Final Chapter

Web Applications

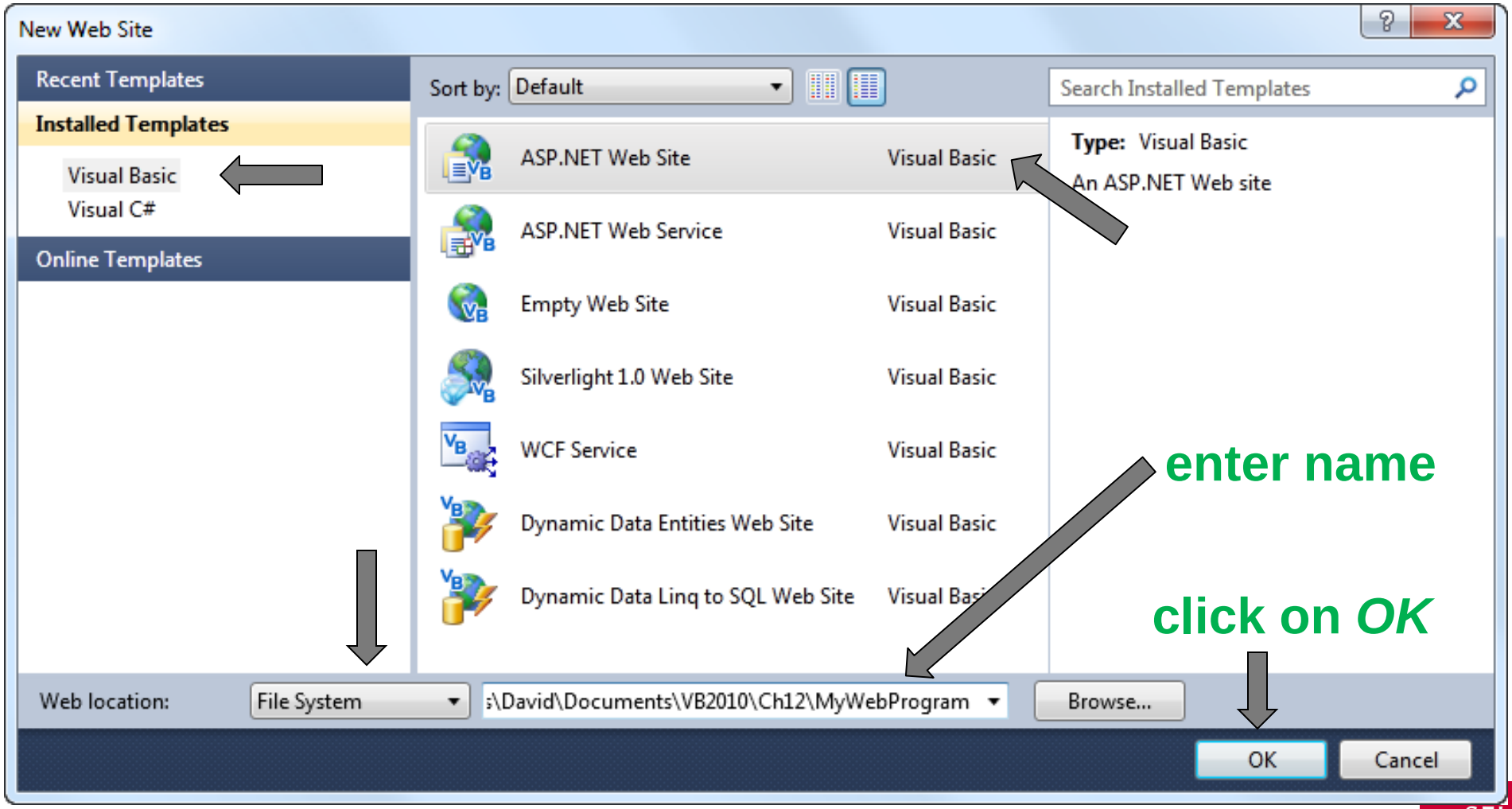
Web Programs

- The programs in this chapter require the use of either Visual Web Developer 2010 (packaged with this textbook) or the complete version of Visual Studio 2010.
- We assume that you are using one of these two software products.

Creating a Web Program

- Click on *New Web Site* in the *File* menu.
- Select *Visual Basic* in the left pane.
- Select *ASP.NET Web Site* in the middle pane.
- Select *File System* as the Web location.
- Give a name and path for the program.
- Click on the *OK* button.

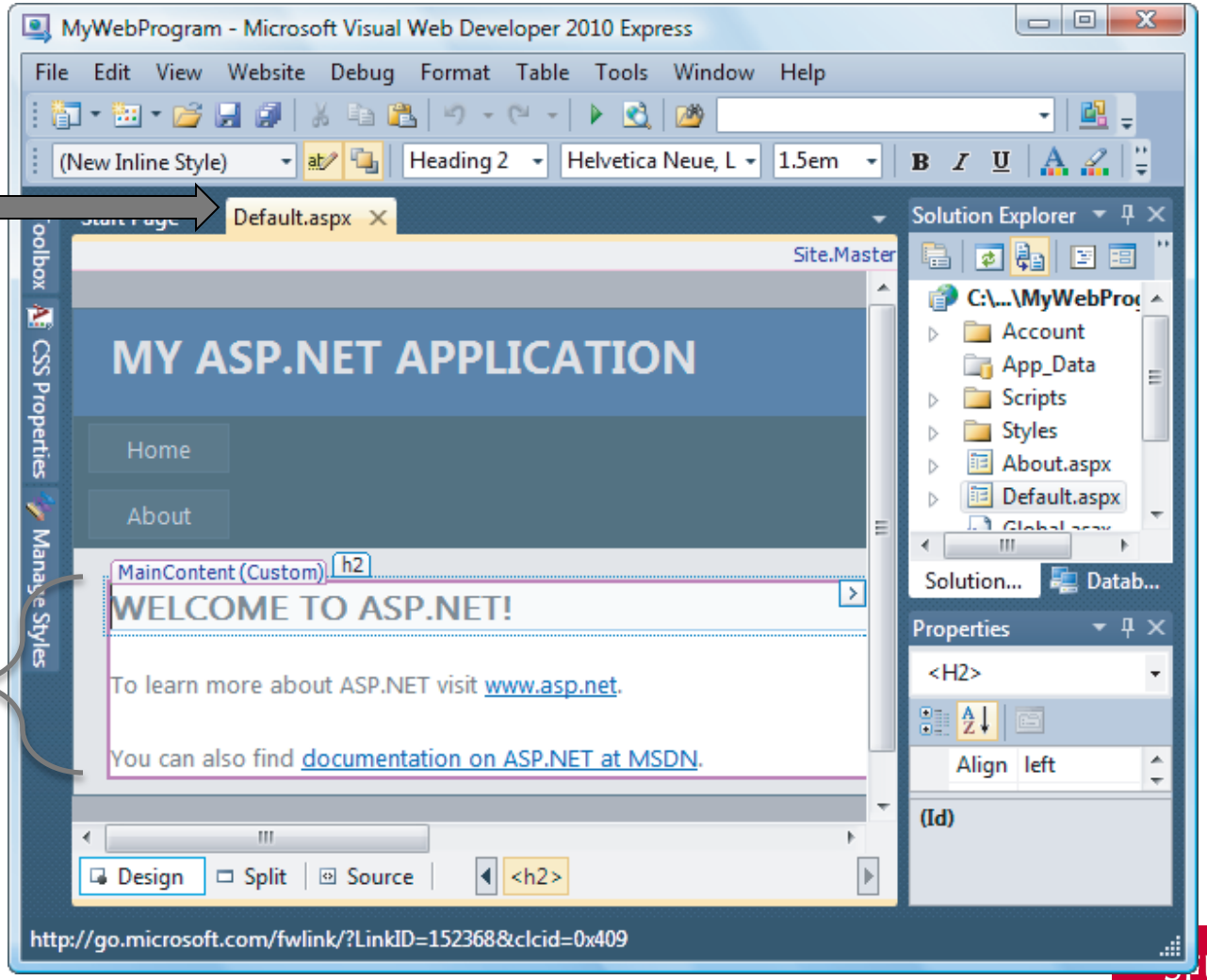
Creating a Web Program (continued)



Web Page (VWD Equivalent of the Form Designer)

Web page tab

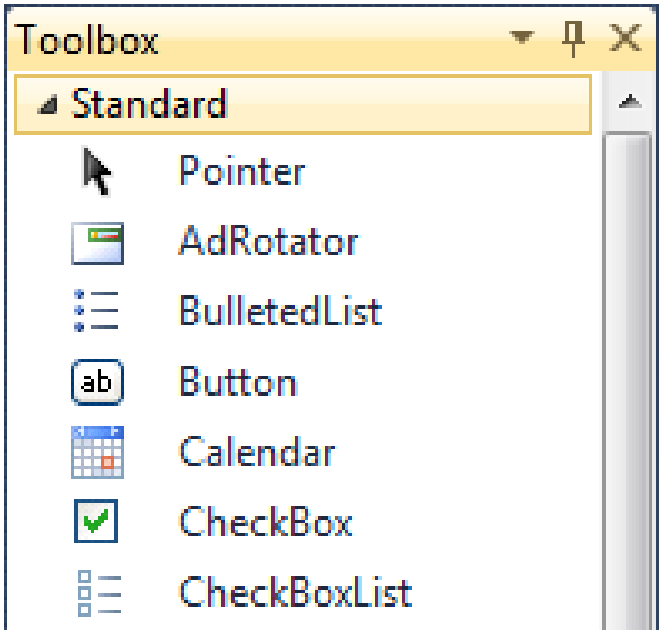
Main Content region



Web Page Tab

- The Web page tab is titled *Default.aspx* instead of *Form1. vb [Design]*
- The Web page is referred to as *Default.aspx* in the Solution Explorer window

TOOLBOX

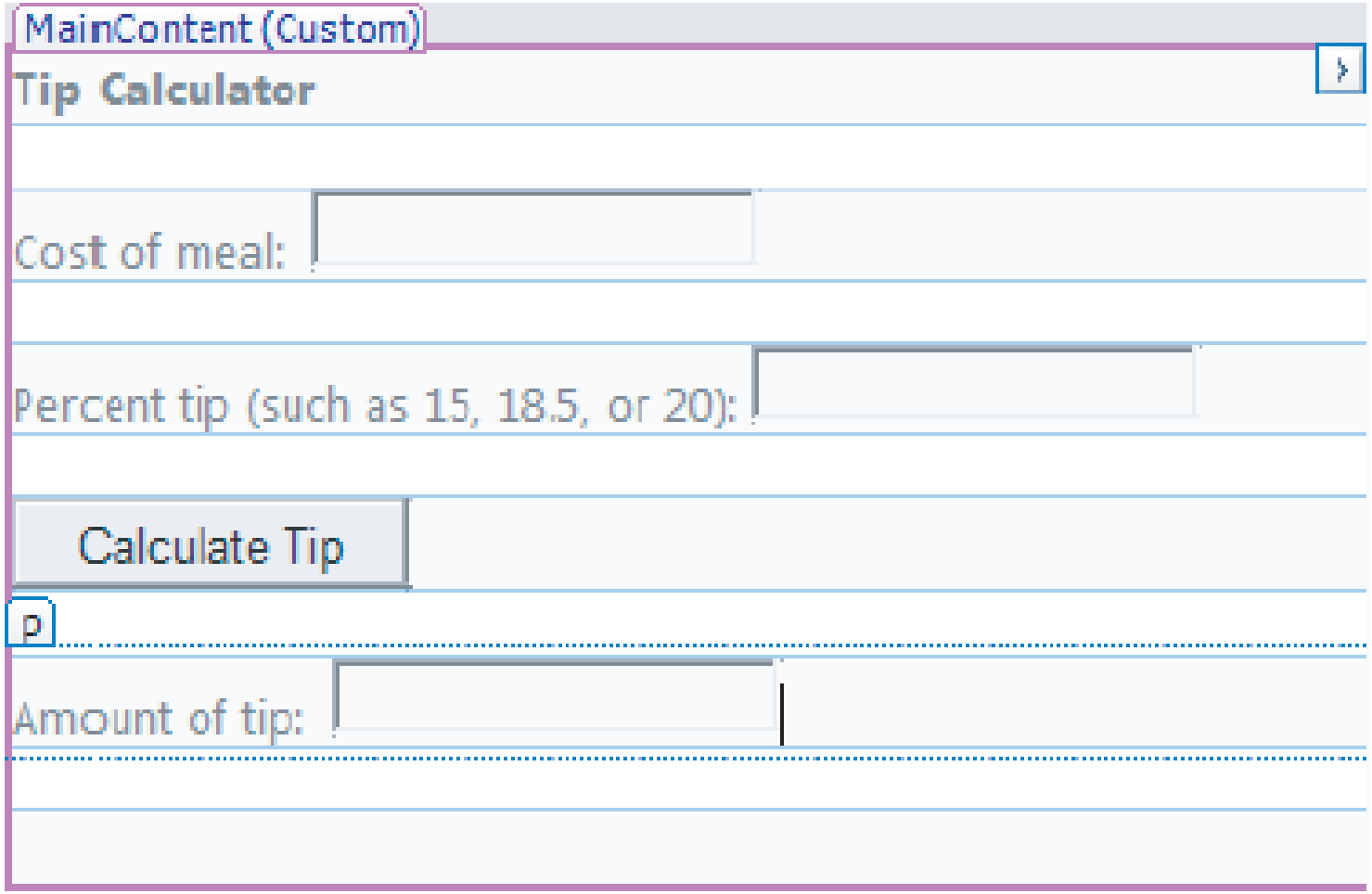


The common controls, such as button, text box, and list box are contained in the *Standard* group of the Toolbox.

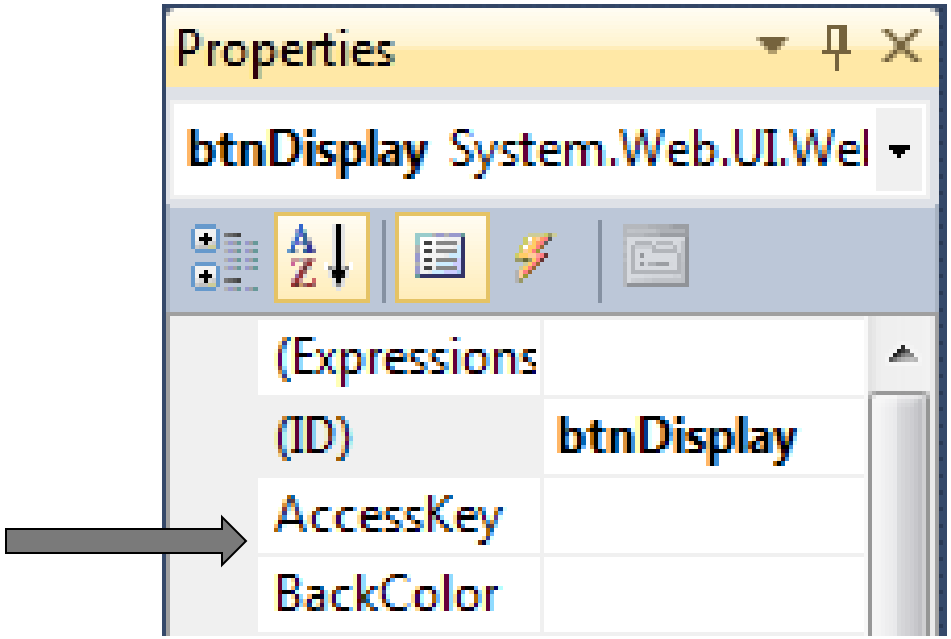
Designing the Web Page

- Begin by clearing the Main Content region
- Permanent text (called *static text*) can be typed into the page and formatted directly without the use of labels
- Text boxes and buttons can be placed at the cursor position (called the *insertion point*) by double-clicking on them in the Toolbox

Sample Web Page



Properties Window



The name of a control is specified by the *ID* property instead of the *Name* property

Code Editor

- The Code Editor tab reads *Default.aspx.vb* instead of *Form1.vb*

- The code in the editor is referred to as the *code behind*.

Summary Thus Far

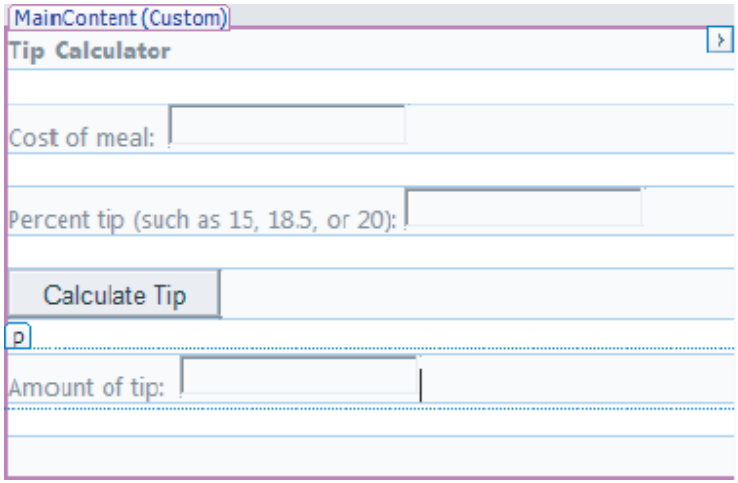
TABLE 12.2 Web page counterparts to designing a Visual Basic Windows form.

	Visual Basic Windows Form	VWD Web Page
1.	There is no cursor on the form designer.	The page designer has a cursor (also referred to as the *insertion point*).
2.	Text to appear permanently on a form is usually placed in a label and formatted with settings from the Properties window.	Permanent text (called *static text*) can be typed directly into the page at the insertion point and formatted from the Toolbar in the same way as text typed into a word processor.
3.	The most frequently used controls (such as Button, ListBox, and TextBox) are found in the *Common Controls* group of the Toolbox.	The most frequently used controls (such as Button, ListBox, and TextBox) are found in the *Standard* group of the Toolbox.
4.	When you double-click on a control in the Toolbox, the control appears either in the upper-left corner of the form or just below the most recently created control.	When you double-click on a control in the Toolbox, the control appears at the insertion point.
5.	The name of a control is specified by setting its Name property.	The name of a control is specified by setting its ID property.
6.	In design mode, controls are placed anywhere on a form and aligned with the *Format* menu.	In design mode, text and controls are placed in a top-to-bottom fashion, each entered at the insertion point.
7.	The tab for the Designer reads "*frmName*.vb [Design]".	The tab for the Designer reads "Default.aspx".

Sample Code

```
Protected Sub btnCalculate_Click(...) Handles _
  btnCalculate.Click
   Dim cost As Double = CDbl(txtCost.Text)
   Dim percent As Double =
                        CDbl(txtPercent.Text) / 100
   txtTip.Text = FormatCurrency(percent * cost)
End Sub
```

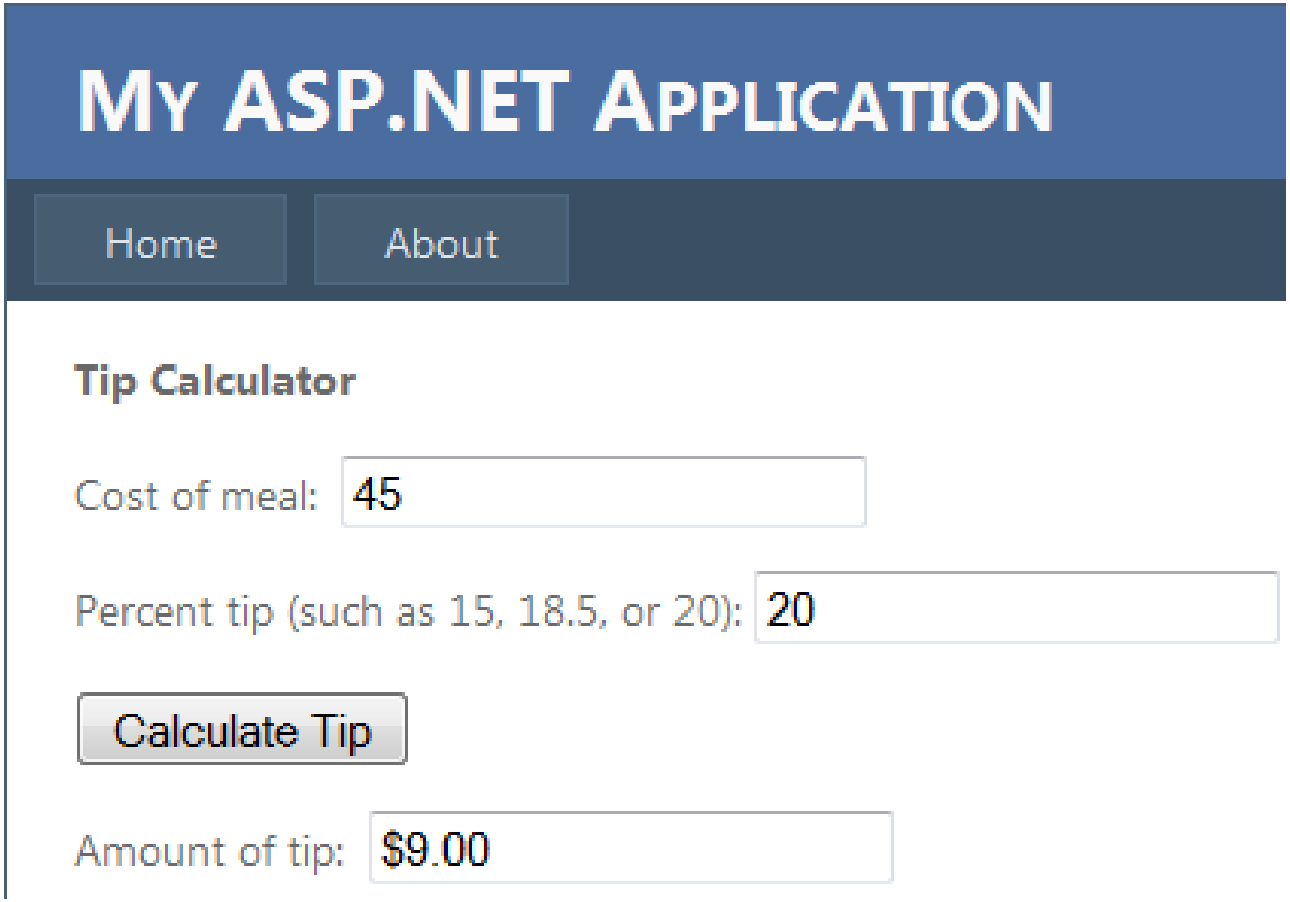
Notice that “Sub” is proceeded by “Protected” instead of “Private”.



Running a Program

- Press Ctrl+F5 to run program without debugging
- Program runs in the computer's Web browser
- To terminate the program, close the browser by clicking on  , the Close button
- Close program by clicking on *Close Project* in the *File* menu.

A Run of the Sample Program

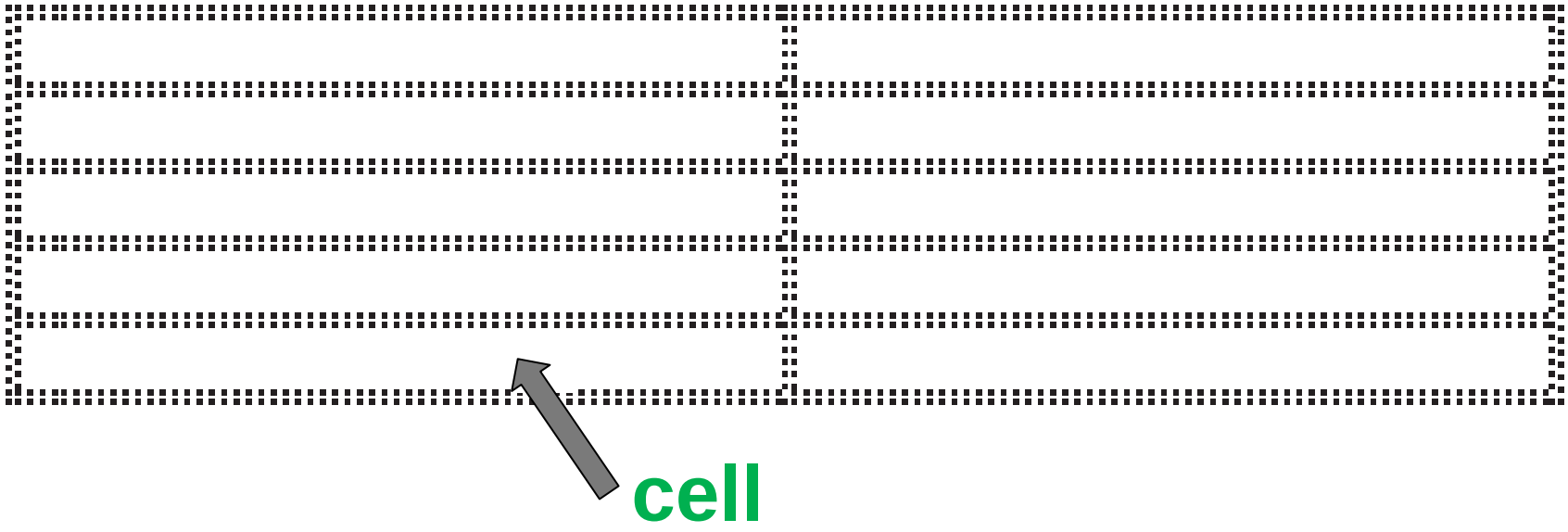


TABLES

- A **table control** can be used to improve the layout of a Web page

- Tables are created with the *Insert Table* command from the *Table* menu in the Toolbar

Sample Table



This table has 5 rows and 2 columns. Each subdivision is called a *cell*.

Cells

- Text and controls can be placed into cells
- The alignment (such as *right*, *left*, or *center*) of the contents of a cell can be specified with the Align property from the Properties window
- Commands from the *Table* menu allow you to insert and delete rows and columns, and to merge cells

Managing Tables

- Assorted arrows can be used to highlight groups of cells and resize tables

- Dragging of the cursor also can be used to highlight groups of cells

Text Files

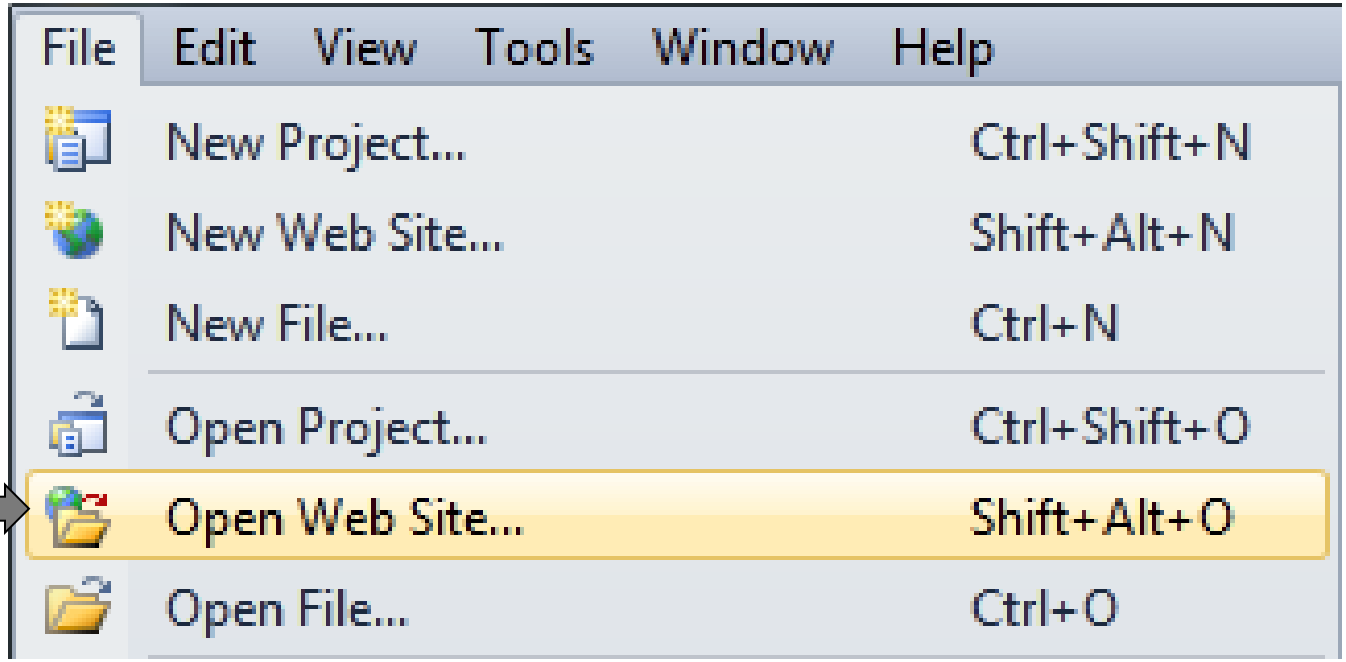
- Normally placed in the Solution Explorer's App_Data folder

- A text file can be read into an array with a statement of the form

```
Dim strArrayName() As String =
    IO.File.ReadAllLines(MapPath("App_Data\" &
                                 filename))
```

How to Open an Existing Web Program

first click here

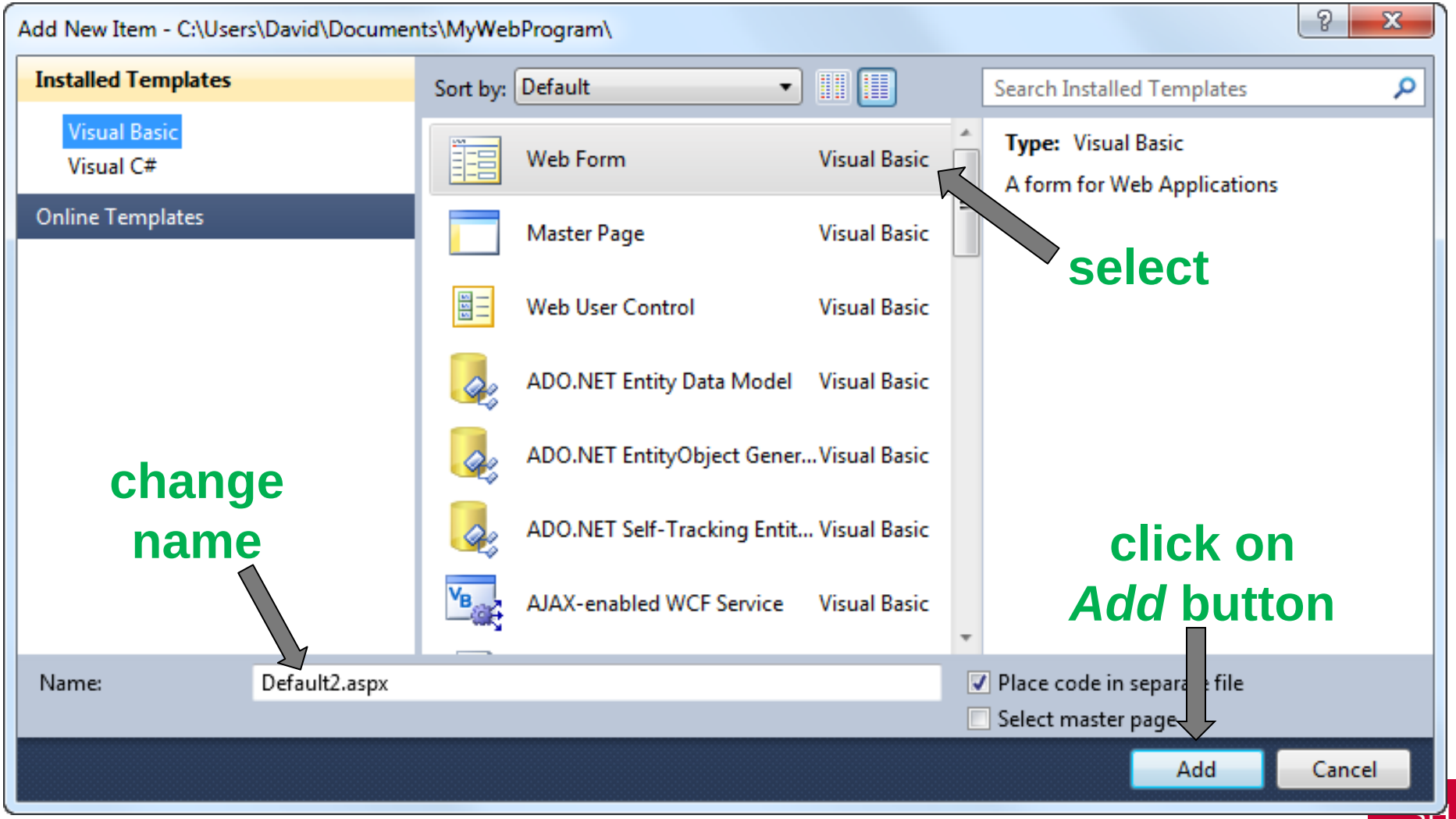


Then navigate to the program's folder and click on the *Open* button.

How to Add an Additional Web Page to a Program

- Click on an existing Web page to make sure it has the focus
- Click on *Add New Item* in the *Website* menu. (An Add New Item dialog box will appear.)
- Select *Web Form* in the center pane, type a name into the *Name* box, and click on the *Add* button.

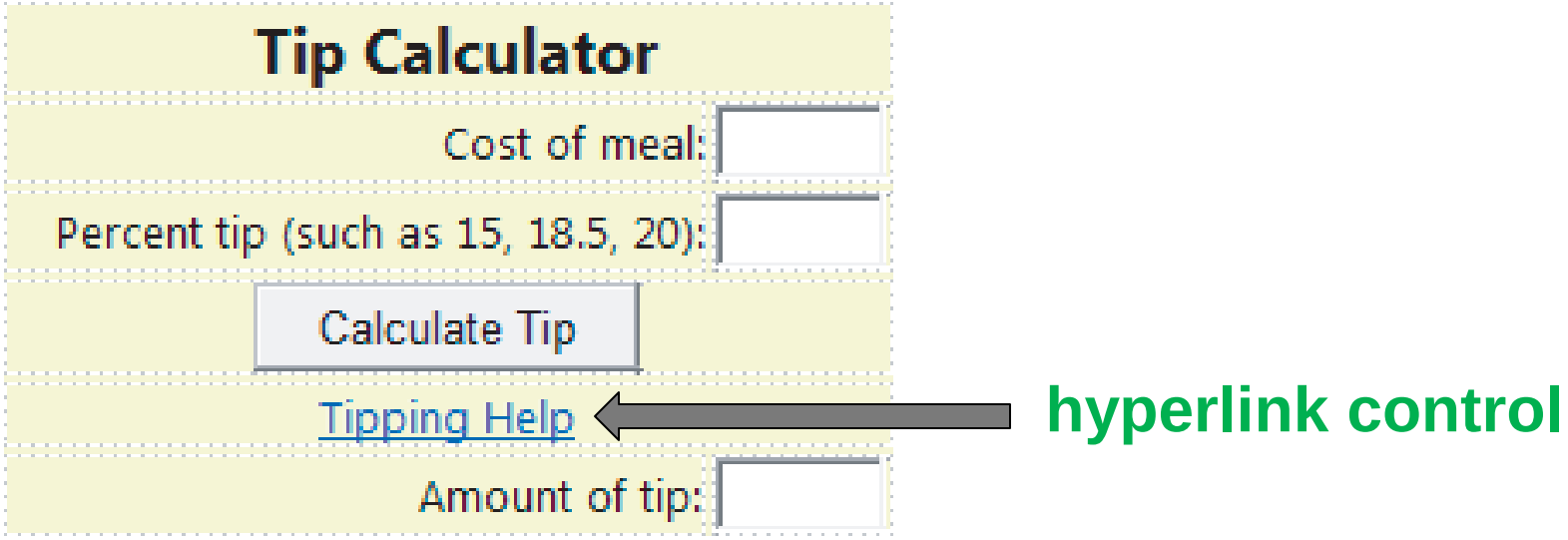
How to Add an Additional Web Page to a Program (cont.)



Hyperlink Control

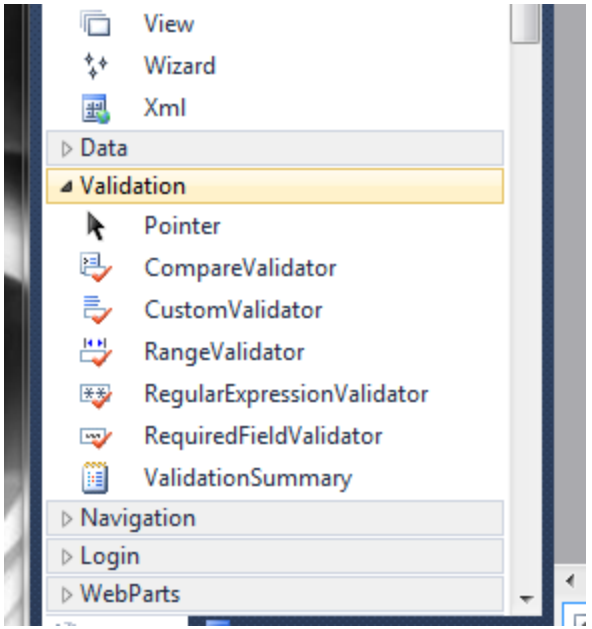
- Found in the *General* group of the Toolbox
- Appears on a page as underlined text
- Used to navigate to another page
- NavigateUrl property specifies the page to navigate to

Sample Web Page

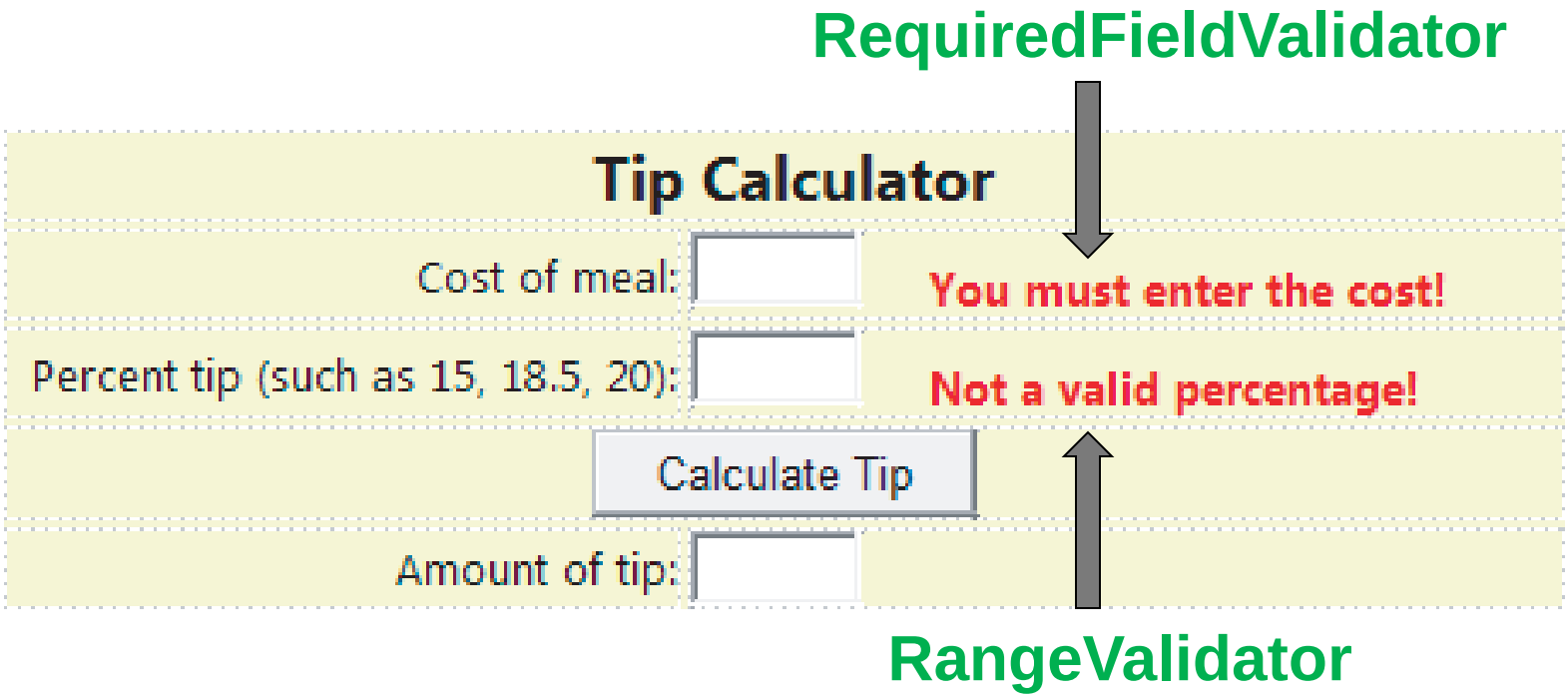


Validation Controls

- Used to validate user input
- The **RequiredFieldValidator** control checks that data has been entered into a text box or that an item of a list box has been selected
- The **RangeValidator** control checks that the entry in a text box falls within a specified range of values.



Sample Web Page

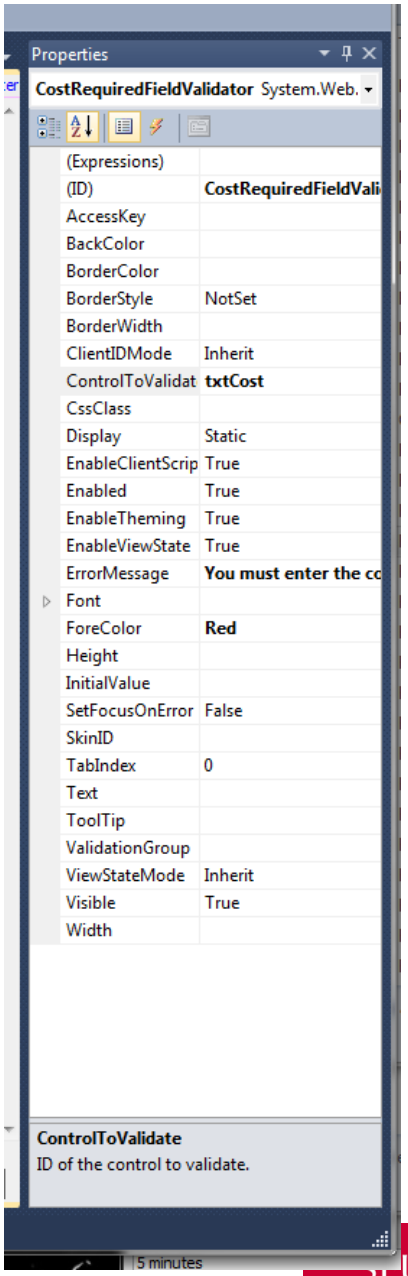


Validation controls are not visible at run time.
Only appear when input is missing or invalid.

RequiredFieldValidator Control

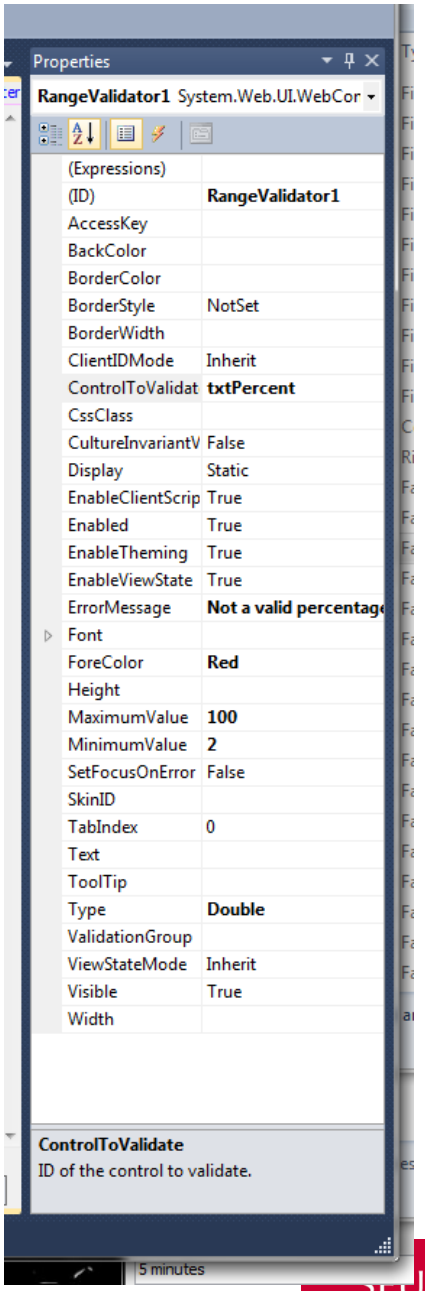
- The key properties are **ControlToVerify** and **ErrorMessage**

- The **ErrorMessage** setting is the text that appears when input into the specified control does not meet the given criteria



RangeValidator Control

- The key properties are **ControlToVerify**, **ErrorMessage**, **Type**, **MinimumValue**, and **MaximumValue**
- Possible settings for Type are *String*, *Integer*, *Double*, *Date*, and *Currency*
- The entry in the text box must lie between the *MinimumValue* and the *MaximumValue*

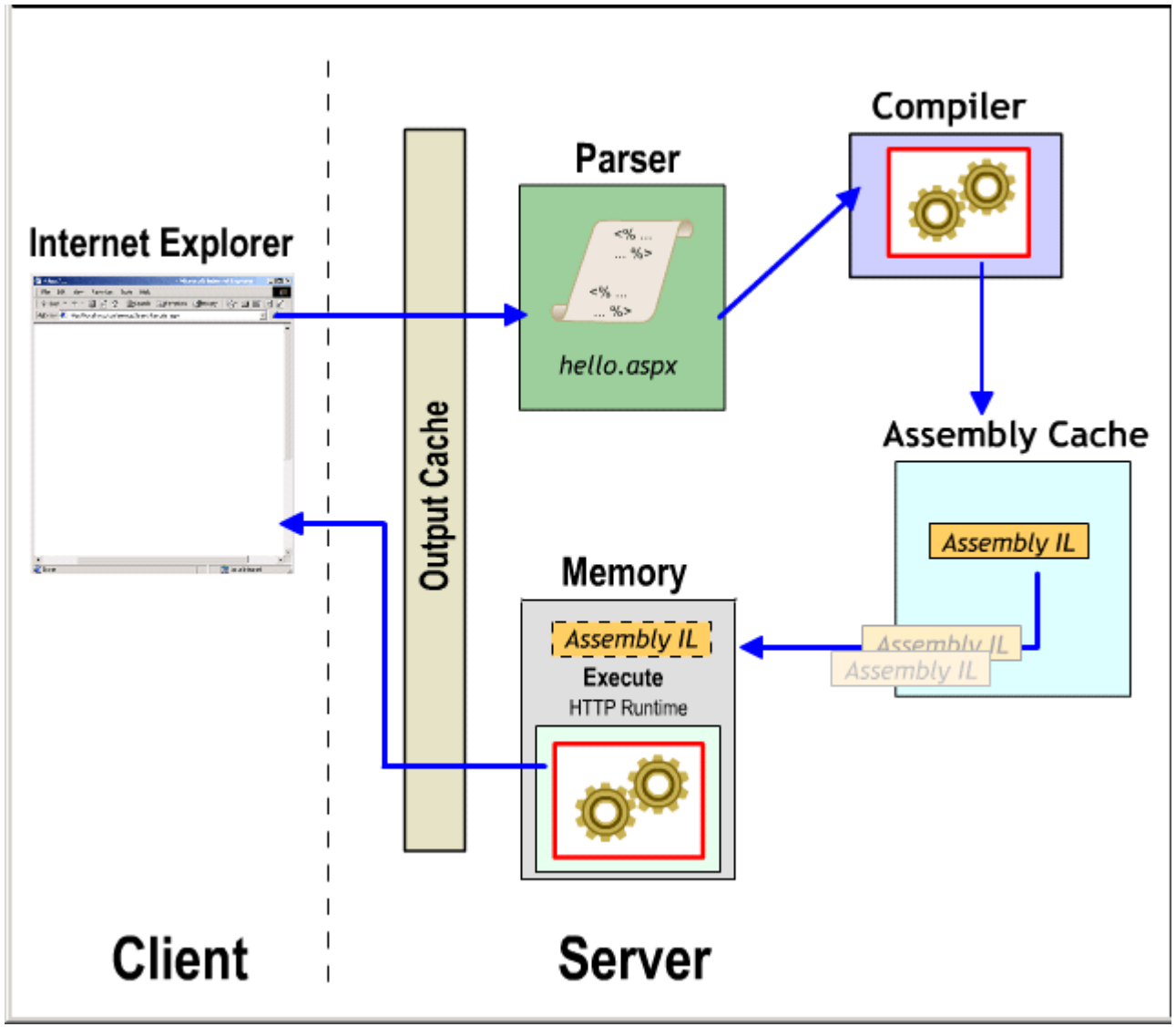


POSTBACK

- A **postback** occurs when the contents of a Web page are sent to the server for processing. Afterwards, the server sends a new page back to the browser

- When a validation control is triggered, the matter is handled entirely by the browser—no postback occurs

Programming Model



The Page Load Event

- Raised when a Web page is first loaded and every time it is reloaded after a postback

- The IsPostBack property can be used to guarantee that the page load event is raised only once

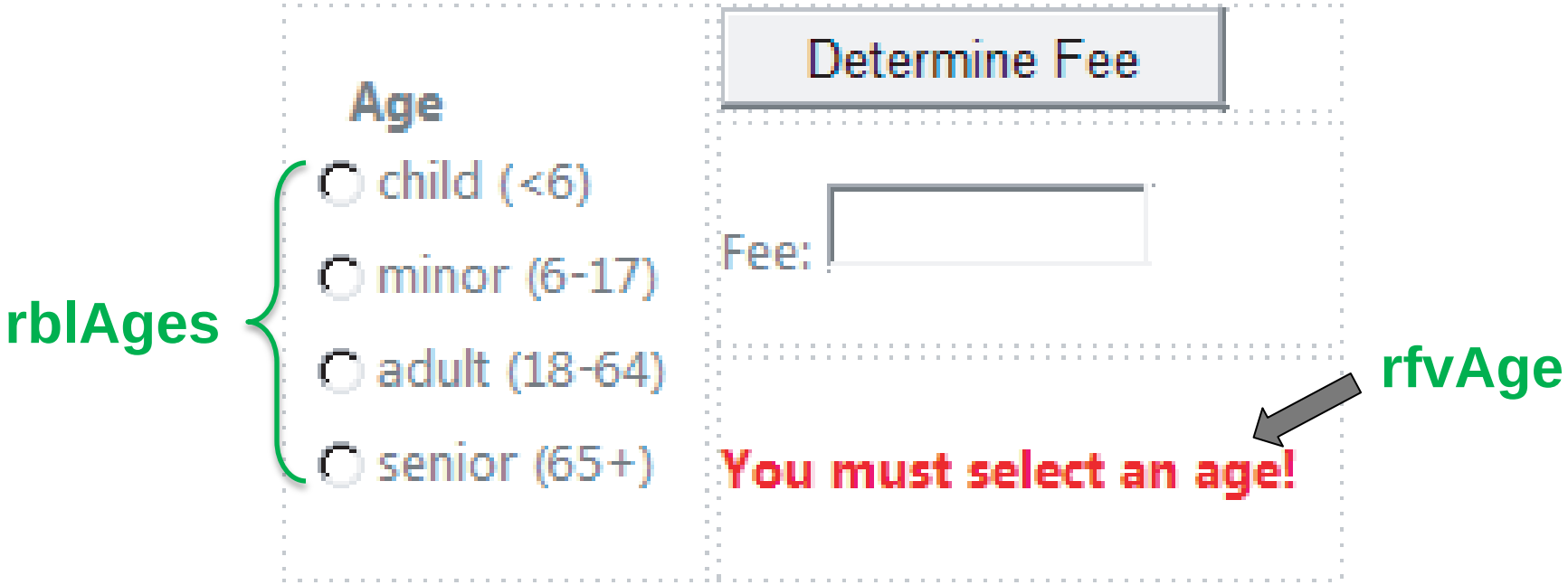
```
if Not Page.IsPostBack Then
…
End if
```

Class-Level Variables

- In VWD, class-level variables are of limited value since they do not retain their values after postbacks

- Devices known as *cookies* or *session variables* can be used to retain values

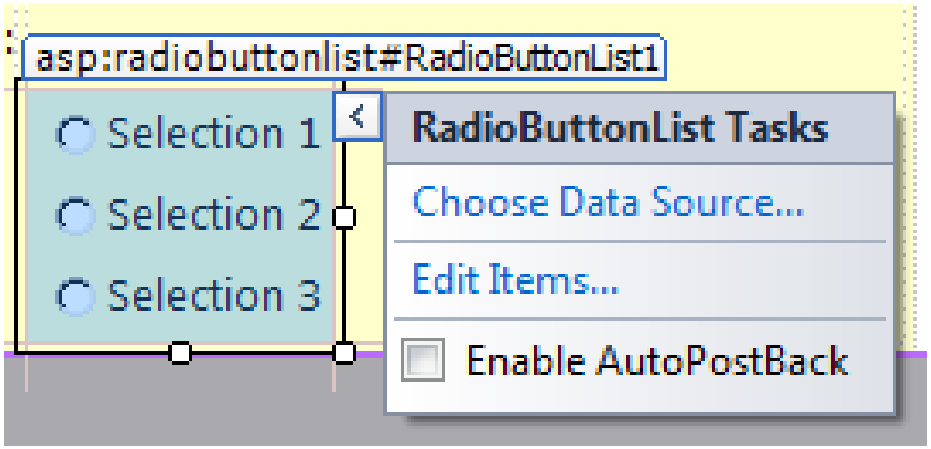
RadioButtonList Control



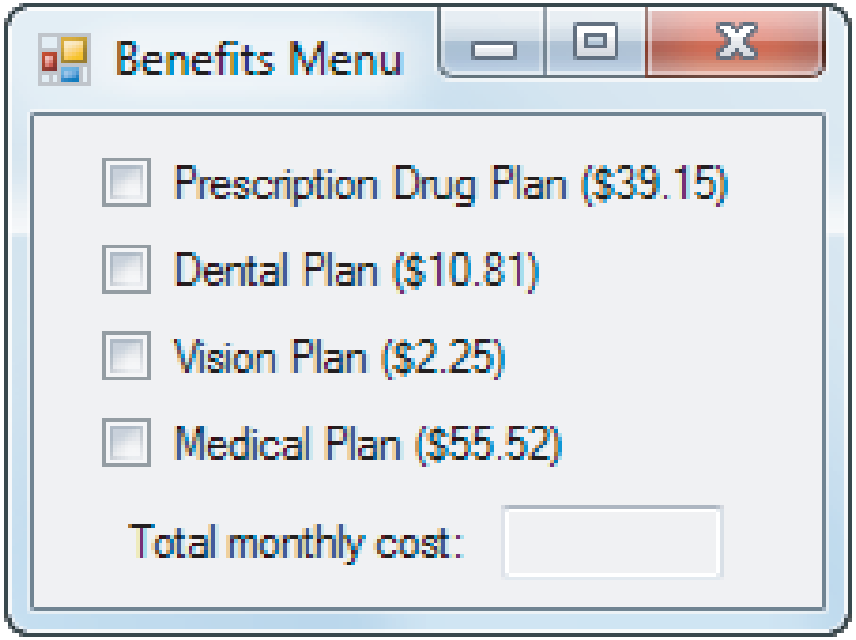
VWD does not have a group box control. The radio-button list control is the counterpart of the VB group box containing a set of radio buttons.

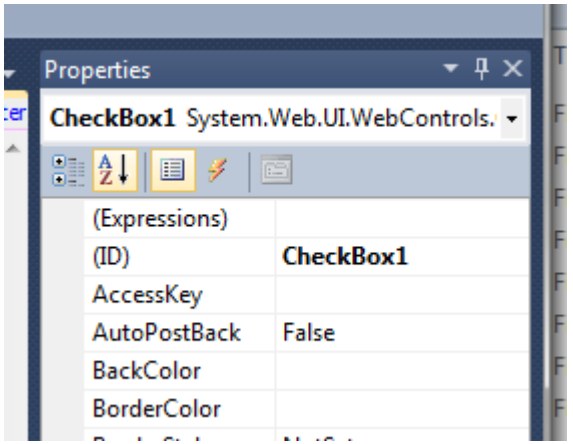
RadioButtonList Control (continued)

- The radio-button list control is populated via a ListItem Collection Editor that is invoked from the Tasks button
- In the previous slide, the control rfvAge, a RequiredFieldValidator, guarantees that a radio button has been selected before the button is clicked on.



Check Box Control





Example 5 of Section 4.4.

- In regular VB clicking on a check-box can cause code to run
- VWD does not do this
- To convert this VB program to a VWD program, the AutoPostBack property of each check box must be set to *True*

Review

Chapter 1

Communicating with the Computer

- Machine language – low level, hard for humans to understand
- Visual Basic – high level, understood by humans, consists of instructions such as Click, If, Do
  - Usable in other applications (Word, Excel…)

Computers and Complicated Tasks

- Tasks are broken down into instructions that can be expressed by a computer language
- A *program* is a sequence of instructions
- Programs can be only a few instructions or millions of lines of instructions

Performing a Task on the Computer

- Determine Output
- Identify Input
- Determine process necessary to turn given Input into desired Output

Problem-solving: approach like algebra class

- How fast is a car traveling if it goes 50 miles in 2 hours?
- **Output**:
- **Input**:
- **Process**:

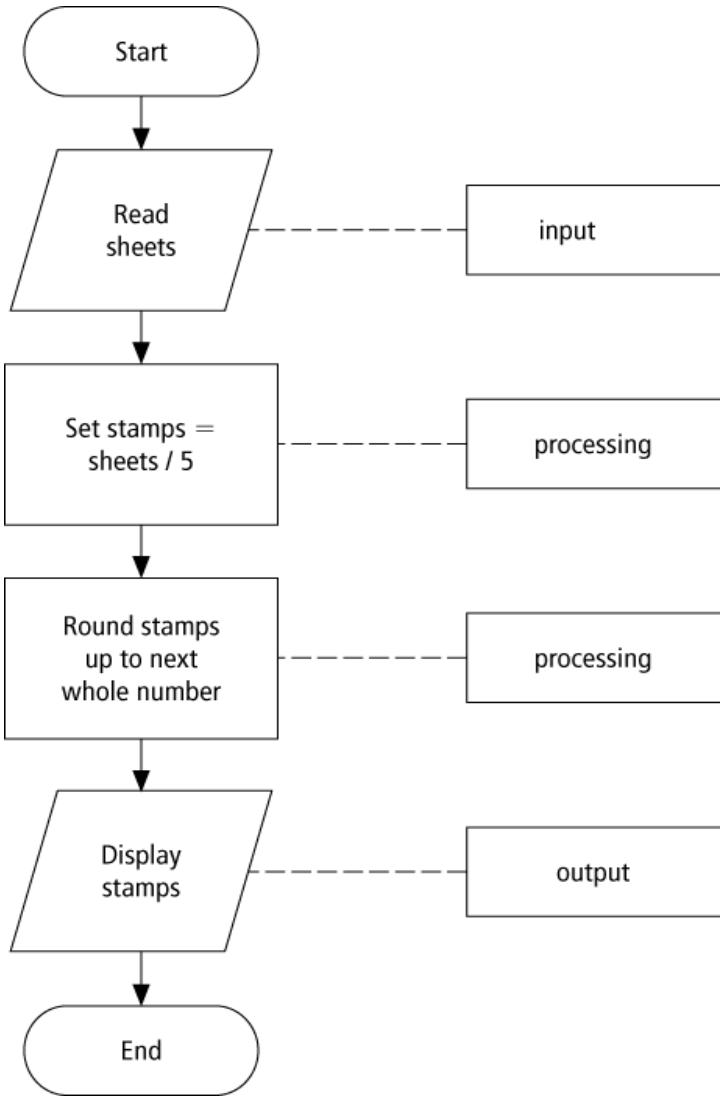
Program development cycle

1. ***Analyze*:** Define the problem.
2. ***Design*:** Plan the solution to the problem.
3. ***Choose the interface*:** Select the objects (text boxes, buttons, etc.).
4. ***Code*:** Translate the algorithm into a programming language.
5. ***Test and debug*:** Locate and remove any errors in the program.
6. ***Complete the documentation*:** Organize all the materials that describe the program.

Programming Tools

- Three tools are used to convert algorithms into computer programs:
  - **Flowchart** - Graphically depicts the logical steps to carry out a task and shows how the steps relate to each other
  - **Pseudocode** - Uses English-like phrases with some Visual Basic terms to outline the program
  - **Hierarchy chart** - Shows how the different parts of a program relate to each other

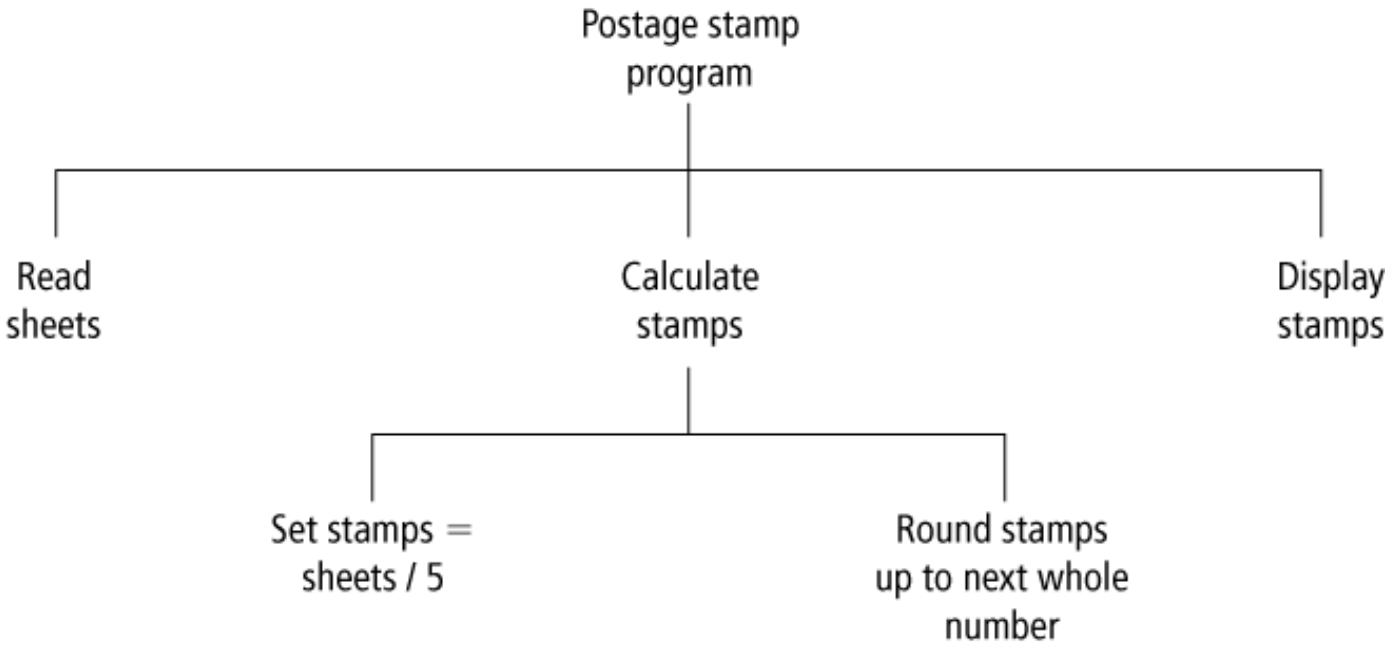
FLOWCHART EXAMPLE



Pseudocode example

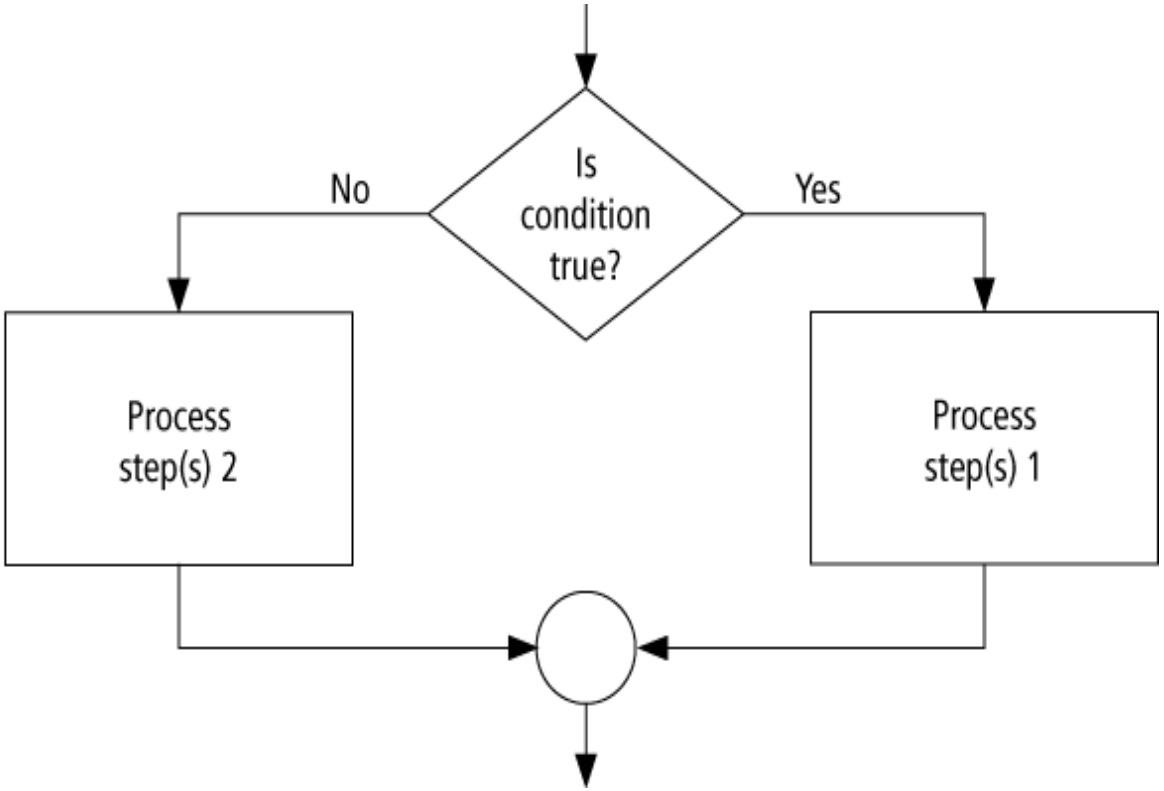
- Determine the proper number of stamps for a letter
- Read Sheets *(input)*
- Set the number of stamps to Sheets / 5 *(processing)*
- Round the number of stamps up to the next whole number *(processing)*
- Display the number of stamps *(output)*

Hierarchy charts example



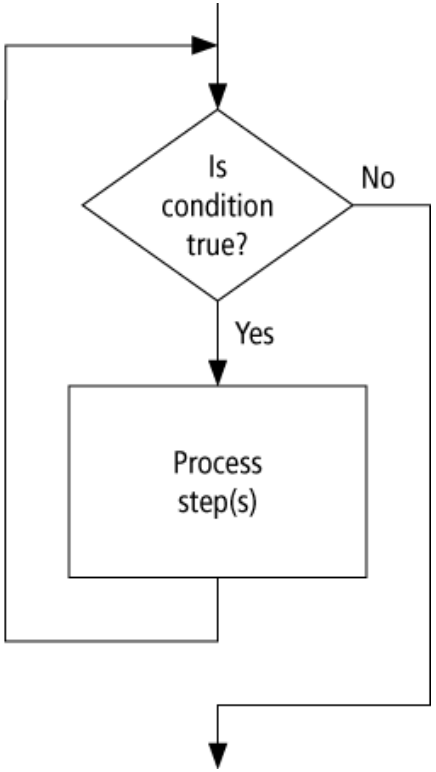
Decision flow chart

```
If condition is true Then
      Process step(s) 1
  Else
      Process step(s) 2
End If
```



Looping flow chart

```
Do While condition is true
    Process step(s)
Loop
```



Review

Chapter 2&3

Control Name Prefixes

Control	Prefix	Example
button	btn	btnCompute
label	lbl	lblAddress
text box	txt	txtAddress
list box	lst	lstOutput

Event

- An **event** is an action, such as the user clicking on a button
- Usually, nothing happens in a Visual Basic program until the user does something and generates an event.
- What happens is determined by statements.

Three Types of Errors

- Syntax error
- Run-time error
- Logic error

Some Types of Syntax Errors

- Misspellings

```
lstBox.Itms.Add(3)
```

- Omissions

```
lstBox.Items.Add(2 + )
```

- Incorrect punctuation

```
Dim m; n As Integer
```

Displayed as blue underline in VS

Syntax Error

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

$$(2 < n < 5)$$

A Type of Run-time Error

```
Dim numVar As Integer = 1000000
numVar = numVar * numVar
```

What's wrong with the above?

A Logical Error

```
Dim average As Double
Dim m As Double = 5
Dim n As Double = 10
average = m + n / 2
```

What's wrong with the above?

Value of *average* will be 10. Should be 7.5.

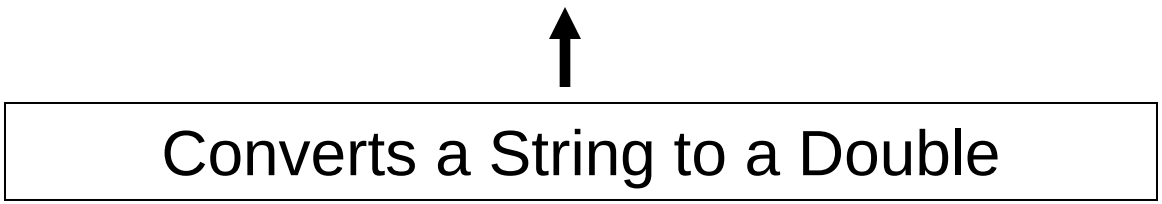
Common Error in Boolean Expressions

- A common error is to replace the condition *Not (2 < 3)* with the condition *(2 > 3)*

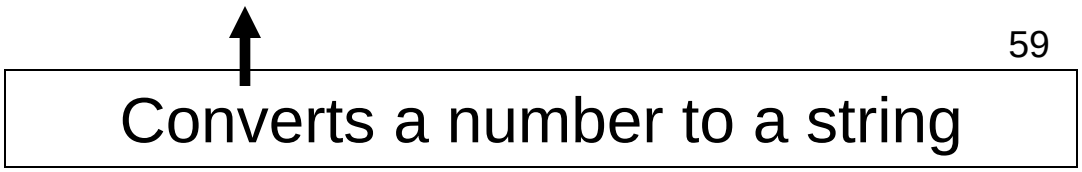
Data Conversion

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

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




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



String Properties and Methods

"Visual".ToUpper is VISUAL.

.ToUpper makes everything upper case.

Varname = “blah”

Varname.ToUpper → “BLAH”

String Properties and Methods

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

.trim removes leading/trailing spaces

Varname = “ blah “

Varname.trim → “blah”

Substring Method

Let *str* be a string.

`str.Substring(`*`m`*`, `*`n`*`)` is the substring of length *n*, beginning at position *m* in *str*.

"Visual Basic".Substring(2, 3) is "sua"

"Visual Basic".Substring(0, 1) is "V"

CHR FUNCTION

For n between 0 and 255,

```
Chr(n)
```

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

EXAMPLES: `Chr(65) is "A"`

`Chr(162) is "¢"`

ASC FUNCTION

For a string *str*,

```
Asc(str)
```

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

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

`Asc("¢25") is 162`

Scope

- The **scope** of a variable is the portion of the program that can refer to it

- Variables declared inside an event procedure are said to have **local scope** and are only available in the event procedure in which they are declared

Scope

- Variables declared outside an event procedure are said to have **class-level scope** and are available to every event procedure

- Usually declared after

  ```
  Public Class formName
  ```

  (Declarations section of Code Editor.)

Review

Chapter 4

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"

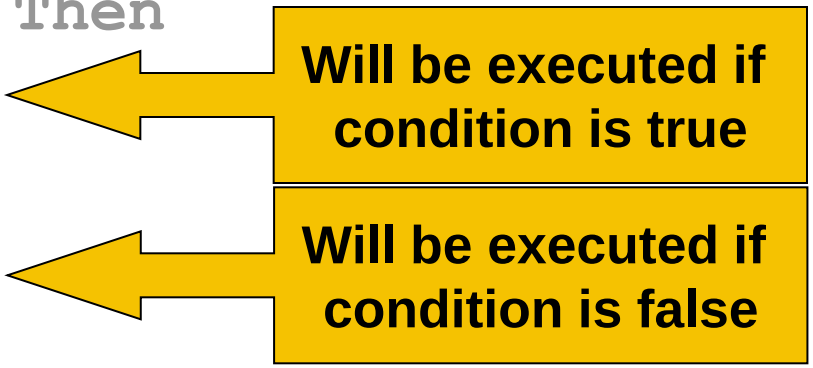
Are the following expressions true or false?

1) Not (n < 6)
2) (answ = "Y") Or (answ = "y")
3) (answ = "Y") And (answ = "y")
4) Not(answ = "y")

If Block

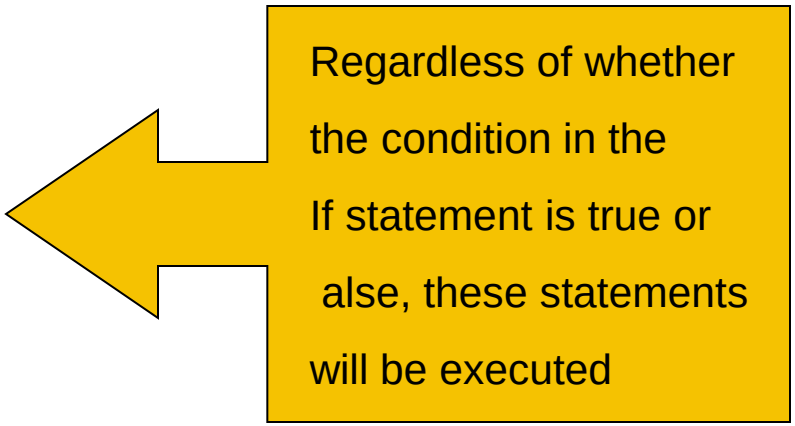
The program will take a course of action based on whether a condition is true.

```
If condition Then
   action1
Else
   action2
End If
```



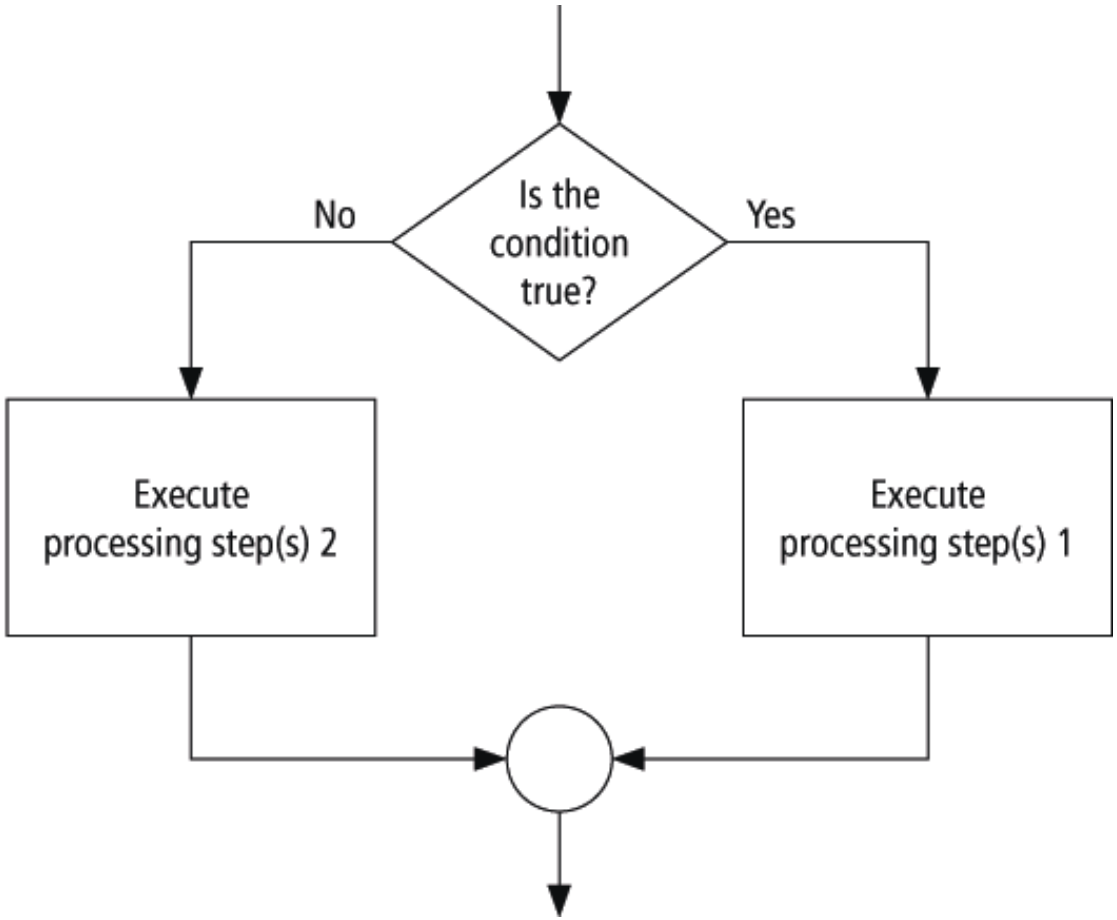
Another example If block

```
If condition Then
   action1
End If
Statement2
Statement3
```



Pseudocode and Flowchart

```
If condition is true Then
  Processing step(s) 1
Else
  Processing step(s) 2
End If
```



ElseIf clause

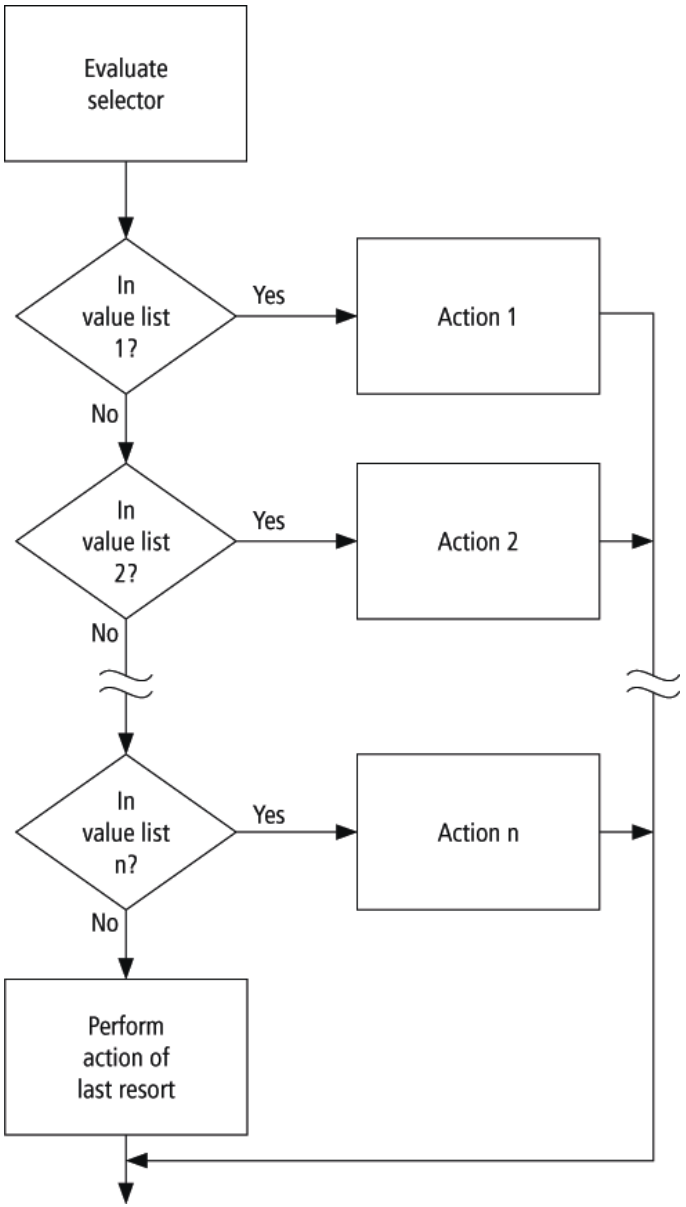
```
If condition1 Then
   action1
ElseIf condition2 Then
   action2
ElseIf condition3 Then
   action3
Else
   action4
End If
```

Select Case Syntax

The general form of the Select Case block is

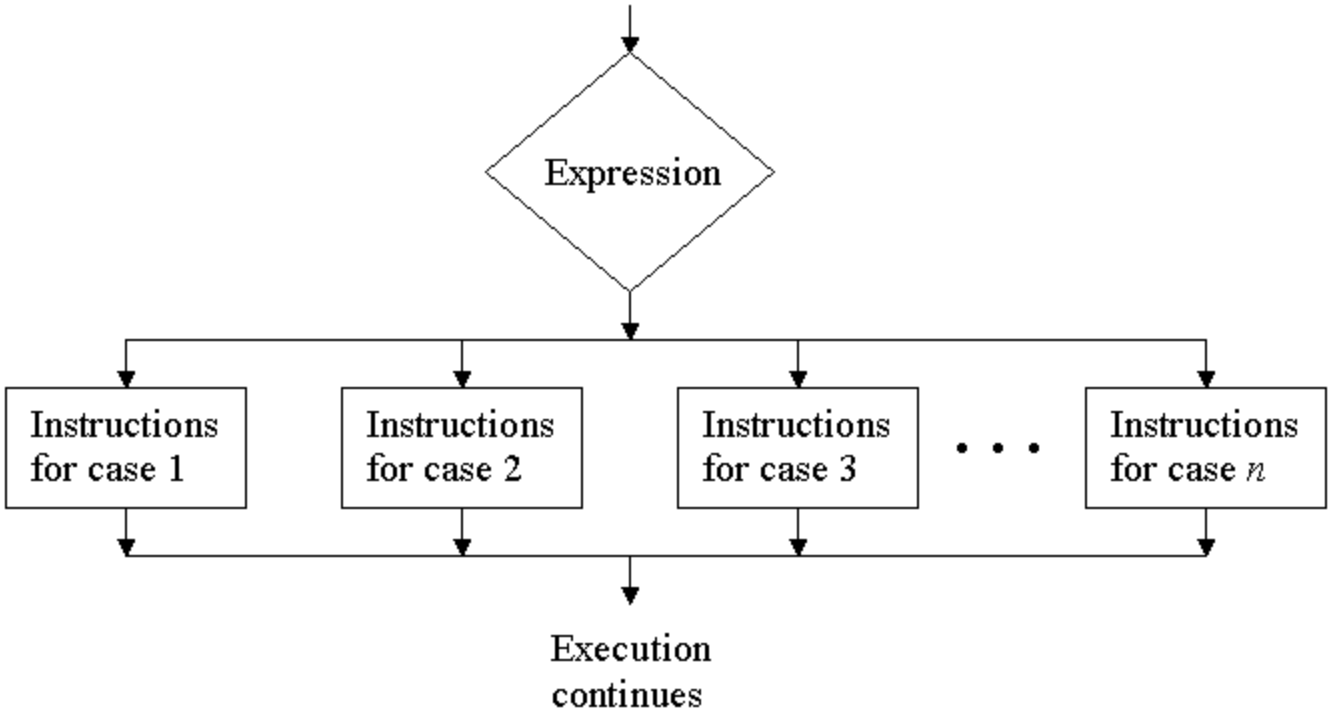
```
Select Case selector
   Case valueList1
      action1
   Case valueList2
      action2
   Case Else
      action of last resort
End Select
```

Flowchart for Select Case



Flowchart for Select Case

- (not in book, but equivalent)



Review

Chapter 5

Devices for Modularity

- Visual Basic has two devices for breaking problems into smaller pieces:
  - Sub procedures
  - Function procedures

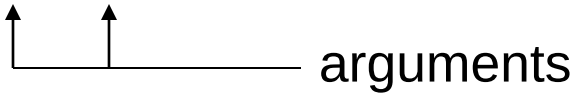
Sub Procedures

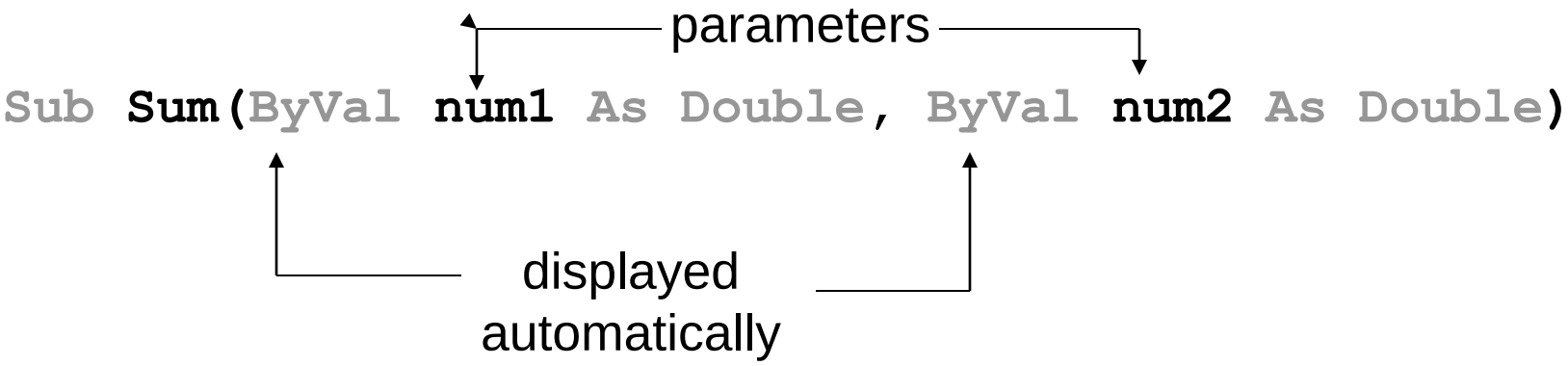
- Perform one or more related tasks
- General syntax

```
Sub ProcedureName()
  statements
End Sub
```

Arguments and Parameters

```
Sum(2, 3)
```





Function Procedures

- Function procedures (aka user-defined functions) always return one value

- Syntax:

```
Function FunctionName(ByVal var1 As Type1, _
                      ByVal var2 As Type2, _
                        …) As dataType
   statement(s)
   Return expression
End Function
```

Functions vs. Procedures

- Both can perform similar tasks
- Both can call other subs and functions
- Use a function when you want to return one and only one value

EXAMPLE: NUM

```
Public Sub btnOne_Click (...) Handles _
                                    btnOne.Click
  Dim num As Double = 4
  Triple(num)
  txtBox.Text = CStr(num)
End Sub

Sub Triple(ByVal num As Double)
  num = 3 * num
End Sub
```

Output: **4**

EXAMPLE

```
Public Sub btnOne_Click (...) Handles _
                                    btnOne.Click
  Dim num As Double = 4
  Triple(num)
  txtBox.Text = CStr(num)
End Sub

Sub Triple(ByRef num As Double)
  num = 3 * num
End Sub
```

Output: 12

Review

Chapter 6

Do Loop Syntax

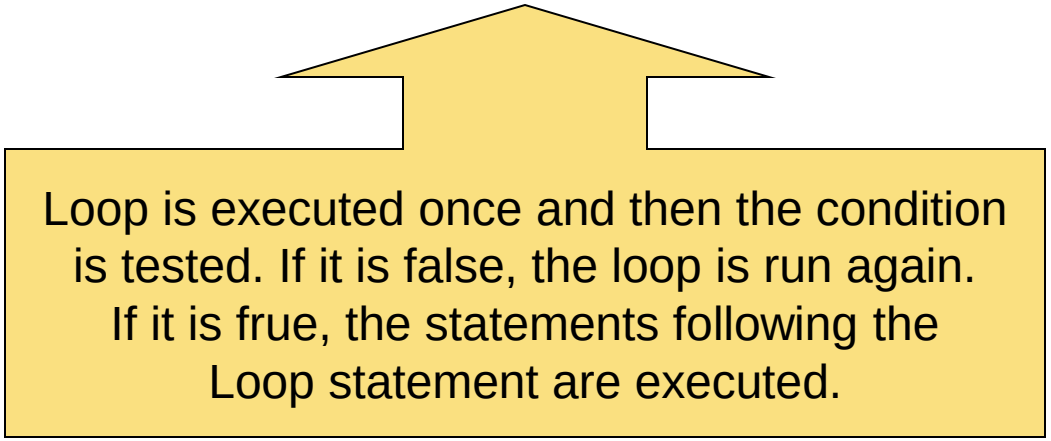
```
Do While condition
   statement(s)
Loop
```

Condition is tested,
If it is true,
the loop is run.
If it is false,
the statements
following the
Loop statement
are executed.

These statements are inside
the body of the loop and
are run if the condition
above is true.

Post Test Loop

```
Do
   statement(s)
Loop Until condition
```



What's the diff?

```
Do
   statement(s)
Loop Until condition
```

```
Do While condition
   statement(s)
Loop
```

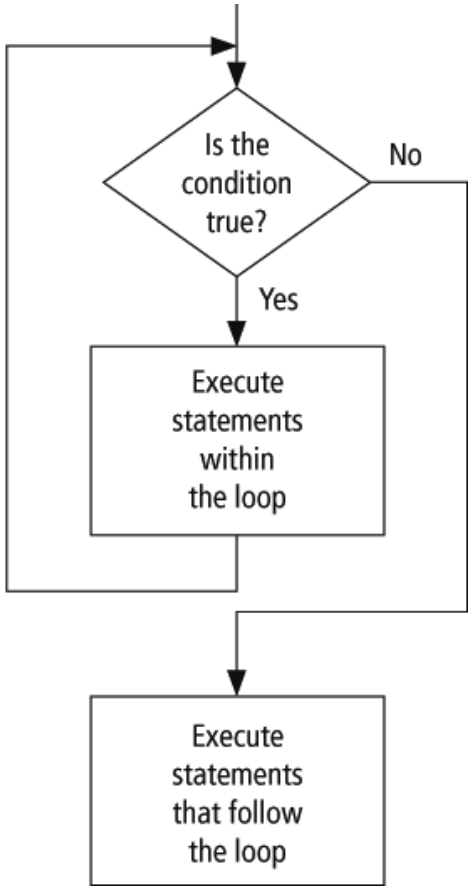
What's the difference between a

Do Until

and

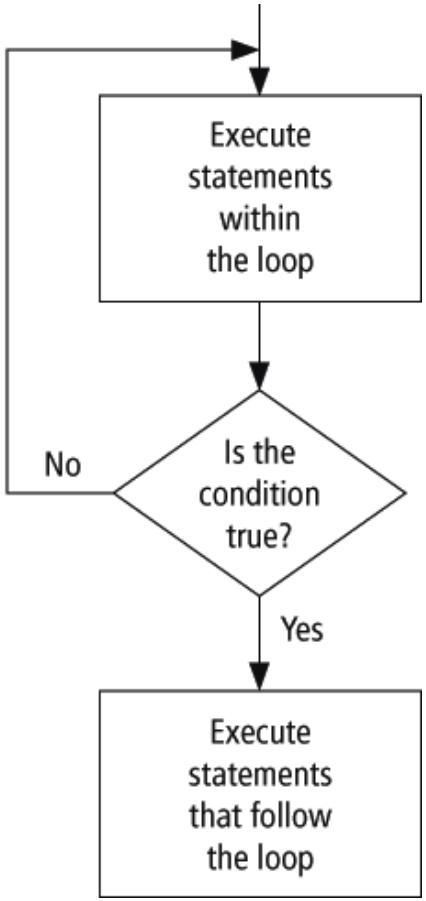
Do While?

Pseudocode and Flowchart




```
Do While condition is true
   Processing step(s)
Loop
```

```
Do
   statement(s)
Loop Until condition is true
```

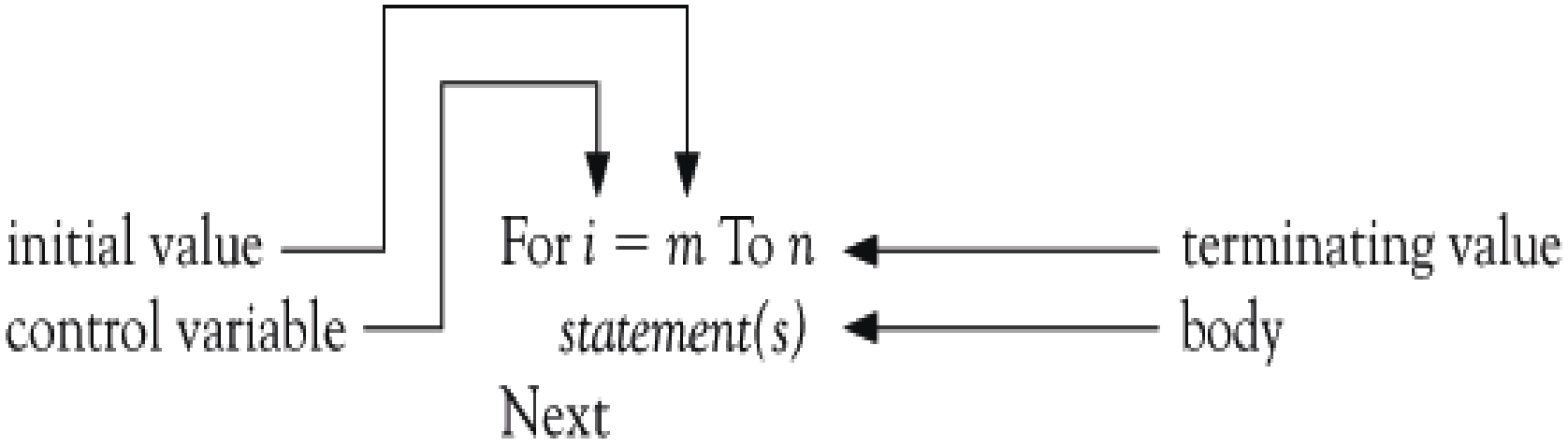


Example 1: Display the Total Contents of a File

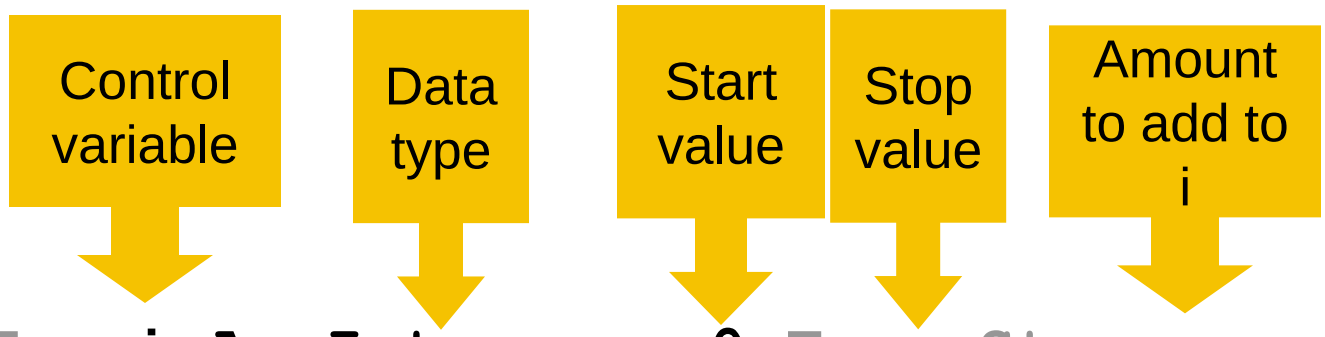
```
Dim sr As IO.StreamReader = _

  IO.File.OpenText("PHONE.TXT")
lstNumbers.Items.Clear()
Do While sr.Peek <> -1
   name = sr.ReadLine
   phoneNum = sr.ReadLine
   lstNumbers.Items.Add(name & " " _
                        & phoneNum)
Loop
sr.Close()
```

For...Next Loop Syntax



EXAMPLE 2




```
For i As Integer = 0 To n Step s
   lstValues.Items.Add(i)
Next
```

Comments

- The value of the control variable should not be altered within the body of the loop (For ... Next).
- To skip an iteration in a For .. Next loop:
  Continue For
- To skip an iteration in a Do .. While loop:
  Continue Do

COMMENTS

```
For i As Integer = 1 To 5
    (some statements)
      Continue For
    (some statements)
Next
```

What will happen?

Comments

- To break out of a For .. Next loop:
  Exit For
- To break out of a Do .. While loop:
  Exit Do

Review

Chapter 7

Simple and Array Variables

- A **variable** (or simple variable) is a name to which Visual Basic can assign a single value

- An **array variable** is a *collection* of simple variables of the *same type* to which Visual Basic can efficiently assign a list of values

Initializing Arrays

- Arrays may be initialized when they are created:

```
Dim arrayName() As varType = {value0, _
        value1, value2, ..., valueN}
```

- For Ex:

```
Dim Students() As String = {"Jack",
     "John", "Julie", "Jimmy", "Janet"}
```

Initializing Arrays

- Arrays may be initialized when they are created:

```
Dim arrayName() As varType =
  IO.File.ReadAllLines(filespec)
```

- Opens *filespec*, reads all lines from it, and stores it in *arrayName*
- Each line in *filespec* is stored in one location of *arrayName*

EXAMPLE

```
Dim Grades() As integer = {70, 75, 80, 85, 90}
```

Grades.Average → 80

Grades.Count → 5

Grades.Min → 70

Grades.Max → 90

Grades.Sum → 400

Preserve Keyword

`ReDim` ***`arrayName(m)`*** resets all values to their default. This can be prevented with the keyword **Preserve**.

`ReDim Preserve` ***`arrayName(m)`***

resizes the array and retains as many values as possible.

Set Operations

- Concat
  - Contains elements of array1 and array2
  - Duplication is OK

```
Dim States1() As String = {"A", "B", "C", "D"}
Dim States2() As String = {"E", "F", "G", "H"}

Dim States3() As String = _
                    States1.Concat(States2).ToArray()
```

Set Operations

- Union
  - Contains elements of array1 and array2
  - No Duplication

```
Dim States1() As String = {"A", "B", "C", "D"}
Dim States2() As String = {"E", "F", "G", "H"}

Dim States3() As String = _
                    States1.Union(States2).ToArray()
```

Set Operations

- Intersect
  - Contains elements from array1 and array2 which exist in *both* array1 and array2

```
Dim States1() As String = {"A", "B", "C", "D"}
Dim States2() As String = {"E", "F", "G", "H"}

Dim States3() As String = _
            States1.Intersect(States2).ToArray()
```

Set Operations

- Except
  - Contains elements from array1 which do not exist in array2

```
Dim States1() As String = {"A", "B", "C", "D"}
Dim States2() As String = {"E", "F", "G", "H"}

Dim States3() As String = _
                    States1.Except(States2).ToArray()
```

Structures

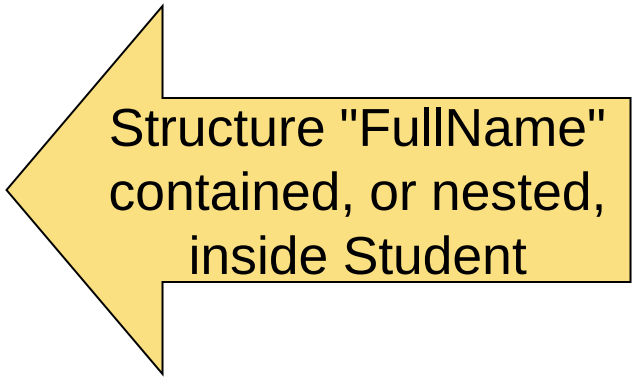
- A way of grouping heterogeneous data together
- Also called a UDT (User Defined Type)
- Sample structure definition:

```
Structure College
  Dim name As String
  Dim state As String
  Dim yearFounded As Integer
End Structure
```

EXAMPLE 4

```
Structure FullName
  Dim firstName As String
  Dim lastName As String
End Structure

Structure Student
  Dim name As FullName
  Dim credits() As Integer
End Structure
```



SORTING

- Sorting is an algorithm for ordering an array.
- We discuss two sorting algorithms:
  - **bubble sort**
  - **Shell sort**
- Both use the swap algorithm:

```
temp = var1
var1 = var2
var2 = temp
```

Bubble Sort Algorithm: *N* Items

1. Compare the first and second items. If they are out of order, swap them.
2. Compare the second and third items. If they are out of order, swap them.
3. Repeat this pattern for all remaining pairs. The final comparison and possible swap are between the next-to-last and last items.
4. The last item will be at its proper place.
5. Do another pass through first $n - 1$ items.
6. Repeat this process with one less item for each pass until a pass uses only the first and second items.

Shell Sort Algorithm

1. Begin with a gap of $g = Int(n/2)$
2. Compare items 0 and g, 1 and $1 + g$, . . ., $n - g$ and n. Swap any pairs that are out of order.
3. Repeat Step 2 until no swaps are made for gap g.
4. Halve the value of g.
5. Repeat Steps 2, 3, and 4 until the value of g is 0.

Efficiency of Bubble and Shell Sorts

Array Elements	Bubble Sort Comparisons	Shell Sort Comparisons
5	10	17
10	45	57
15	105	115
20	190	192
25	300	302
30	435	364
50	1225	926
100	4950	2638
500	124,750	22,517
1000	499,500	58,460

Review

- Efficiency of Sequential and Binary Search

Array Elements	Sequential Search Comparisons	Binary Search Comparisons
2000	1000 (Average)	11 (At most)

Review

Chapter 8

Review

name = input.Substring(0, (input.IndexOf(",")))

Input = “James,88”

0	1	2	3	4	5	6	7
J	a	m	e	s	,	8	8

WriteAllLines

```
IO.File.WriteAllLines _
        ("fileName.txt", States)
```

- Creates a new text file
- Copies the contents of a string array
- Places one element on each line
- Close the file

RealAllLines

- Read *all* the lines of a text-file into an array
  - Method opens a file
  - Reads each line of the file
  - Adds each line as an element of a string array
  - Closes the file
- A line is defined as a sequence of characters followed
  - carriage return
  - a line feed
  - a carriage return followed by a line feed

Deleting Information from a Sequential File

- An individual item of a file cannot be changed or deleted directly.
- A new file must be created by reading each item from the original file and recording it, with the single item changed or deleted, into the new file.
- The old file is then erased, and the new file renamed with the name of the original file.

Delete and Move Methods

- Delete method:

```
IO.File.Delete(filespec)
```

- Move method (to change the filespec of a file):

```
IO.File.Move(oldfilespec,
             newfilespec)
```

- ***Note:*** The IO.File.Delete and IO.File.Move methods cannot be used with open files.

Structured Exception Handling

- Two types of problems in code:
  - *Bugs* (logic error) – something wrong with the code the programmer has written
  - *Exceptions* – errors beyond the control of the programmer

- Programmer can use the debugger to find bugs; but must anticipate exceptions in order to be able to keep the program from terminating abruptly.

Try Catch Block Syntax

```
Try
   normal code
Catch exc1 As FirstException
   exception-handling code for FirstException
Catch exc2 As SecondException
   exception-handling code for SecondException
.
.
Catch
   exception-handling code for any remaining
   exceptions
Finally
   clean-up code
End Try
```

Catch Blocks

- Visual Basic allows Try-Catch-Finally blocks to have one or more specialized Catch clauses that only trap a specific type of exception

- The general form of a specialized Catch clause is

```
Catch exp As ExceptionName
```

- where the variable **exp** will be assigned the name of the exception. The code in this block will be executed only when the specified exception occurs.

Working with HashTable

- Arrays can store only one data type
- collections can hold any objects
- Accessing the element is very simple and very fast
- Removing the element in Collection is very simple

Working with HashTable

- Array:
  - MyArray(100) returns element 100
- What if I want element “George”?
- Hashtable
  - MyHash.Item(“George”)

Working with HashTable

- **Adding an element to the HashTable**

- {hash table object}.Add(Key as Object, value as Object)

- Ex: MyHash.Add("George", 45)

Working with HashTable

- **Accessing an element**

  {hash table object}.Item({key})

  Ex: MyArray.Item("George")

Working with HashTable

- **Searching for an element**

- {hash table object}.Contains({key})

  Ex: MyArray.Contains("George")

Review

Chapter 9

List Box Events

Three main types of events with list boxes:

1. ***Click*** – the user clicks on an item in the list box

2. ***SelectedIndexChanged*** - the user clicks on an item or uses the arrow keys to select it

3. ***DoubleClick*** - the user double-clicks on an item

All three events are triggered when the user double-clicks on an item.

Using an Array to Fill a List Box

The statement

```
lstBox.DataSource = arrayName
```

fills the list box with the elements of the array.

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 a control to a group box, create the group box, then drag the control you want to attach into the group box.

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 is *True* when the check box is checked and *False* when it is not
- *CheckedChanged* event is triggered when the user clicks on the check box

The Radio Button Control

- Consists of a small circle with a caption (that is set by the Text property)
- Normally several radio buttons are attached to a group box
- Gives the user a single choice from several options
- Clicking on one radio button removes the selection from another

The Timer Control

- Invisible during run time
- Triggers an event after a specified period of time
- The *Interval* property specifies the time period – measured in milliseconds
- To begin timing, set the *Enabled* property to *True*
- To stop timing, set the *Enabled* property to *False*
- The event triggered each time *Timer1.Interval* elapses is called *Timer1.Tick*

Scroll Bar Properties

- The main properties of a scroll bar control are
  - Minimum
  - Maximum
  - Value
  - SmallChange,
  - LargeChange

- *hsbBar.Value*, a number between *hsbBar.Minimum* and *hsbBar.Maximum*, gives the location of the scroll box,

The Clipboard Object

- Used to copy information from one place to another
- Maintained by Windows, so it can even be used with programs outside Visual Basic
- A portion of memory that has no properties or events

The Random Class

- A random number generator declared with the statement:

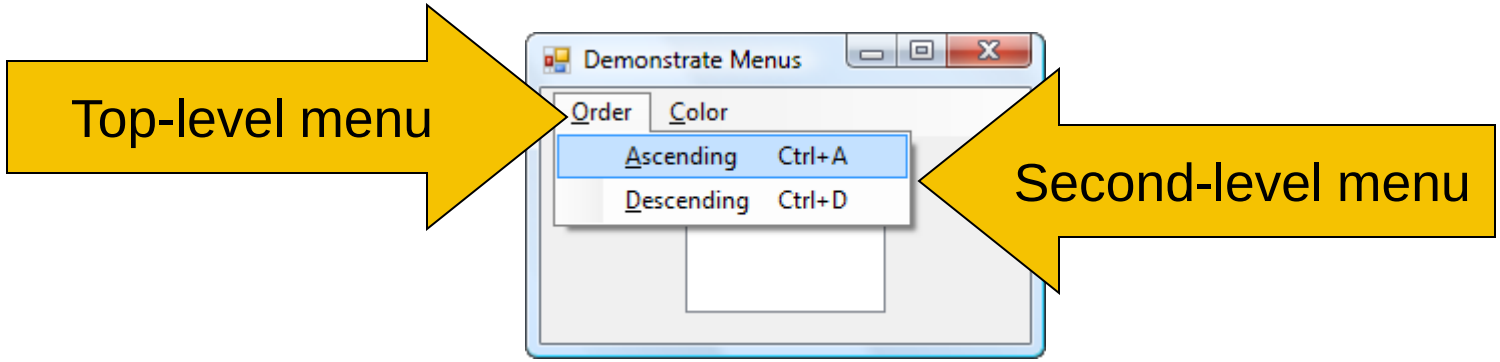
```
Dim randomNum As New Random()
```

- If *m* and *n* are whole numbers and *m* < *n* then the following generates a whole number between *m* and *n (including m, but excluding n)*

```
randomNum.Next(m, n)
```

The MenuStrip Control

Used to create menus like the following:



Menu Events

- Each menu item responds to the Click event

- Click event is triggered by
  - the mouse
  - Alt + access key
  - Shortcut key

Multiple Forms

- Visual Basic programs can contain more than one form

- To add the new form, select *Add Windows Form* from the *Project* menu, to invoke the *Add New Items* dialog box.

Variables and Multiple Forms

- Variables declared in the *Declarations* section of a form with *Public*, instead of *Dim*, will be available to all forms in the program

- When a *Public* variable is used in another form, it is referred to by an expression such as

```
secondForm.variableName
```

Review

Chapter 10

What is a database?

- A *database* (*DB*) is a very large, integrated, permanent collection of data

- Models real-world
  - Entities (e.g., students, courses)
  - Relationships (e.g., Madonna is taking CMPT354)

- Example databases:
  - Customer Transactions
  - Human Genome
  - Online Bookstore
  - . . .

Database Terminology

- A **table** is a rectangular array of data
- Each column of the table, called a **field**, contains the same type of information
- Each row, called a **record**, contains all the information about one entry in the database

Connecting with a DataTable

```
Dim dt As New DataTable()
Dim connStr As String = _
        "Provider=Microsoft.Jet.OLEDB.4.0;" & _
        "Data Source=MEGACITIES.MDB"
Dim sqlStr As String = "SELECT * FROM Cities"
Dim dataAdapter As New _
         OleDb.OleDbDataAdapter(sqlStr, connStr)
dataAdapter.Fill(dt)
dataAdapter.Dispose()
```


(Boilerplate to be inserted into every program in chapter.)

Primary Keys

- A **primary** key is used to uniquely identify each record
- Databases of student enrollments in a college usually use a field of Social Security numbers as the primary key
- Why wouldn't names be a good choice as a primary key?

Two or More Tables

- When a database contains two or more tables, the tables are usually related
- For instance, the two tables *Cities* and *Countries* are related by their country field
- Notice that every entry in *Cities.country* appears uniquely in *Countries.country* and *Countries.country* is a primary key
- We say that *Cities.country* is a **foreign key** of *Countries.country*

- **Structured Query Language** developed for use with relational databases
- Very powerful language
- Allows for the request of specified information from a database
- Allows displaying of information from database in a specific format

Virtual Tables

- SQL statements create a new "virtual" table from existing tables

```
SELECT city, pop2015 FROM Cities
  WHERE pop2015>=20
```

Results in "virtual" table

city	pop2015
Bombay	22.6
Delhi	20.9
Mexico City	20.6
Sao Paulo	20.0
Tokyo	36.2

Review

Chapter 11

What is Object Oriented Programming?

An object is like a black box.

The internal details are hidden.

- Identifying *objects* and assigning *responsibilities* to these objects.
- Objects communicate to other objects by sending *messages*.
- Messages are received by the *methods* of an object

OOP Analogy

Classes → **noun** A word used to denote or name a person, place, thing, quality, or act.

Methods → **verb** That part of speech that expresses existence, action, or occurrence.

Properties → **adjective** Any of a class of words used to modify a noun or other substantive by limiting, qualifying, or specifying.

Get and Set

Property block

```
Private m_name As String

Public Property Name() As String
  Get
    Return m_name
  End Get
  Set(ByVal value As String)
    m_name = value
  End Set
End Property
```

Student Class: WriteOnly Property Blocks

```
Public WriteOnly Property Midterm() As Double
  Set(ByVal value As String)
    m_midterm = value
  End Set
End Property

Public WriteOnly Property Final() As Double
  Set(ByVal value As String)
    m_final = value
  End Set
End Property
```

Student Class: Method

```
Function CalcSemGrade() As String
  Dim grade As Double
  grade = (m_midterm + m_final) / 2
  grade = Math.Round(grade)
  Select Case grade
    Case Is >= 90
      Return "A"
    Case Is >= 80
      Return "B"
:
End Function
```

Steps Used to Create a Class

1. Identify a *thing* in your program that is to become an object

2. Determine the properties and methods that you would like the object to have. (As a rule of thumb, properties should access data, and methods should perform operations.)

3. A class will serve as a template for the object. The code for the class is placed in a class block of the form

```
Class ClassName
   statements
End Class
```

Steps continued

4. For each of the properties in Step 2, declare a private member variable with a statement of the form

```
Private m_variableName As DataType
```

5. For each of the member variables in Step 4, create a Property block with Get and/or Set procedures to retrieve and assign values of the variable.

6. For each method in Step 2, create a Sub procedure or Function procedure to carry out the task.

Events

- User-defined events can be created for classes.
- The statement for triggering an event is located in the class block
- The event is dealt with in the form's code.

User Defined Event

- Suppose that the event is named UserDefinedEvent and has the parameters *par1*, *par2*, and so on.

- In the class block, place the following statement in the Declarations section

```
Public Event UserDefinedEvent(ByVal par1 As _
      DataType1, ByVal par2 As DataType2, ...)
```

- The next statement should be placed at the locations in the class block code at which the event should be triggered

```
RaiseEvent UserDefinedEvent(arg1, arg2, ...)
```

Responding to Events

- When declaring an object variable, the keyword WithEvents must be added so that the object will respond to events:

```
Dim WithEvents object1 As ClassName
```

- The declaration line of an event procedure would be

```
Private Sub object1_UserDefinedEvent(ByVal par1 As _
        DataType1, ...) Handles object1.UserDefinedEvent
```

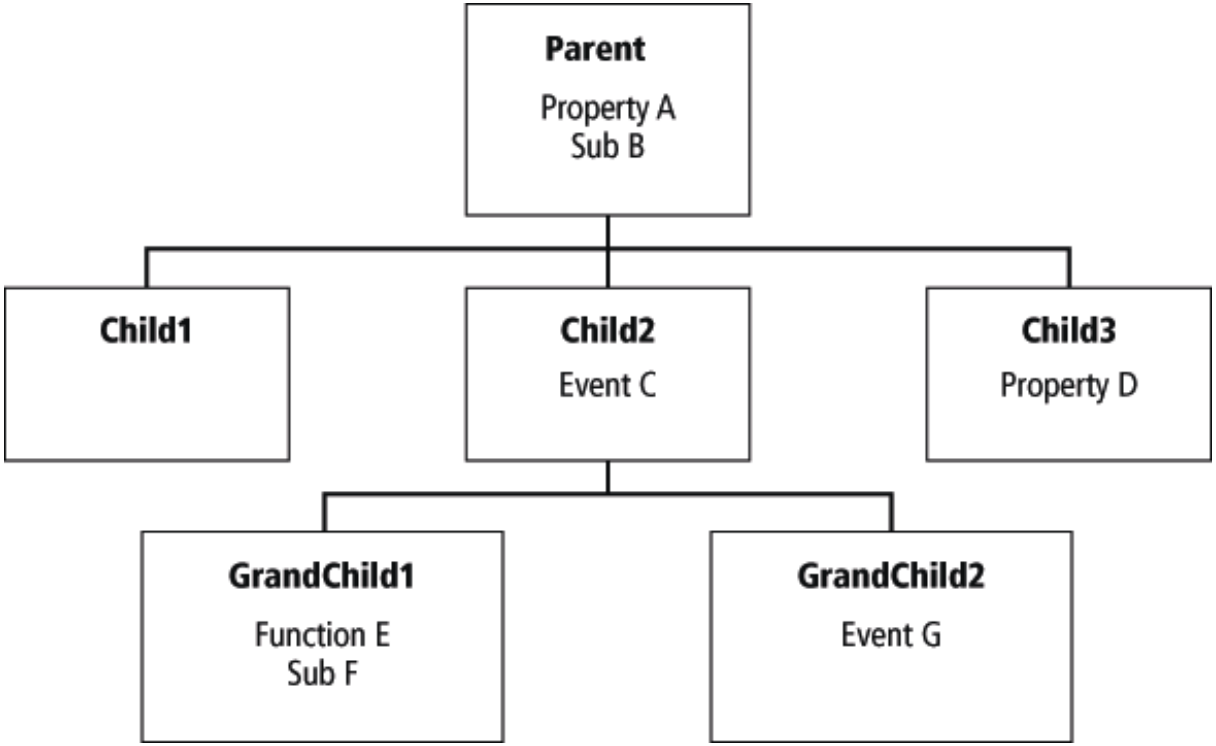
CONTAINMENT

- Class A **contains** class B when a member variable of class A is an object of type class B.

```
Class DeckOfCards
  Private m_deck(51) As Card
  'Class DeckOfCards contains class
Card
```

Inheritance Hierarchy

- *GrandChild1* has access to *Property A*, *Sub B*, and *Event C* from its parent and adds *Function E* and *Sub F*



OVERRIDABLE

- The keyword **`Overridable`** is used to designate the parent's methods that are overridden, and the keyword **`Overrides`** is used to designate the child's methods that are doing the overriding

- There are situations where a child class's needs to access the parent class's implementation of a method that the child is overriding. Visual Basic provides the keyword **`MyBase`** to support this functionality

MustOverride

- Sometimes you want to insist that each child of a class have a certain property or method that it must implement for its own use

- Such a property or method is said to be **abstract** and is declared with the keyword **MustOverride**

- Good luck!