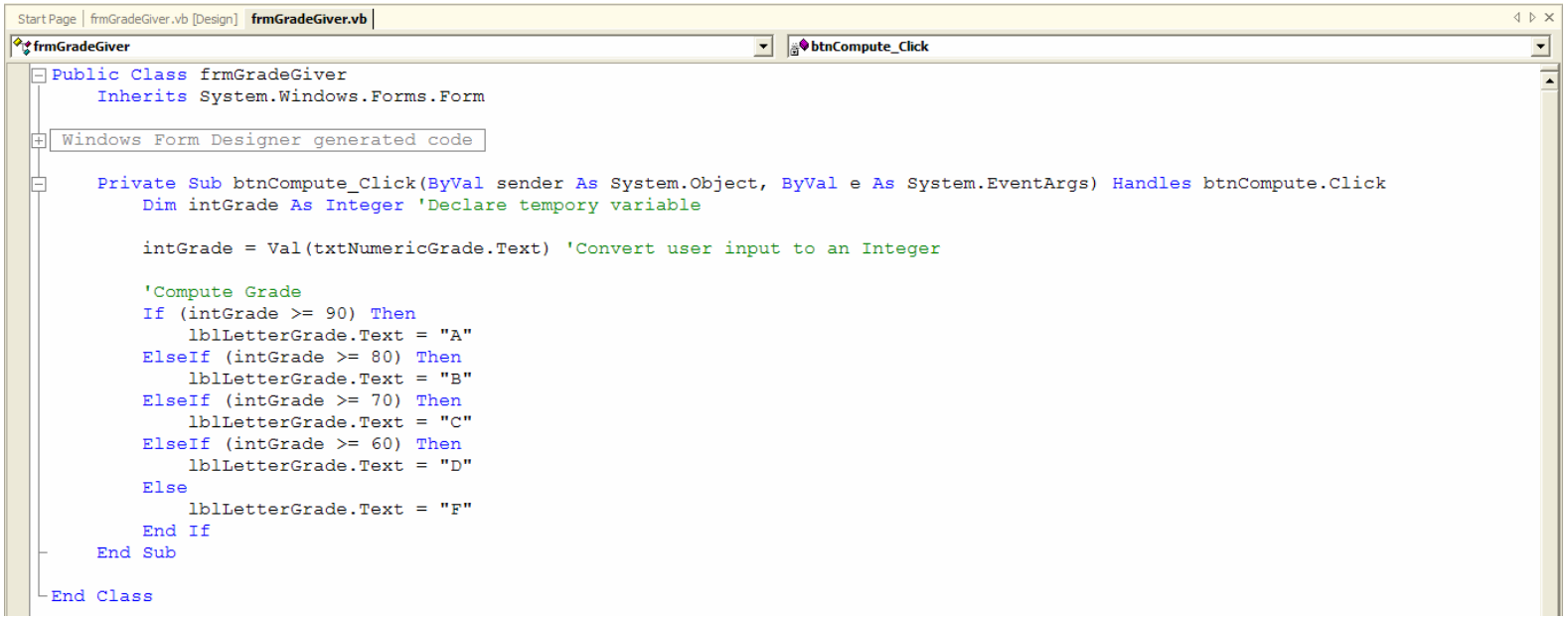


Chapter 4 – Decision Making

Add Compute Grade Button Continued

Step 4: Double-click on the button.

Step 5: Attach the code to display the results of the grade calculation in the `lblLetterGrade` label control.




```
Public Class frmGradeGiver
    Inherits System.Windows.Forms.Form

    Windows Form Designer generated code

    Private Sub btnCompute_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnCompute.Click
        Dim intGrade As Integer 'Declare tempory variable

        intGrade = Val(txtNumericGrade.Text) 'Convert user input to an Integer

        'Compute Grade
        If (intGrade >= 90) Then
            lblLetterGrade.Text = "A"
        ElseIf (intGrade >= 80) Then
            lblLetterGrade.Text = "B"
        ElseIf (intGrade >= 70) Then
            lblLetterGrade.Text = "C"
        ElseIf (intGrade >= 60) Then
            lblLetterGrade.Text = "D"
        Else
            lblLetterGrade.Text = "F"
        End If
    End Sub

End Class
```

Chapter 4 – Decision Making

Example: Improved Voting Booth

Problem Description

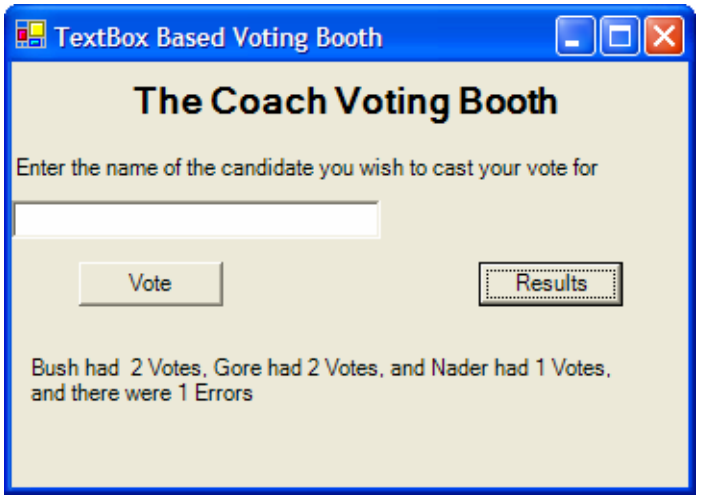
Previously, your Voting Booth application did not keep track of the number of errors in voting.

Aside from curiosity's sake, there is an important reason to track these errors.

A good voting machine should prevent mistakes from ever being entered.

The application will look relatively the same.

The only visible difference will be the additional of the display of the number of improper votes being cast.



Chapter 4 – Decision Making

Problem Discussion

Your new application will take advantage of `ElseIf` and `Else` statements to total the votes more efficiently as well as keep a total of improper votes.

By using the `ElseIf` statement, you can process each vote more efficiently.

By using the `Else` statement, you can capture all of the errors.

Chapter 4 – Decision Making

Problem Solution

Create Project and Form

Step 1: From the Start window, click on New Project. The New Project window will appear.

Step 2: Specify the name of the application as `ErrorTrackingVotingBooth`.

Step 3: Specify the location as "`C:\VB Net Coach\Chapter 4\Code\` ".

Step 4: Click on the OK button.

Step 5: Rename the form to `frmVoting`.

Step 6: Rename the file by right-clicking on the file name in the Solution Explorer and setting the name to `frmVoting.vb`.

Step 7: Set the `Text` property to `TextBox Based Voting Booth`.

Chapter 4 – Decision Making

Modify the Applications' Code

Follow the instructions on adding the controls as before. The code, however, has a few modifications.

You must first declare an additional variable, in the `Declarations` section of the form, to hold the number of errors encountered.

```
Dim intBushCount As Integer
Dim intGoreCount As Integer
Dim intNaderCount As Integer
Dim intErrorCount As Integer
```

Chapter 4 – Decision Making

Modify the Applications' Code Continued

You must make sure that you initialize that variable to 0 in the constructor of the form.

```
Public Sub New()
    MyBase.New()

    'This call is required by the Windows Form Designer.
    InitializeComponent()

    'Add any initialization after the InitializeComponent() call
    intBushCount = 0
    intGoreCount = 0
    intNaderCount = 0
    intErrorCount = 0
End Sub
```

Chapter 4 – Decision Making

Modify the Applications' Code Continued

You need to modify the `btnVote` button so that it will use `ElseIf` and `Else` statements to process the vote efficiently and so that it now records the number of errors by using the `Else` statement.

```
Private Sub btnVote_Click(...
     If (txtVote.Text = "Bush") Then
         intBushCount += 1
     ElseIf (txtVote.Text = "Gore") Then
         intGoreCount += 1
     ElseIf (txtVote.Text = "Nader") Then
         intNaderCount += 1
     Else
         intErrorCount += 1
     End If
     'Erase the vote
     txtVote.Text = ""
End Sub
```

Chapter 4 – Decision Making

Modify the Applications' Code Continued

You need to modify the `btnResults` button so that it will display the additional information.

```
Private Sub btnResults_Click(...
     lblResults.Text = "Bush had " & Str(intBushCount) & _
               " Votes, Gore had " & intGoreCount & _
               " Votes, and Nader had " & intNaderCount & " Votes" & _
               ", and there were " & intErrorCount & " Errors"
End Sub
```

Chapter 4 – Decision Making

4.3 Compound Conditional Statements

Sometimes comparisons are not as simple as a single comparison.

The more complex conditions are known as **compound expressions**.

Visual Basic .NET gives you additional expression operators to help you map a problem or algorithm to a program. **Boolean** logic operators like `And`, `Or`, and `Not` assist you in representing a condition.

Chapter 4 – Decision Making

`And` is used to represent the logical "anding" of two conditions. A simple truth table of all the possible conditions follows:

`True And True = True`
`True And False = False`
`False And True = False`
`False And False = False`

`Or` is used to represent the logical or. Here is a simple truth table of all the possibilities:

`True Or True = True`
`True Or False = True`
`False Or True = True`
`False Or False = False`

`Not` is used to negate the value of an expression. Here is a truth table of all the possibilities:

`Not True = False`
`Not False = True`

Chapter 4 – Decision Making

Here are some expressions that evaluate to `True`:

`(1 = 1) And (2 = 2)`
`(2 >= 1) Or (1 <> 1)`
`(2 >= 2) And (1 < 3)`
`(1 <= 2) Or (2 > 1)`
`(1 < 2) And (1 <> 2)`
`("CAT" = "CAT") And (1 < 2)`
`("a" <> "c") Or ("b" <> "c")`
`Not ("A" = "a")`

Here are some expressions that evaluate to False:

`(1 = 2) Or (2 = 1)`
`(2 <= 1) Or (1 > 2)`
`(2 < 2) And (1 = 1)`
`(3 > 4) And (3 < 5)`
`(1 >= 2) Or (2 < 1)`
`Not (1 = 1)`
`("a" = "A") Or ("b" = "B")`

Chapter 4 – Decision Making

Drill 4.10

Indicate whether each expression evaluates to `True` or `False`

```
1 Not (5 >= 4)
```
Answer: False

```
2 (-3 < -4) Or (1 = 1)
```
Answer: True

```
3 ("BOB" = "bob") And (2 >= 2)
```
Answer: False

```
4 (2 < 1) Or (5 <> 4)
```
Answer: True

```
5 (1 < 2) Or (4 >= 4)
```
Answer: True

```
6 Not (4 <= 4) And (1 <= 1)
```
Answer: False

Chapter 4 – Decision Making

If Statement Using an And Operator

You can use compound conditional expressions in a program the same way as with the previous conditional statements.

The following code shows the use of a compound conditional statement. The code assumes that a button, `btnCompoundIf`, has been created to contain the code. Three text boxes – `txtRetailPrice`, `txtSalePrice`, and `txtOutput` have been created to hold the input and output of the user.

```
Private Sub btnCompoundIf_Click(...
   Dim sngRetailPrice As Single
   Dim sngSalesPrice As Single

   sngRetailPrice = Val(txtRetailPrice.Text)
   sngSalesPrice = Val(txtSalesPrice.Text)

   If ((sngRetailPrice = sngSalesPrice) And (sngRetailPrice > 100)) Then
       txtOutput.Text = "This product is not on sale and is expensive"
   Else
       txtOutput.Text = "This product may not be too expensive and " _
                        "may be on sale"
   End If
End Sub
```

Chapter 4 – Decision Making

If Statement Using an And Operator Continued

What do you think would be contained in `txtOutput`:

1 If the user enters `50.25` for the retail price and `50.25` for the sales price?

Answer: The condition will evaluate to `False` and the text “This product may not be too expensive and may be on sale” is assigned to the output text box.

2 If the user enters 12`5.13` for the retail price and 125`.13` for the sales price?

Answer: The condition will evaluate to `True` and the text “This product is not on sale and is expensive” is assigned to the output text box.

3 If the user enters 150`.00` for the retail price and 125`.13` for the sales price?

Answer: The condition will evaluate to `False` and the text “This product may not be too expensive and may be on sale” is assigned to the output text box.

4 If the user enters 99`.90` for the retail price and 75`.00` for the sales price?

Answer: The condition will evaluate to `False` and the text “This product may not be too expensive and may be on sale” is assigned to the output text box.

Chapter 4 – Decision Making

If Statement Using an Or Operator

This code demonstrates the use of an `Or` operator. It assumes that a button, `btnCompoundIf`, has been created to contain the code.

Three text boxes – `txtRetailPrice`, `txtSalePrice`, and `txtOutput` have been created to hold the input and output of the user.

```
Private Sub btnCompoundIf_Click(...
    Dim sngRetailPrice As Single
    Dim sngSalesPrice As Single

    sngRetailPrice = Val(txtRetailPrice.Text)
    sngSalesPrice = Val(txtSalesPrice.Text)

    If ((sngRetailPrice = sngSalesPrice) Or (sngRetailPrice > 100)) Then
        txtOutput.Text = "This product is either not on sale or very
                          expensive"
    Else
        txtOutput.Text = "This product is on sale and not expensive"
    End If
End Sub
```

If Statement Using an Or Operator Continued

What do you think would be contained in `txtOutput`:

1 If the user enters `50.25` for the retail price and `50.25` for the sales price?

Answer: The condition will evaluate to `True` and the text “This product is either not on sale or very expensive” is assigned to the output text box.

2 If the user enters 12`5.13` for the retail price and 125`.13` for the sales price?

Answer: The condition will evaluate to `True` and the text “This product is either not on sale or very expensive” is assigned to the output text box.

3 If the user enters 150`.00` for the retail price and 125`.13` for the sales price?

Answer: The condition will evaluate to `True` and the text “This product is either not on sale or very expensive” is assigned to the output text box.

4 If the user enters 99`.90` for the retail price and 75`.00` for the sales price.

Answer: The condition will evaluate to `False` and the text “This product is on sale and not expensive” is assigned to the output text box.

Chapter 4 – Decision Making

If Statement Using a Not Operator

This code demonstrates the use of an `Not` operator.

It assumes that a button, `btnCompoundIf`, has been created to contain the code as in previous examples.

```
Private Sub btnCompoundIf_Click(...
   Dim sngRetailPrice As Single
   Dim sngSalesPrice As Single

   sngRetailPrice = Val(txtRetailPrice.Text)
   sngSalesPrice = Val(txtSalesPrice.Text)

   If (Not (sngRetailPrice >= sngSalesPrice)) Then
     txtOutput.Text = "The Sales Price is greater than the Retail Price"
   Else
     txtOutput.Text = "The Sales Price is less than or equal to "_
                      "the Retail Price"
   End If
End Sub
```

If Statement Using a Not Operator Continued

What do you think would be contained in `txtOutput`:

1 If the user enters `50.25` for the retail price and `50.25` for the sales price?

Answer: The condition will evaluate to `False` and the text “The Sales Price is less than or equal to the Retail Price” is assigned to the output text box.

2 If the user enters 49.95 for the retail price and `125.13` for the sales price?

Answer: The condition will evaluate to `True` and the text “The Sales Price is greater than the Retail Price” is assigned to the output text box.

Chapter 4 – Decision Making

Drill 4.11

Use the same application as the previous drills, but change the code in the button as follows:

```
Private Sub btnCompoundIf_Click(...
   Dim sngRetailPrice As Single
   Dim sngSalesPrice As Single

   sngRetailPrice = Val(txtRetailPrice.Text)
   sngSalesPrice = Val(txtSalesPrice.Text)

   If ((sngRetailPrice >= sngSalesPrice) And _
       (Not (sngSalesPrice > 75))) Then
       txtOutput.Text = "This crazy drill outputs True"
   Else
     txtOutput.Text = "This crazy drill outputs False"
   End If
End Sub
```

Drill 4.11 Continued

What do you think would be contained in `txtOutput`:

1 If the user enters `99.95` for the retail price and `50.25` for the sales price?

Answer: The condition will evaluate to `True` and the text “This crazy drill outputs True” is assigned to the output text box.

2 If the user enters 199.95 for the retail price and 99.95 for the sales price?

Answer: The condition will evaluate to `False` and the text “This crazy drill outputs False” is assigned to the output text box.

Chapter 4 – Decision Making

Example: Improved Voting Booth

Problem Description

Our previous Voting Booth application allowed for the counting of votes for three candidates and a count of the number of incorrect votes.

If this system were used in the real world, you would have a great number of incorrect votes that were really meant to be a vote for one of the three candidates.

Since you checked only the spelling for each name, what do you think would happen if you type Al Gore instead of Gore? The answer is that the vote would be counted as an incorrect vote.

Problem Discussion

One way to solve this problem is to use compound conditional statements to check for the additional spellings of each name.

The only modification required to the application would be the code in the `btnVote` button's `Click` event.

Chapter 4 – Decision Making

Problem Solution

Observe the modifications to the `btnVote` button that add additional spellings for each candidate.

```
Private Sub btnVote_Click(...
   If (txtVote.Text = "Bush") Or (txtVote.Text = "George Bush") Then
        intBushCount += 1
   ElseIf (txtVote.Text = "Gore") Or (txtVote.Text = "Al Gore") Then
        intGoreCount += 1
   ElseIf (txtVote.Text = "Nader") Or (txtVote.Text = "Ralph Nader") Then
        intNaderCount += 1
   Else
        intErrorCount += 1
   End If

   'Erase the vote
   txtVote.Text = ""
End Sub
```

Chapter 4 – Decision Making

4.4 Nested Conditional Statements

Compound conditional statements are useful for mapping real-world situations to the computer.

If a part of the condition needs to be repeated more than once, it would be inefficient to repeat the check of that condition each time.

Visual Basic .NET provides the ability to **nest conditional statements**.

It is simply a matter of placing one conditional statement inside another.

This simply requires treating the inner `If` statement as an individual `If` statement to be evaluated as you would any other statement.

Here is a real-world example of when this would be useful:

Chapter 4 – Decision Making

Nested If Statements

The following code loosely implements the previous flowchart. It will not ask the questions depicted, it will process the answers to the three questions as if they were asked as portrayed in the flowchart.

The code assumes that a `btnCompoundConditional` button has been created to place the code and that three text boxes – `txtQuestion1`, `txtQuestion2`, and `txtOutput` were created to hold the input and output.

```
Private Sub btnCompoundConditional_Click(...
    If (txtQuestion1.Text = "Yes") Then
       If (txtQuestion2.Text = "Yes") Then
           txtOutput.Text = "Basketball"
       Else
           txtOutput.Text = "Hockey"
       End If
    Else
       If (txtQuestion2.Text = "Yes") Then
           txtOutput.Text = "Opera"
       Else
           txtOutput.Text = "Philharmonic"
       End If
    End If
End Sub
```

Chapter 4 – Decision Making

Nested If Statements Continued

What do you think would be contained in `txtOutput`:

1 If the user enters `"Yes"` in `txtQuestion1` and `"Yes"` in `txtQuestion2`?

Answer: The text “Basketball” is placed in the text box.

2 If the user enters `"Yes"` in `txtQuestion1` and `"No"` in `txtQuestion2`?

Answer: The text “Hockey” is placed in the text box.

3 If the user enters `"No"` in `txtQuestion1` and `"Yes"` in `txtQuestion2`?

Answer: The text “Opera” is placed in the text box.

4 If the user enters `"No"` in `txtQuestion1` and `"No"` in `txtQuestion2`?

Answer: The text “Philharmonic” is placed in the text box.

Chapter 4 – Decision Making

Drill 4.12

Assume that a `btnCompoundConditional` button has been created to place the code and that two text boxes, `txtInput` and `txtOutput` were created to hold the input and output.

```
Private Sub btnCompoundConditional_Click(...
    Dim intDrillValue As Integer
    intDrillValue = Val(txtInput.Text)
    If (intDrillValue = 1) Then
       If (intDrillValue <= 1) Then
           txtOutput.Text = "This will output, from the 1st Inner If"
       Else
           txtOutput.Text = "This will output, from the 1st Inner Else"
       End If
    Else
       If (intDrillValue < 1) Then
           txtOutput.Text = "This will output, from the 2nd Inner If"
       Else
           txtOutput.Text = "This will output, from the 2nd Inner Else"
       End If
    End If
End Sub
```

Drill 4.12 Continued

What do you think would be contained in `txtOutput`:

1 If the user enters `0` in `txtInput`?

Answer: The text “This will output, from the 2nd Inner If” is placed in the text box.

2 If the user enters `1` in `txtInput`?

Answer: The text “This will output, from the 1st Inner If” is placed in the text box.

3 If the user enters `2` in `txtInput`?

Answer: The text “This will output, from the 2nd Inner Else” is placed in the text box.

Chapter 4 – Decision Making

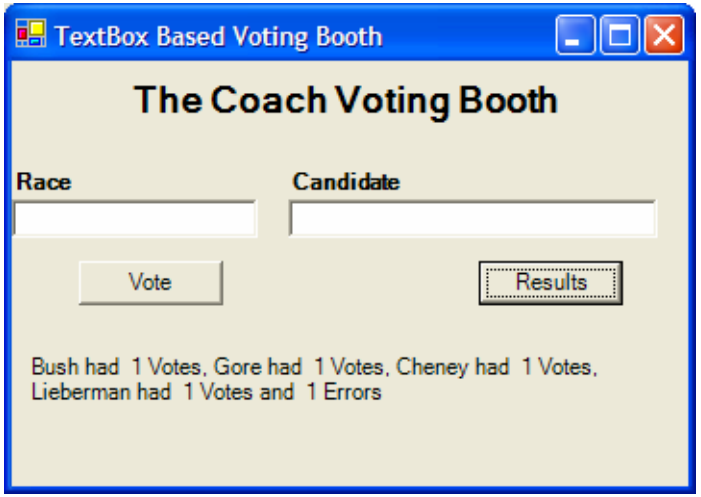
Example: Improved Voting Booth

Problem Description

Imagine if instead of writing a Voting Booth application for a single presidential race, you needed to develop a Voting Booth application that could be used for additional races as well.

Change your current application to count votes for the presidential and vice presidential elections.

For simplicity's sake, you will limit the candidates to George Bush and Al Gore for the presidency and Dick Cheney and Joe Lieberman for the vice presidency.



Chapter 4 – Decision Making

Problem Discussion

You will need a variable for each candidate to track the number of valid votes that they receive.

You will also keep a single variable to track all of the improperly cast votes.

Additionally, you will need to modify the results to display the additional candidates and modify the processing of the votes to handle the new race text box as well as the additional candidates.

Chapter 4 – Decision Making

Problem Solution

The code required for the additional variables needs to be declared in the `Declarations` section of the form.

```
Dim intBushCount As Integer
Dim intGoreCount As Integer
Dim intCheneyCount As Integer
Dim intLiebermanCount As Integer
Dim intErrorCount As Integer
```

You would need to change the code for the btnResults button so that it outputs all of the results of the election.

```
Private Sub btnResults_Click(...
   lblResults.Text = "Bush had " & intBushCount.ToString() & _
   " Votes, Gore had " & intGoreCount.ToString() & _
   " Votes, Cheney had " & intCheneyCount.ToString() & _
   " Votes, Lieberman had " & intLiebermanCount.ToString() & " Votes" & _
   " and " & intErrorCount.ToString() & " Errors"
End Sub
```

Problem Solution Continued

If you didn't nest the conditional statements, your code would execute more slowly.

When a condition is repeatedly checked, consider using the nested form.

Each time you check a candidate which the nonnested example, you have to recheck the condition to indicate whether this vote is for a president or a vice president.

Chapter 4 – Decision Making

Problem Solution Continued

Here is correct code for the `btnVote Click` even:

```
Private Sub btnVote_Click(...
  If (txtRace.Text = "Pres") Then
     If (txtVote.Text = "Bush") Or (txtVote.Text = "George Bush") Then
         intBushCount += 1
     ElseIf (txtVote.Text = "Gore") Or (txtVote.Text = "Al Gore") Then
         intGoreCount += 1
     Else
         intErrorCount += 1
     End If
  ElseIf (txtRace.Text = "Vice") Then
     If (txtVote.Text = "Cheney") Or (txtVote.Text = "Dick Cheney") Then
         intCheneyCount += 1
     ElseIf (txtVote.Text = "Lieberman") Or
              (txtVote.Text = "Joe Lieberman") Then
         intLiebermanCount += 1
     Else
         intErrorCount += 1
     End If
  Else
     intErrorCount += 1
  End If

  'Erase the vote
  txtVote.Text = ""
  txtRace.Text = ""
End Sub
```

Chapter 4 – Decision Making

Problem Solution Continued

Here is incorrect code for the `btnVote Click` even:

```
Private Sub btnVote_Click(...
   If (txtRace.Text = "Pres") And _
      ((txtVote.Text = "Bush") Or (txtVote.Text = "George Bush")) Then
        intBushCount += 1
   ElseIf (txtRace.Text = "Pres") And _
      ((txtVote.Text = "Gore") Or (txtVote.Text = "Al Gore")) Then
        intGoreCount += 1
   ElseIf (txtRace.Text = "Vice") And _
      ((txtVote.Text = "Cheney") Or (txtVote.Text = "Dick Cheney")) Then
        intCheneyCount += 1
   ElseIf (txtRace.Text = "Vice") And _
      ((txtVote.Text = "Lieberman") Or _
       (txtVote.Text = "Joe Lieberman")) Then
        intLiebermanCount += 1
   Else
        intErrorCount += 1
   End If

   'Erase the vote
   txtVote.Text = ""
   txtRace.Text = ""
End Sub
```

Chapter 4 – Decision Making

4.5 Select Case Statements

As your applications become more complex, you may have many conditions to check.

Using multiple `If`, `ElseIf`, and `Else` statements can become burdensome.

A `Select Case` statement gives the programmer the ability to shortcut the process of describing under what conditions certain code should be executed.

```
Select Case Expression
     Case Possible Value or Range of Values
          Statement(s)
     Case Another Possible Value or Range of Values
          Statement(s)
     .
     .
     .

     Case Else
          Statement(s)
End Select
```

Chapter 4 – Decision Making

The expression in a `Select Case` statement may be

- a numeric variable
- a string variable
- a simple expression composed of operators and variables

The possible values in a `Case` statement may be

- a numeric constant
- a string constant
- a numeric variable
- a string variable
- a range of values
- a combination of the above

Chapter 4 – Decision Making

Select Case Statement with Numeric Values

You can use a `Select Case` statement in a program in the same way as conditional statements.

The following code shows the use of a `Select Case` statement to demonstrate how many dozens of roses are being ordered.

The code assumes that a button, `btnSelectCase`, has been created to contain the code.

The text boxes `txtInput` and `txtOutput` have been created to hold the input and output, respectively.

```
Private Sub btnSelectCase_Click(...
   Dim intExampleValue As Integer

   intExampleValue = Val(txtInput.Text)

   Select Case intExampleValue
     Case 12
        txtOutput.Text = "Your order of a dozen roses has been placed"
     Case 24
        txtOutput.Text = "Your order of two dozen roses has been placed"
     Case Else
        txtOutput.Text = "You must order either one or two dozen roses"
   End Select

End Sub
```

Select Case Statement with Numeric Values Continued

What do you think would be contained in `txtOutput`:

1 If the user enters `12` in `txtInput`?

Answer: The text “Your order of a dozen roses has been placed” is placed in the text box.

2 If the user enters `24` in `txtInput`?

Answer: The text “Your order of two dozen roses has been placed” is placed in the text box.

3 If the user enters `0` in `txtInput`?

Answer: The text “You must order either one or two dozen roses” is placed in the text box.

Chapter 4 – Decision Making

Select Case Statement with String Values

`Select Case` statements can also be used with `Strings`.

The following code shows the use of `Strings`.

```
Private Sub btnSelectCase_Click(...
    Select Case txtPlayer.Text
      Case "Allen Iverson"
         txtOutput.Text = "Iverson Rules the NBA"
      Case "Theo Ratliff"
         txtOutput.Text = "Ratliff is the ultimate shot blocker"
      Case Else
         txtOutput.Text = "Try again"
    End Select
End Sub
```

Chapter 4 – Decision Making

Select Case Statement with Numeric Values Continued

What do you think would be contained in `txtOutput`:

1 If the user enters `"Allen Iverson"` in `txtInput`?

Answer: The text “Iverson Rules the NBA” is placed in the text box.

2 If the user enters `"Theo Ratliff"` in `txtInput`?

Answer: The text “Ratliff is the ultimate shot blocker” is placed in the text box.

3 If the user enters `"Michael Jordan"` in `txtInput`?

Answer: The text “Try again” is placed in the text box.

Chapter 4 – Decision Making

Select Case Statement with Multiple String Values

One great feature of a `Select Case` statement is the ability to indicate a `Case` as a series of `Strings` to compare against.

If you wish the same code to execute for more than one `String`, simply list them one after another separated by commas.

```
Select Case VariableToTestAgainst
    Case "FirstString", "SecondString", "ThirdString"
        txtOutput.Text = "1st Output"
    Case "FourthString", "FifthString", "SixthString"
        txtOutput.Text = "2nd Output"
    .
    .
    .

    Case Else
        txtOutput.Text = "String Not Found"
End Select
```

Chapter 4 – Decision Making

Select Case Statement with Multiple String Values Continued

Following is a simple example demonstrating how you can check for which sport an athlete plays.

It takes advantage of the use of multiple `Strings` in a `Select Case` statement to simplify the code and assumes a text box `txtAthlete` has been created.

```
Select Case txtAthlete.Text
    Case "Serena Williams", "Martina Hingis", "Anna Kournikova"
        txtOutput.Text = "Tennis"
    Case "Sheryl Swoopes", "Katie Smith", "Brandy Reed"
        txtOutput.Text = "Basketball"
    Case "Marion Jones", "Michelle Kwan"
        txtOutput.Text = "Olympics"
    Case Else
        txtOutput.Text = "Some Other Event"
End Select
```

Chapter 4 – Decision Making

Select Case Statement with Multiple String Values Continued

What do you think would be contained in `txtOutput`:

1 If the user enters `"Serena Williams"` in `txtInput`?

Answer: The text “Tennis” is placed in the text box.

2 If the user enters `"Katie Smith"` in `txtInput`?

Answer: The text “Basketball” is placed in the text box.

3 If the user enters `"Michael Jordan"` in `txtInput`?

Answer: The text “Some Other Event” is placed in the text box.

Chapter 4 – Decision Making

Select Case Statement with a Range of Values

`Select Case` statements can also be used with multiple values in each `Case` statement. The following code shows the use of a compound conditional expression and assumes a `txtPoints` text box has been created.

```
Private Sub btnSelectCase_Click()
   Dim intTotalPoints As Integer

   intTotalPoints = Val(txtPoints.Text)
   Select Case intTotalPoints
     Case 0 To 10
        txtOutput.Text = "Quite a bad night for Iverson"
     Case 11 To 20
        txtOutput.Text = "Allen should be able to do better"
     Case 21 To 30
        txtOutput.Text = "Not too shabby"
     Case Is > 30
        txtOutput.Text = "He shoots, he scores!"
     Case Else
        txtOutput.Text = "Error in Input"
     End Select
End Sub
```

Chapter 4 – Decision Making

Select Case Statement with a Range of Values Continued

What do you think would be contained in `txtOutput`:

1 If the user enters `0` in `txtInput`?

Answer: The text “Quite a bad night for Iverson” is placed in the text box.

2 If the user enters `15` in `txtInput`?

Answer: The text “Allen should be able to do better” is placed in the text box.

3 If the user enters `30` in `txtInput`?

Answer: The text “Not too shabby” is placed in the text box.

4 If the user enters `50` in `txtInput`?

Answer: The text “He shoots, he scores!” is placed in the text box.

5 If the user enters `-5` in `txtInput`?

Answer: The text “Error in Input” is placed in the text box.

Chapter 4 – Decision Making

Drill 4.13

The following code assumes that a button, `btnSelectCase`, has been created to contain the code. Additionally, the text boxes `txtInput` and `txtOutput` have been created to hold the input and output, respectively, of the user.

```
Private Sub btnSelectCase_Click(...

   Dim intDrillValue As Integer
   intDrillValue = Val(txtInput.Text)

   Select Case intDrillValue
      Case Is < 0
           txtOutput.Text = "Error in Input"
      Case 0 To 20
           txtOutput.Text = "2nd Case Statement"
      Case 21 To 30
           txtOutput.Text = "3rd Case Statement"
      Case 31 To 50
           txtOutput.Text = "4th Case Statement"
      Case Is > 50
           txtOutput.Text = "5th Case Statement"
      Case Else
           txtOutput.Text = "Can I get here?"
   End Select
End Sub
```

Drill 4.13 Continued

What do you think would be contained in `txtOutput`:

1 If the user enters `0` in `txtInput`?

Answer: The text “2nd Case Statement” is placed in the text box.

2 If the user enters `100` in `txtInput`?

Answer: The text “5th Cast Statement” is placed in the text box.

3 If the user enters `-50` in `txtInput`?

Answer: The text “Error in Input” is placed in the text box.

4 Is there any value that the user can enter that will allow the `Case Else` statement to execute?

Answer: No.

Chapter 4 – Decision Making

Example: Improved Compute Grade Application

Problem Description

The Compute Grade application from Section 4.2 determined a letter grade for a class given a numerical grade as input.

Rewrite that example but implement it using a `Select Case` statement instead of `If`, `ElseIf`, and `Else` statement.

To the user of the application it will appear that nothing has changed.

Problem Discussion

The only code that must change is in btnCompute_Click().

You can take advantage of the fact that you can list multiple String values to check against for a single case on a single line to greatly simplify the code.

Chapter 4 – Decision Making

Problem Solution

Examine the following code:

```
Private Sub btnCompute_Click(...
  Dim intGrade As Integer 'Declare temporary variable

  intGrade = Val(txtNumericGrade.Text) 'Convert user input to an Integer

  'Compute Grade
  Select Case intGrade
    Case Is >= 90
        lblLetterGrade.Text = "A"
    Case Is >= 80
        lblLetterGrade.Text = "B"
    Case Is >= 70
        lblLetterGrade.Text = "C"
    Case Is >= 60
        lblLetterGrade.Text = "D"
    Case Else
        lblLetterGrade.Text = "F"
  End Select
End Sub
```

Chapter 4 – Decision Making

4.6 Case Study

Problem Description

This case study will be a continuation of last chapter's case study to compute the payroll of four workers for a company.

You want to add the functionality to compute the pay of each worker at two different pay rates.

You will have a rate of $25/hour for workers who are in the sales department and a rate of $15/hour for workers in the processing department.

You will need a set of text box controls that allow the user to indicate a department for each employee.

Chapter 4 – Decision Making

Problem Description Continued

Here is a sample input and output of your application.

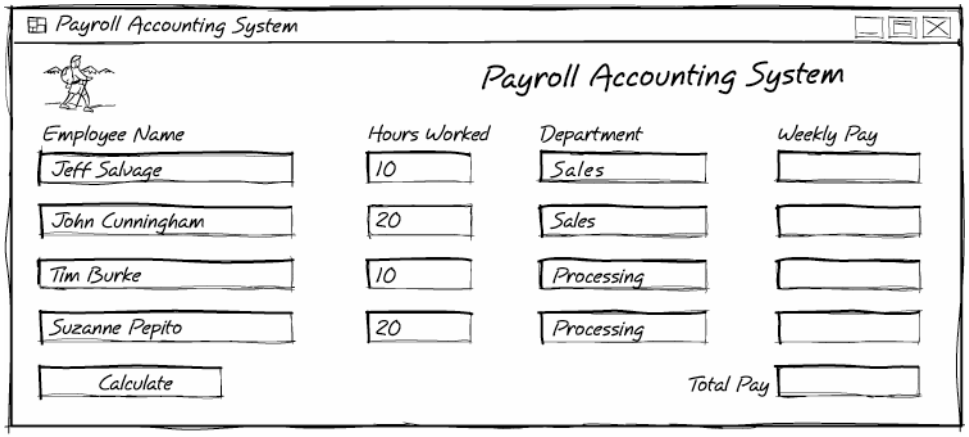


Fig 04-45

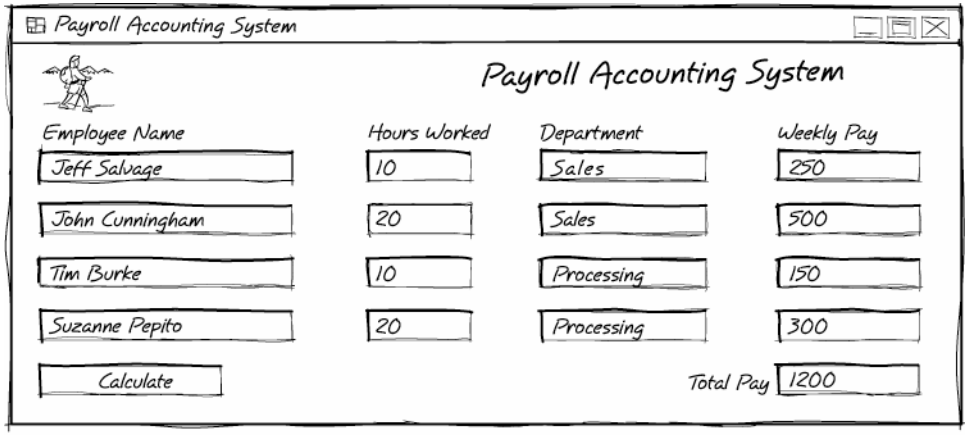


Fig 04-46

Chapter 4 – Decision Making

Problem Discussion

The solution to the problem does not change much from the previous chapter's case study.

The main difference is that you need to check which pay rate to use in the calculation of the weekly pay.

Again, most of the controls for your application were placed on the form in the previous chapter.

You need only add the controls for the department label and text boxes.

What you call the label control is unimportant.

You should call the department text boxes `txtDept1`, `txtDept2`, `txtDept3`, and `txtDept4`.

Chapter 4 – Decision Making

Problem Solution

Although it is not required, the use of constants in this solution is desirable.

You should code a constant to indicate the pay rates for the sales and processing departments.

The constant for the sales department and processing department pay rates will be called `decSalesPayRate` and `decProcessingPayRate`, respectively.

This way you can change either pay rate once and have it affect the entire application.

To set the constant, perform the following steps:

Step 1: Right-click the mouse button and click on `View Code`.

Step 2: Select the `Declarations` area of code.

Step 3: Type "`Const decSalesPayRate As Decimal = 25`".

Step 4: Type "`Const decProcessingPayRate As Decimal = 15`".

Your code should look like this:

```
Const decSalesPayRate As Decimal = 25
Const decProcessingPayRate As Decimal = 15
```

Chapter 4 – Decision Making

Problem Solution Continued

The `btnCalculate` button's `Click` event code must set each weekly pay's value to the number of hours worked multiplied by the pay rate associated with each employee's department. The code is shown here:

```
Private Sub btnCalculate_Click(...
    'Temporary Variables to Store Calculations
    Dim decTotalPay As Decimal
    Dim decWeeklyPay As Decimal

    'First Person's Calculations
    If (txtDept1.Text = "Sales") Then
       decWeeklyPay = decSalesPayRate * Val(txtHours1.Text)
    ElseIf (txtDept1.Text = "Processing") Then
       decWeeklyPay = decProcessingPayRate * Val(txtHours1.Text)
    Else
       decWeeklyPay = 0
    End If
    txtWeeklyPay1.Text = decWeeklyPay.ToString
    decTotalPay = decWeeklyPay

    'Second Person's Calculations
    If (txtDept2.Text = "Sales") Then
       decWeeklyPay = decSalesPayRate * Val(txtHours2.Text)
    ElseIf (txtDept2.Text = "Processing") Then
       decWeeklyPay = decProcessingPayRate * Val(txtHours2.Text)
    Else
       decWeeklyPay = 0
    End If
    txtWeeklyPay2.Text = decWeeklyPay.ToString()
    decTotalPay += decWeeklyPay
```

Chapter 4 – Decision Making

Problem Solution Continued

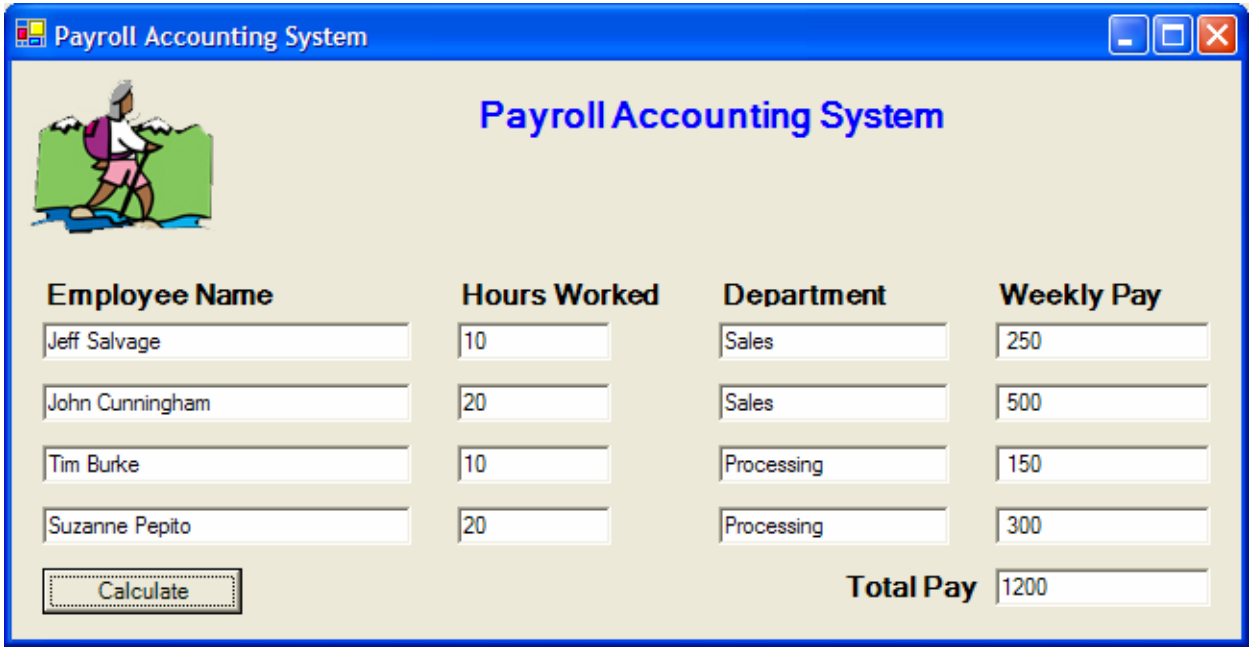
code continued:

```
    'Third Person's Calculations
    If (txtDept3.Text = "Sales") Then
        decWeeklyPay = decSalesPayRate * Val(txtHours3.Text)
    ElseIf (txtDept3.Text = "Processing") Then
        decWeeklyPay = decProcessingPayRate * Val(txtHours3.Text)
    Else
        decWeeklyPay = 0
    End If
    txtWeeklyPay3.Text = decWeeklyPay.ToString()
    decTotalPay += decWeeklyPay

    'Fourth Person's Calculations
    If (txtDept4.Text = "Sales") Then
        decWeeklyPay = decSalesPayRate * Val(txtHours4.Text)
    ElseIf (txtDept4.Text = "Processing") Then
        decWeeklyPay = decProcessingPayRate * Val(txtHours4.Text)
    Else
        decWeeklyPay = 0
    End If
    txtWeeklyPay4.Text = decWeeklyPay.ToString()
    decTotalPay += decWeeklyPay

    'Convert Total Pay to a string and copy to TextBox
    txtTotalPay.Text = decTotalPay.ToString()
End Sub
```

Chapter 4 – Decision Making

Problem Solution Continued

The final application should look like this:



Chapter 4 – Decision Making

Coach's Corner

Adding Functionality to the Message Box

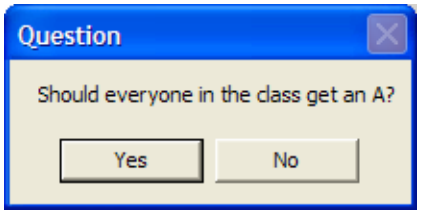
With a slight modification to the `MsgBox` command, you can ask the user a question and get an answer without having to create new forms.

If you want to ask a simple Yes/No question, you can ask it using the `MsgBox` command.

The following code will ask the question "Should everyone in the class get an A?" and store the result in the variable `intAnswer`.

```
intAnswer = MsgBox("Should everyone in the class get an A?", _
                   MsgBoxStyle.YesNo, "Question")
```

The message box would look like this:



Chapter 4 – Decision Making

By using the following constants, you can create dialog boxes with the following buttons:

YesNo	Yes/No
YesNoCancel	Yes/No/Cancel
OkCancel	Ok/Cancel
RetryCancel	Retry/Cancel

By using the following constants, you can check to see what the user's response was:

vbYes	Yes
vbNo	No
vbCancel	Cancel
vbOk	OK
vbRetry	Retry

Chapter 4 – Decision Making

Short Circuit Analysis of Conditional Statements

In Visual Basic .NET, the evaluation of conditional statements is performed using **short circuit analysis**.

A very loose definition is that the conditional statement is evaluated as long as the outcome of the conditional statement is unknown.

Once the outcome is determined, the evaluation of the conditional statement ceases.

When you use short circuit analysis, the performance of your applications increases.

Imagine if you wanted to write a conditional statement that displayed whether the average of a series of homework grades was passing or failing.

You could use code as follows:

```
If (intNumberGrades > 0) And (intGradeTotal / intNumberGrades >= 65)Then
    MsgBox("Pass")
Else
    MsgBox("Fail")
End If
```

Without short circuit evaluation, if `intNumberGrades` equals `0`, the execution of the code would cause a run-time error.

With short circuit evaluation, the second condition never evaluates and the message box displays "`Fail`".

Chapter 4 – Decision Making

Drill 4.14

Determine if the following conditions and values cause all the conditional expressions to be evaluated.

```
Dim intDrillValue As Integer
intDrillValue = 70
If ((intDrillValue >= 65) And (intDrillValue <= 75)) Then
```

Answer: Both conditions have to be evaluated and the result is True

```
Dim intDrillValue As Integer
intDrillValue = 70
If ((intDrillValue >= 65) Or (intDrillValue <= 75)) Then
```

Answer: Only the first condition is evaluated and the result is True

```
Dim intDrillValue As Integer
intDrillValue = 70
If ((intDrillValue <= 65) Or (intDrillValue >= 75)) Then
```

Answer: Both conditions must be evaluated and the result is False