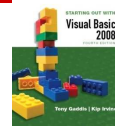


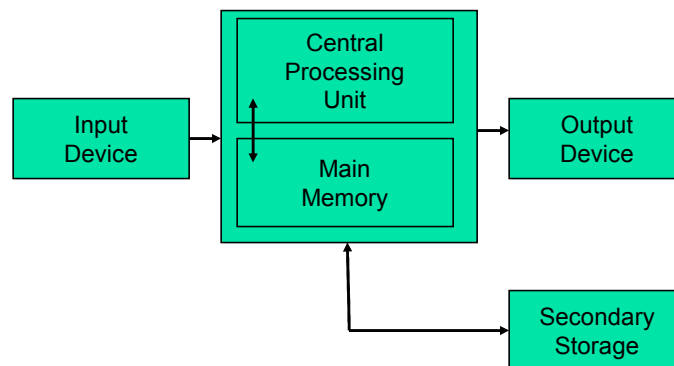
Computer Hardware

- Refers to the physical components
- Not one device but a system of many devices
- Major types of components include:
 - Central Processing Unit
 - Main memory
 - Secondary storage devices
 - Input devices
 - Output devices

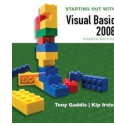
Slide 1- 3



Organization of a Computer System



Slide 1- 4



The CPU

- Fetches instructions from main memory
- Carries out the operations commanded by the instructions
- Each instruction produces some outcome
- A *program* is an entire sequence of instructions
- Instructions are stored as *binary numbers*
- *Binary number* - a sequence of 1's and 0's

Slide 1- 5



Main Memory

- Commonly known as random access memory, or just RAM
- Holds instructions and data needed for programs that are currently running
- RAM is usually a *volatile* type of memory
 - Contents of RAM are lost when power is turned off
- Can visualize memory as a long row of locations each with a numeric address

Slide 1- 6

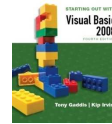
First-Generation and Second-Generation (Low-Level) Languages



- Low-level languages
 - First-generation and second-generation languages
 - Machine-dependent languages
 - The underlying representation the machine actually understands
- First-generation languages
 - Also referred to as machine languages
 - Consist of a sequence of instructions represented as binary numbers
 - E.g.: Code to ADD might be 1001 . To add 1+0 and then 1+1 our program might look like this:
 - 1001 0001 0000
 - 1001 0001 0001

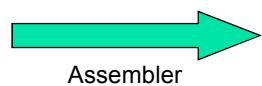
Slide 1- 7

First-Generation and Second-Generation (Low-Level) Languages

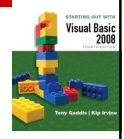


- Second-generation languages
 - Also referred to as assembly languages
 - Abbreviated words are used to indicate operations
 - Allow the use of decimal numbers and labels to indicate the location of the data
- Assemblers
 - Programs that translate assembly language programs into machine language programs
 - Our add program now looks like:

| | | |
|-----------|---|------|
| ■ ADD 1,0 | → | 1001 |
| ■ ADD 1,1 | → | 0001 |
| | | 0000 |
| | | 1001 |
| | | 0001 |
| | | 0001 |

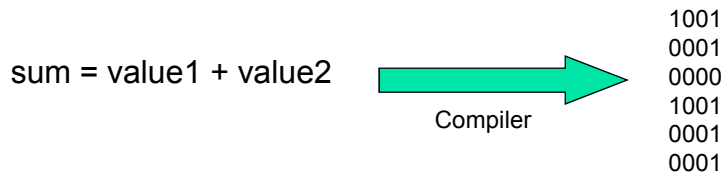


Slide 1- 8

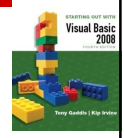


Third-Generation and Fourth-Generation (High-Level) Languages

- High-level languages
 - Third-generation and fourth-generation languages
 - Programs can be translated to run on a variety of computer types
- Third-generation languages
 - Procedure-oriented languages
 - Object-oriented languages
- Our Add program might now look like:

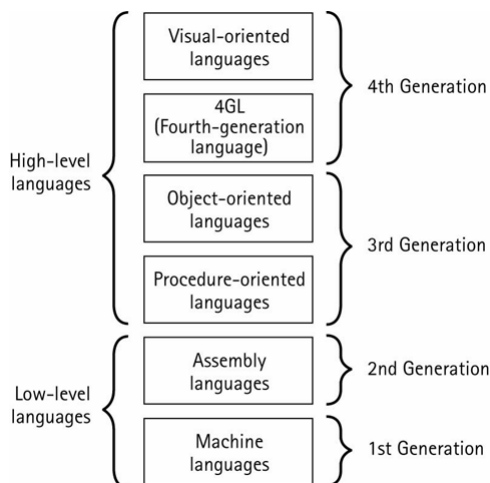


Slide 1- 9



Third-Generation and Fourth-Generation (High-Level) Languages (Continued)

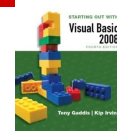
The Evolution of Programming Languages



Slide 1- 10

Third-Generation and Fourth-Generation (High-Level) Languages

- Procedure-oriented languages
 - Programmers concentrate on the procedures used in the program
 - Procedure: a logically consistent set of instructions which is used to produce one specific result
- Object-oriented languages
 - Items are represented using self-contained objects
 - Often used for graphical windows environments, ability to re-use code efficiently

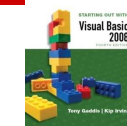


Slide 1- 11

Example of an Object

- This is a Visual Basic *GUI* object called a form
- Contains data and actions
- Data, such as Hourly Pay Rate, is a text *property* that determines the appearance of form objects
- Actions, such as Calculate Gross Pay, is a *method* that determines how the form reacts
- A form is an object that contains other objects such as buttons, text boxes, and labels

| | |
|---|----------------------|
| Number of Hours Worked | <input type="text"/> |
| Hourly Pay Rate | <input type="text"/> |
| Gross Pay Earned | \$ 0.00 |
| <input type="button" value="Calculate Gross Pay"/> <input type="button" value="Close"/> | |



Slide 1- 12

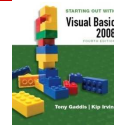
Example of an Object



- Form elements are objects called *controls*
- This form has:
 - Two *TextBox* controls
 - Four *Label* controls
 - Two *Button* controls
- The value displayed by a control is held in the *text property* of the control
- Left button text property is *Calculate Gross Pay*
- Buttons have methods attached to *click events*

Slide 1- 13

Third-Generation and Fourth-Generation (High-Level) Languages



- Graphical user interface (GUI)
 - Provides a graphical way for the user to interact with the program
 - Uses events
- Event
 - A specific procedure that is connected to an object
- Visual languages
 - Permit the programmer to manipulate graphical objects directly, with the language providing the necessary code
 - Permit users to access and format information without the need for writing any procedural code

Slide 1- 14

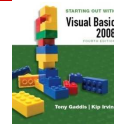
The Visual Basic .NET Platform



- Visual Basic .NET is in a sense one step removed from a typical high-level language
- VB.NET runs using a “Virtual Machine” or “Common Language Runtime”
 - The physical computer simulates a virtual computer that runs your program
- What is .NET?
 - Microsoft’s vision of the future of applications in the Internet age
 - Increased robustness over classic Windows apps
 - New programming platform
 - Built for the web
 - .NET is a platform that runs on the operating system

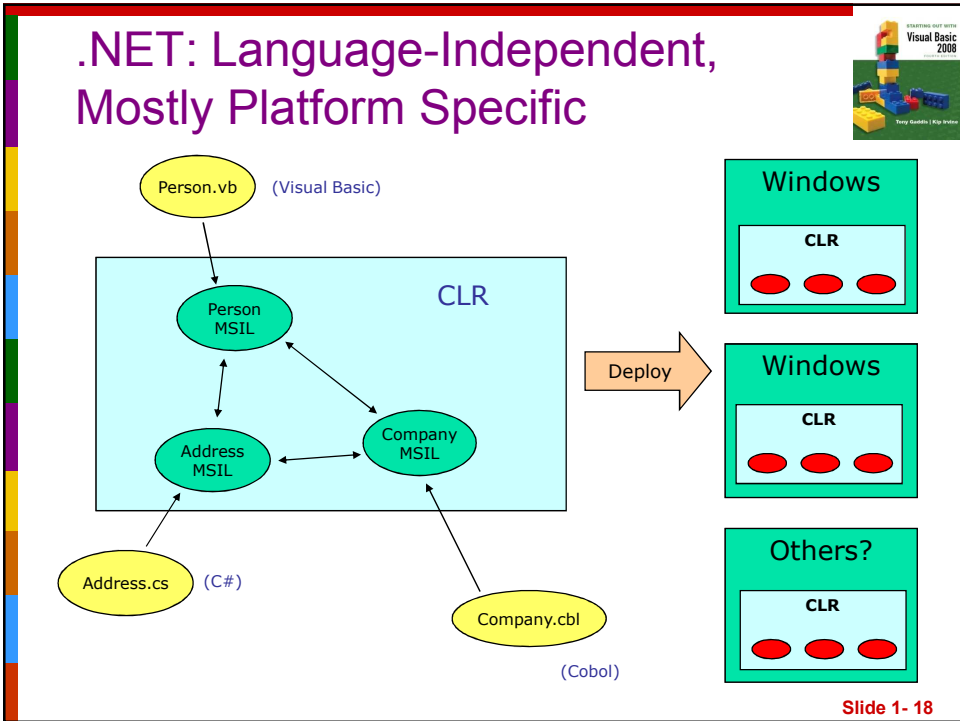
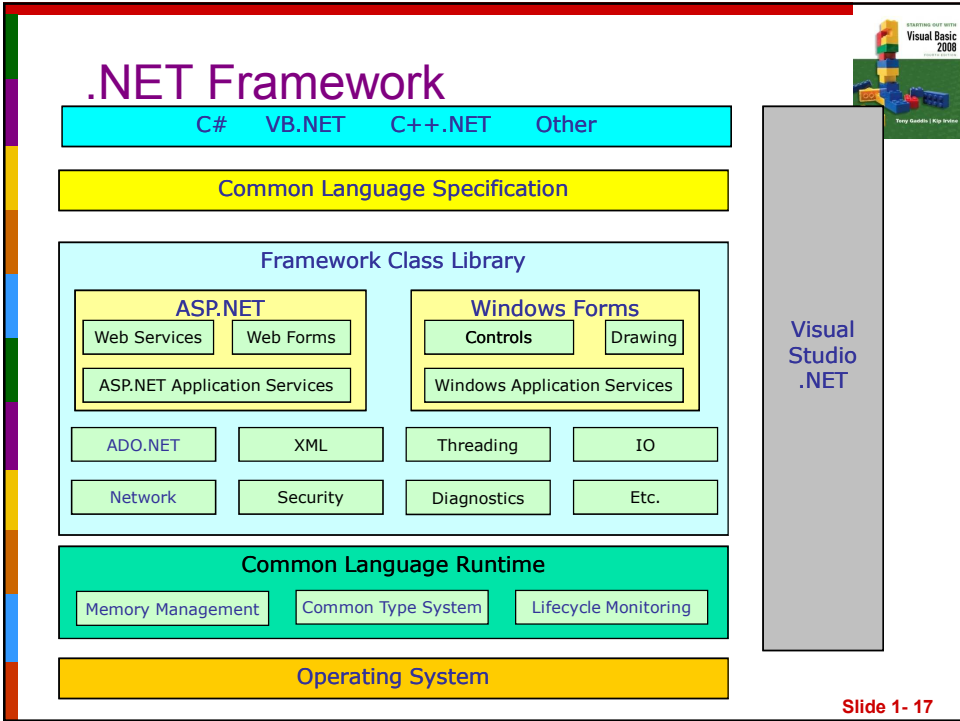
Slide 1- 15

.NET



- .NET is actually a program that sits on top on the Operating System
- Provides language interoperability across platforms
- Strong emphasis on Web connectivity
- Platform/language independent

Slide 1- 16

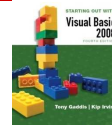




Programming

- Computers can only follow instructions
- In VB.NET our instructions must sometimes be very detailed and sometimes can be more general
- A *computer program* is a set of instructions on how to solve a problem or perform a task
- Example:
 - In order for a computer to compute someone's gross pay, we must tell it to perform the steps on the following slide

Slide 1- 19



Computing Gross Pay

- Display message: "How many hours did you work?"
- Allow user to enter number of hours worked
- Store the number the user enters in memory
- Display message: "How much are you paid per hour?"
- Allow the user to enter an hourly pay rate
- Store the number the user enters in memory
- Multiply hours worked by pay rate and store the result in memory
- Display a message with the result of the previous step

This well-defined, ordered set of steps for solving a problem is called an *algorithm*

Slide 1- 20



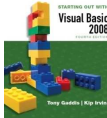
STARTING OUT WITH
**Visual Basic
2008**
FOURTH EDITION
Tony Gaddis | Kip Irvine

1.3 More About Controls and Programming

As a Visual Basic Programmer, You Must Design and Create the Two Major Components of an Application: the GUI Elements (Forms and Other Controls) and the Programming Statements That Respond to And/or Perform Actions (Event Procedures)



Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley



STARTING OUT WITH
**Visual Basic
2008**
FOURTH EDITION
Tony Gaddis | Kip Irvine

Visual Basic Controls

- As a Windows user you're already familiar with many Visual Basic controls:
 - Label - displays text the user cannot change
 - TextBox - allows the user to enter text
 - Button – performs an action when clicked
 - RadioButton - A round button that is selected or deselected with a mouse click
 - CheckBox – A box that is checked or unchecked with a mouse click
 - Form - A window that contains these controls
- Tutorial 1-3 demonstrates these controls

Slide 1- 22

VB.NET Controls

- Invoking VB.NET
- Text Box
- Button
- Label
- Radio Button
- Checkbox Button
- PictureBox
- Help
- Fonts / Auto Hide

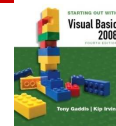
**Follow along and
explore these controls
on your computer!**



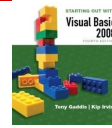
Slide 1- 23

Name Property

- All controls have properties
- Each property has a value (or values)
- Not all properties deal with appearance
- The name property establishes a means for the program to refer to that control
- Controls are assigned relatively meaningless names when created
- Programmers usually change these names to something more meaningful



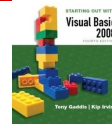
Slide 1- 24



Naming Conventions

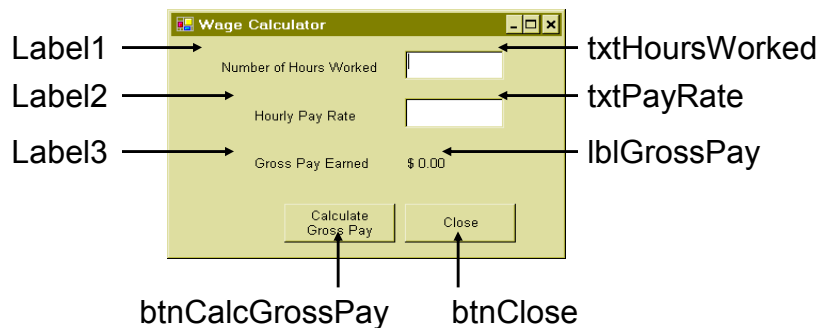
- Control names must start with a letter
- Remaining characters may be letters, digits, or underscore
- 1st 3 lowercase letters indicate the type of control
 - txt... for Text Boxes
 - lbl... for Labels
 - btn... for Buttons
- After that, capitalize the first letter of each word
- txtHoursWorked is clearer than txthoursworked

Slide 1- 25



Examples of Names

- The label controls use the default names (Label1, etc.)
- Text boxes, buttons, and the Gross Pay label play an active role in the program and have been changed



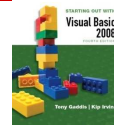
Slide 1- 26



Language Elements

- **Keywords:** Words with special meaning to Visual Basic (e.g., `Private`, `Sub`)
- **Programmer-defined-names:** Names created by the programmer (e.g., `sngGrossPay`, `btnClose`)
- **Operators:** Special symbols to perform common operations (e.g., `+`, `-`, `*`, and `/`)
- **Remarks:** Comments inserted by the programmer – these are ignored when the program runs (e.g., any text preceded by a single quote)

Slide 1- 27



Language Elements: Syntax

- **Syntax** defines the correct use of key words, operators, & programmer-defined names
- Similar to the syntax (rules) of English that defines correct use of nouns, verbs, etc.
- A program that violates the rules of syntax will not run until corrected

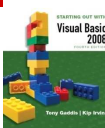
Slide 1- 28



1.4 The Programming Process



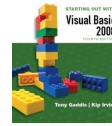
Copyright © 2007 Pearson Education, Inc. Publishing as Pearson Addison-Wesley



Step 1 of Developing an Application

- Clearly define what the program is to do
- For example, the *Wage Calculator* program:
 - Purpose: To calculate the user's gross pay
 - Input: Number of hours worked, hourly pay rate
 - Process: Multiply number of hours worked by hourly pay rate (result is the user's gross pay)
 - Output: Display a message indicating the user's gross pay

Slide 1- 30



Step 2 of Developing an Application

- Visualize the application running on the computer and design its user interface

Number of Hours Worked

Hourly Pay Rate

Gross Pay Earned: \$0.00

Calculate Gross Pay Close

Slide 1- 31

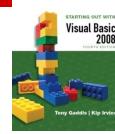


Step 3 of Developing an Application

- Make a list of the controls needed

| <u>Type</u> | <u>Name</u> | <u>Description</u> |
|-------------|-----------------|--|
| TextBox | txtHoursWorked | Allows the user to enter the number of hours worked. |
| TextBox | txtPayRate | Allows the user to enter the hourly pay rate |
| Label | lblGrossPay | Displays the gross pay, after the btnCalcGrossPay button has been clicked |
| Button | btnCalcGrossPay | When clicked, multiplies the number of hours worked by the hourly pay rate |
| Button | btnClose | When clicked, terminates the application |
| Label | (default) | Description for Number of Hours Worked TextBox |
| Label | (default) | Description for Hourly Pay Rate TextBox |
| Label | (default) | Description for Gross Pay Earned Label |
| Form | (default) | A form to hold these controls |

Slide 1- 32



Step 4 of Developing an Application

- Define values for each control's relevant properties:

| <u>Control Type</u> | <u>Control Name</u> | <u>Text</u> |
|---------------------|---------------------|--------------------------|
| Form | (Default) | "Wage Calculator" |
| Label | (Default) | "Number of Hours Worked" |
| Label | (Default) | "Hourly Pay Rate" |
| Label | (Default) | "Gross Pay Earned" |
| Label | lblGrossPay | "\$0.00" |
| TextBox | txtHoursWorked | "" |
| TextBox | txtPayRate | "" |
| Button | btnCalcGrossPay | "Calculate Gross Pay" |
| Button | btnClose | "Close" |

Slide 1- 33

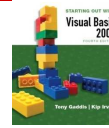


Step 5 of Developing an Application

- List the methods needed for each control:

| <u>Method</u> | <u>Description</u> |
|-----------------------|---|
| btnCalcGrossPay_Click | Multiplies hours worked by hourly pay rate These values are entered into the txtHoursWorked and txtPayRate TextBoxes Result is stored in lblGrossPay Text property |
| btnClose_Click | Terminates the application |

Slide 1- 34

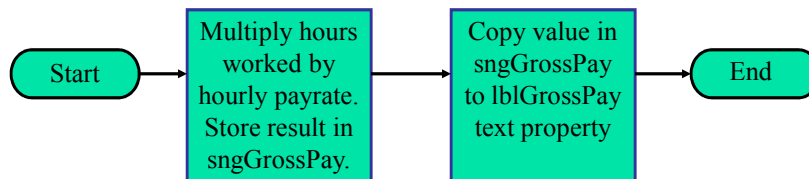


Step 6 of Developing an Application

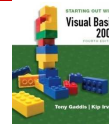
- Create *pseudocode* **or** a *flowchart* of each method:
 - Pseudocode is an English-like description in programming language terms

Store Hours Worked x Hourly Pay Rate in sngGrossPay.
Store the value of sngGrossPay in lblGrossPay.Text.

- A flowchart is a diagram that uses boxes and other symbols to represent each step



Slide 1- 35



Step 7 of Developing an Application

- Check the code for errors:
 - Read the flowchart and/or pseudocode
 - Step through each operation as though **you** are the computer
 - Use a piece of paper to jot down the values of variables and properties as they change
 - Verify that the expected results are achieved

Slide 1- 36

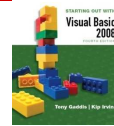
Step 8 of Developing an Application



- Use Visual Basic to create the forms and other controls identified in step 3
 - This is the first use of Visual Basic, all of the previous steps have just been on paper
 - In this step you develop the portion of the application the user will see

Slide 1- 37

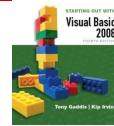
Step 9 of Developing an Application



- Use Visual Basic to write the code for the event procedures and other methods created in step 6
 - This is the second step on the computer
 - In this step you develop the methods behind the click event for each button
 - Unlike the form developed on step 8, this portion of the application is invisible to the user

Slide 1- 38

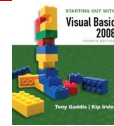
Step 10 of Developing an Application



- Attempt to run the application - find syntax errors
 - Correct any syntax errors found
 - *Syntax errors* are the incorrect use of an element of the programming language
 - Repeat this step as many times as needed
 - All syntax errors must be removed before Visual Basic will create a program that actually runs

Slide 1- 39

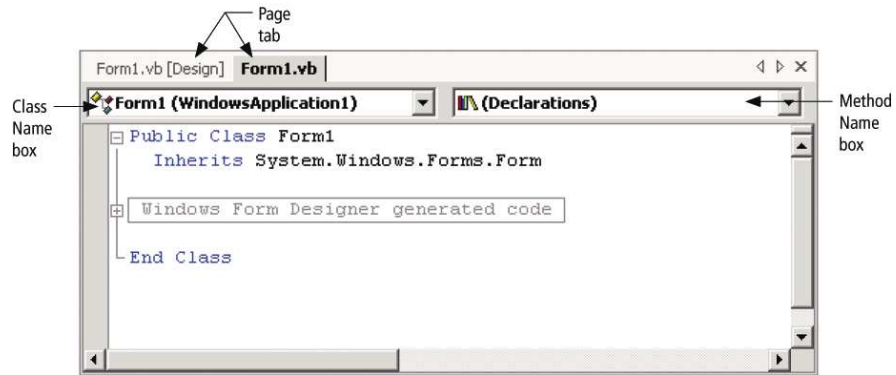
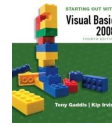
Step 11 of Developing an Application



- Run the application using test data as input
 - Run the program with a variety of test data
 - Check the results to be sure that they are correct
 - Incorrect results are referred to as a *runtime error*
 - Correct any runtime errors found
 - Repeat this step as many times as necessary

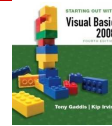
Slide 1- 40

Program Region



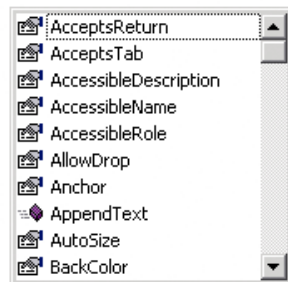
Slide 1- 41

IntelliSense

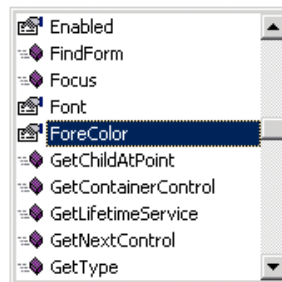


Automatically pops up to give the programmer help.

txtFirst.



txtFirst.For



Slide 1- 42

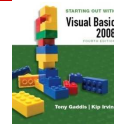
Event Driven Programming: Events



- The GUI environment is *event-driven*
- An event is an action that takes place within a program
 - Clicking a button (a Click event)
 - Keying in a TextBox (a TextChanged event)
- Visual Basic controls are capable of detecting many, many events
- A program can respond to an event if the programmer writes an *event procedure*

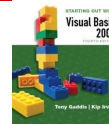
Slide 1- 43

Adding Code to an Event



- To add code for an event:
 - In the VB Code Window select the control on the left side menu and the event of interest on the right side menu
 - Or double-click the control in the designer to bring up the most common event for that control
- Other methods for opening the Code window:
 - If the Code window is visible, click on it
 - Double-click anywhere on the Form window
 - Select the Code option from the View menu
 - Press the F7 method key anywhere on the design form
 - Select the View Code icon from the Project Window

Slide 1- 44



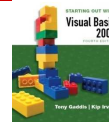
Event Procedures - Subroutines

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

For now you can ignore most of this, aside from knowing the name of the subroutine:

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

Slide 1- 45



Structure of an Event Procedure

```
Private Sub objectName_event(...)  
    Handles objectName.event  
    statements ' Your code goes here  
End Sub
```

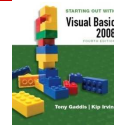
Slide 1- 46



The Text Property of a TextBox

- A user can change the text property of a text box simply by typing in the text box
- A programmer can change the text property of a text box with an assignment statement
 - Uses the form Object.Property just as we did to change the text property of a label
 - The following code assigns the text to the left of the equal sign to the text property of the text box txtInput
 - `txtInput.Text = "Type your name"`

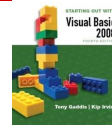
Slide 1- 47



Changing the title of the form in code

- The following won't work:
`Form1.Text = "Demonstration"`
- The current form is referred to by the keyword *Me*.
`Me.Text = "Demonstration"`

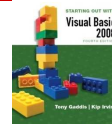
Slide 1- 48



In-Class Walkthrough

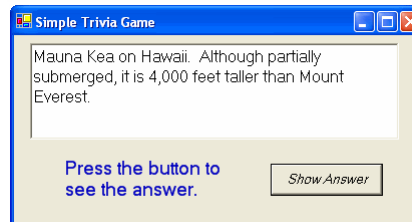
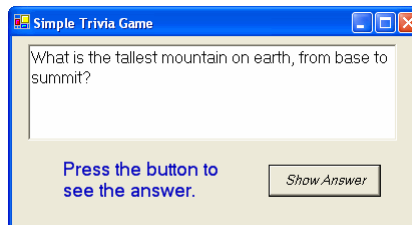
- Create a form with a textbox, button, and label
- Upon clicking the button, store some text in the label and change the color of the button

Slide 1- 49



In-Class Exercise

- Write a program to do something like this:



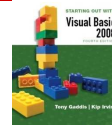
Slide 1- 50



The Text Property of a TextBox

- We can use the text property of a text box to retrieve something the user has typed
 - The following code assigns the text in txtInput to the text property of the label lblSet
 - `lblSet.Text = txtInput.Text`
 - Once again we use the form Object.Property
 - This is the typical means to refer to a property of any object

Slide 1- 51



Clearing a TextBox

- Can be done with an assignment statement:
 - `txtInput.Text = ""`
 - Two adjacent quote marks yields a null string
 - So this replaces whatever text was in txtInput with "nothing" -- a string with no characters
- Can also be done with a method:
 - `txtInput.Clear()`
 - Clear is a *Method*, not a *Property*
 - Methods are *actions* – as in clearing the text
 - Uses the form *Object.Method*

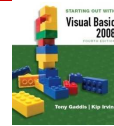
Slide 1- 52



The MessageBox.Show Method

- The MessageBox.Show method is used to display a box with a message for the user
- The message box also contains a title and an icon
- General forms of the MessageBox.Show method
 - `MessageBox.Show(text)`
 - `MessageBox.Show(text, caption)`
 - `MessageBox.Show(text, caption, buttons)`
 - `MessageBox.Show(text, caption, buttons, icon)`
 - `MessageBox.Show(text, caption, buttons, icon, defaultbutton)`
- To do: Add a MessageBox.Show to the button click event
 - Hard-coded text, `textbox.text`

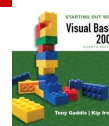
Slide 1- 53



Console.WriteLine

- Another handy way to output information is to the Console:
 - `Console.WriteLine("Hello there")`
 - Outputs the message in double quotes and adds a newline
 - `Console.Write("Hello again. ")`
 - Outputs the message in double quotes without a newline
- Useful for debugging, don't have to push the OK button and clutter up the screen with message boxes

Slide 1- 54



Load Event Procedure

- Every form has a *Load event procedure*
- Automatically executed when the form is displayed
- Double-click in any empty space on the form
- The code window will appear
- Place the code to be executed between the Private Sub and End Sub lines

```
Private Sub Form1_Load(ByVal sender As System.Object, _  
    ByVal e As System.EventArgs) Handles MyBase.Load  
    ' Code to be executed when the Form loads  
End Sub
```

Slide 1-55