

Visual Basic - Beginning

Intro to Visual Studio 2005

Notes	Activity
Text References	Read Chapter 2

• Programs	Pages	4 – 6
• Microsoft Visual Studio 2005	Pages	20 – 26
• Solutions, Projects and Source files	Pages	53-54
• Starting a new Project	Pages	42 – 52
• IDE	Pages	26 – 32
• Controls	Pages	9 – 13, 47
•		100 – 101
• Forms	Pages	42 – 43
• Labels	Pages	44 – 47
		76 – 77
• PictureBoxes	Pages	48 – 50
• Buttons	Pages	64 – 65
• Code Window	Pages	66 – 68
• Help System	Pages	80 – 86

Creating Computer Programs

- To create your own programs for a computer, you need a programming language. There are dozens (hundreds?) of programming languages, including COBOL, RPG, Visual Basic, Java, C++ and C#.
- You use the algorithm/pseudocode you developed in the design phase to convert your logic into the programming language statements.
 - Computers can not understand these statements, they only understand 0's and 1's.
 - The **compiler** converts the programming language statements into machine language.
 - The compiler also informs you if you have broken any of the rules of the language.
- Today, most compilers come packaged with other tools
 - editors to key in your statements
 - debuggers to help locate errors in your program

Notes

Activity

Procedural versus Object-Oriented Programming

Procedural Programming Languages:

- Break the logic down into major tasks called procedures.
- Program determines and controls the order the computer processes the tasks
- Examples: COBOL, RPG, Procedural C++

Object-Oriented Programming Languages:

- Focus is on the objects the program can use to accomplish its goal/task.
- Objects are treated as independent units.
- Objects can be used in more than one program with little change.
- Many predefined objects available

Objects

An object is anything you can see, touch or use – a thing.

- Common objects: label, button, checkbox, menu
- Other objects: can represent something in real life – person, employee, building and a purchase order.

All objects have:

- Attributes/properties – characteristics that describe them
- Behaviors – methods and events
 - Methods – action the object can perform
 - Events – action that the object responds to such as clicking a button or when a textbox control changes.

Objects are created from classes. A **class** is:

- Pattern or blueprint used to create an object.
- All objects of the same class have the same properties and behaviors of that class.

Notes	Activity
-------	----------

About Microsoft Visual Studio 2005

In this class (and Programming Logic - Intermediate), we'll be using Microsoft Visual Basic 2005 (which is part of the Visual Studio software). This software comes with an editor, a compiler, a debugger and a help system to help you learn the language. These components together are called the *integrated development environment* (IDE).

Visual Basic 2005 runs in Microsoft .NET Framework 2.0.

.NET framework is

- Platform on which applications are created
- Goal is to connect information, people, systems and devices.
- Supports VB, C++, C# (C sharp) and J#.
- Contains the common IDE
- Contains common set of classes shared by the above languages:
 - Prewritten code for Windows components (form, toolbars, textboxes, etc).
 - Common utilities – math utilities, string manipulation.
- .NET Framework must be installed on computers that run/use Visual Studio languages.

Notes	Activity
-------	----------

Solutions, Projects and Source Files

- Visual Studio NET includes the opportunity to combine many projects into a *solution*.
 - Solutions are containers that can hold many projects.
 - Each project could be written in a different language
 - Used for large, multi-person system development
 - For our purposes, solutions will only have one project (but that one project must still be included in a *solution*)
 - File extension for the solution is *.sln*
- All the source files for one program are combined into a *project*
 - All project files are written in the same language.
 - Projects must belong to a solution.
 - The project folder is created and stored in the Solution folder.
 - File extension for the project is *.vbproj* . The project will be stored in a folder with the same name as the solution.
 - Other files are created in the project folder as well.
- VB.NET programs include many *source files*.
 - Code required to create forms and manipulate the objects on those forms
 - Utility code used by more than one form or more than one program.
 - File extension for the Visual Basic source files is *.vb*
- All source files are combined to form a project which is part of a solution (always).

Notes	Activity
<p data-bbox="186 233 792 268">Starting Microsoft Visual Studio 2005</p> <p data-bbox="186 275 1084 342">To start the process of developing your own program, you'll first have to start Microsoft Visual Studio 2005</p> <ol data-bbox="240 348 1084 680" style="list-style-type: none"><li data-bbox="240 348 1084 493">1. Open the Start Menu, select and click Microsoft Visual Studio 2005.<ul data-bbox="293 426 1084 493" style="list-style-type: none"><li data-bbox="293 426 1084 493">• Note: there will be slight differences if you are using Visual Studio 2005 Express versus Visual Studio 2005.<li data-bbox="240 499 1084 680">2. Once you have the application running, you will see a tabbed window - Start Page. The screen will vary from the book and from the first time and consecutive times you enter the IDE.<ul data-bbox="334 611 1084 680" style="list-style-type: none"><li data-bbox="334 611 1084 680">• If you do not have the Start Page, click <i>View</i> menu option, click <i>Other Windows</i> and then click <i>Start Page</i>.	<p data-bbox="1122 233 1425 268">Start Visual Basic 2005</p>

Notes	Activity
-------	----------

Customizing the Visual Studio IDE



- Tools ▶ Options on the menu
- Option Strict
 - Projects and Solutions ▶ VB Defaults
 - Forces VB to use *strict syntax* rules
 - Turn on
 - Discussed later
- Default Solution Folder
 - Projects and Solutions ▶ General
 - Designates the where solutions and folders are stored by default
 - Suggest H: drive or Desktop.
 - Pen drives may reduce performance.
 - Work on Desktop, copy to pen when done
- Save New Project
 - Projects and Solutions ▶ General
 - By default, VS doesn't save new projects until you run or save manually. Doesn't even create the folder
 - Check *Save new projects when created*
- Tab Settings
 - Text Editor ▶ Basic ▶ Tabs
 - By default, VB indents code 4 spaces. Three (3) saves space.
 - Leave Smart Indenting on
- Remove Start Page
 - Environment
 - Show all settings
 - Startup
 - Change *At startup*:
 - Constantly closing Start Page gets to be a pain
 - Suggest *Show empty environment*
 - If launch project from My Computer, that project will be loaded
 - If launch Visual Studio, can still open projects from the File menu
 - If want to see Start Page use the menu View ▶ Other Windows ▶ Start Page
- Setting *stick* even after you close Visual Studio
 - Will need to set these on all computers you use.

Notes	Activity
<p>Starting a New Project</p> <p>Each application you create within Visual Studio is called a project . We are now ready to start a new project.</p> <ol style="list-style-type: none">1. Under <i>Recent Projects</i>, click on <i>Project...</i> next to <i>Create</i>.2. In the <i>Project Types</i> pane on the left, ensure <i>Visual Basic</i> is selected.3. In the <i>Template</i> pane, select <i>Windows Application</i>.4. In the <i>Name</i> textbox, you will enter the name of your project. Do not keep the default of <code>WindowsApplication1</code>.5. Change the <i>Name</i> to something meaningful to describe the project. (Note: follow standards/guidelines identified by the instructor).6. You can close the <i>Start</i> window if you want by clicking the <i>X</i> when the <i>Start</i> tab is selected. Be careful not to close the entire IDE.	<p>Create a new project called Sample</p> <p>Show Solution, Project and files generated</p>

Notes	Activity
-------	----------

Common Windows and features in the IDE

The IDE contains 5 common windows:

- Start page – create and open projects, access information on how to use Visual Studio 2005, latest news and technical articles.
- Solution Explorer – displays the names of projects and files included in this solution (application). Appears on the top right portion of the IDE.
 - Shows all the projects in the solution (only 1 for our purposes) and the source files included in the projects
 - Windows Form files are designated with this icon 
 - Code files are designated with this icon 
 - Using the Solution Explorer Toolbar (from left to right)
 - Properties: makes the properties window active
 - Show All Files: shows the files generated by Visual Studio. (We will use this in the next unit)
 - Refresh: Refreshes the current project
 - View Code: displays the Code window
 - View Designer: displays the Design Window for that active form
- Toolbox – displays objects/controls you can use/add when creating a project. The toolbox tab appears on the left side of the screen.
- Properties – displays properties associated with the active object/control of the form.
- Database Explorer – displays data connections and servers (will be used in Programming Logic Intermediate)

Other features:

- Menu – provides access to activities you can do within the IDE.
- Toolbar – buttons under the menu; can be moved, hidden, displayed and customized.

Locate each Window

In Solution Explorer:

Select the Show All files and View Code/View Designer

Notes	Activity
-------	----------

Saving the project

Save Sample

Save your work often to prevent loss of changes. You can save the project by:

- Option 1: Select *Save All* from *File* menu.
 - The *Location* textbox shows where the project folder will be created. You can change this by clicking *Browse* button and select the drive/folder location.
 - The *Solution Name* will be the folder the project will be stored in. Keep the Solution and project names the same.
 - Then click *OK*.
- Option 2: Press Ctrl-Shift-S on keyboard and follow above steps for location and Solution name.
- Option 3: Click *Save All* button on the toolbar.
- Option 4: Run the program. By default Visual Studio saves all the project files before running the program.

Closing the project

Close Sample

If you did not do a *Save All*, you will be prompted to Save the solution. In the *Save Project* dialog box, complete the following.

1. The *Location* textbox shows where the project folder will be created. You can change this by clicking *Browse* button and select the drive/folder location.
2. The *Solution Name* will be the folder the project will be stored in. Keep the Solution and project names the same. Then click *OK*.

If you saved the project earlier but made changes, you will be prompted to save the project.

Opening Existing Projects

Open Sample

You can open an existing project in (at least) 4 ways:

1. Locate the *.sln* file using Windows Explorer (My Computer) and double-click it
2. Open *Microsoft Visual Studio 2005*. While on the *Start Page*, select the project listed in the *Recent Projects* list. The project will open (unless it was moved to a different location).
3. Click *Open Project...* option and locate the project in the Open Project dialog box. Select the *solution file (.sln)* and click *Open*.
4. Click *File* in the menu, click *Open Project* in the file menu and browse for the project using the *Open Project* dialog box. Select the solution file and click *Open*.

Notes	Activity
<p>Managing the IDE</p> <p>As mentioned above, there are five common windows open in the IDE. More windows can be open depending on what activity you are doing.</p> <ul style="list-style-type: none"> • To close an open window, click the <i>Close</i> button • To open a window, use the <i>View</i> menu • To auto-hide a window, click the <i>Auto Hide</i> button (looks like a push pin) • To display an auto-hidden window, place the mouse pointer on the window's tab. • To permanently display an auto-hide window, click the <i>Auto Hide</i> push pin. • Typically, do not hide the <i>Solution Explorer</i> and <i>Properties</i> windows. Hide the <i>Toolbox</i> after you have developed the form because you will not need this window out as much. • Resize a window <ul style="list-style-type: none"> - Move cursor to the edge of the window and the sizing pointer appears for you to drag the window to the needed size. - You can the <i>Properties</i> window (and other windows) longer by placing the cursor at the top of the window and dragging the title up. This makes it easier to work with changing the objects properties. 	<p>Practice closing, opening and auto hiding windows</p>
<p>Windows Form Designer Window</p> <p>You create/design the GUI for the project using the Windows Form Designer.</p> <ul style="list-style-type: none"> • The form object is the foundation/container for the user interface in Windows applications. • Add objects (controls) from the <i>ToolBox</i> (such as labels, buttons and textboxes) to the form. The form is an object itself. • If the form is not displaying, double click on the <i>Form1.vb</i> (or associated name) in the <i>Solution Explorer</i> to display the designer view. 	<p>Display the properties of the form in alphabetical and then category order</p>

All objects have properties that describe the object. As you design your application, you can change the properties using the *Properties Window*. The properties can be ordered alphabetically or by category.

Notes	Activity
<p>Changing Source File Name</p> <p>A source file is a file that contains program instructions or <i>code</i>. The Form1.vb source file is known as the <i>form file</i>.</p> <ul style="list-style-type: none">• Contains the code associated with the Windows <i>form object</i> or <i>form</i>.• Change the default name of the source file from <i>Form1.vb</i> to something <u>more meaningful</u>. There are two ways possible:<ul style="list-style-type: none">- Click on Form1.vb in the Solution Explorer to get focus. Click again to be put in edit mode to rename the form.- Click on the Form1.vb to get focus. In the properties window, change the <i>Name</i> property.- REMEMBER: include the <i>.vb</i> extension.- DO NOT change the solution, project or source file names directly from Windows. Use the IDE to change the names. The appropriate directories in the application will not get updated if you do not go through the IDE.- Use meaningful names that describe the purpose of the form. (Ex: WelcomeScreen, EmployeeEntry, UpdateScreen)	<p>Change the name to frmSample.vb</p>

Naming Objects

Rename the form to frmSample

Objects that are **referenced in code or by an event**, will need to be named. If the object requires an event procedure or the properties of the object are changed in code – rename the object from the default value.

1. Click on the object to select it
2. In the properties window, change the (*Name*) property.
3. Enter a descriptive name for the object
 - Our programming standards dictate all object names should begin with a prefix.
 - The name begins with the 3 character prefix in lower case followed by descriptive name to identify the control. The first letter of each word of this name is capitalized.
 - Examples:
frmSample, lblName, txtAddress, btnExit, picLogo
 - Refer to the class standard prefixes for a complete list.

Note: If an object is NOT referenced in code or by an event, you can keep the default name.

Notes	Activity
-------	----------

Form

Properties

- *Text* – changes what appears on the form's title bar
- *BackColor* – changes the background color of the form
 - Note: when form's background color is changed, the background color of all objects on the form whose background color was not set will adapt the background color of the form.
- *Icon* – set the icon that appears in the form's title bar
- *StartPosition* - where the form appears on the screen (normally – CenterScreen)
- *Font* – set the font (type, size, etc) for all objects on the form (unless those objects properties are set)
- *ForeColor* - set the font color for all objects on the form
- *BackgroundImage*
 - Select an image to display on the form's background.
 - Use the *backgroundImageLayout* for desired look.
 - Note: your project now has a folder called *Resources* into which the source image has been copied.

Change these properties on the Sample form

Text – *Class Sample*

BackColor – select a color of choice

Icons: select an icon from the class folder

BackgroundImage: select from class folder

Changing Properties of Controls

As a control is introduced in the notes, the common properties that are changed for that control are presented. Note: if you change properties and want to reset them back to the original value, just right-click on the property and select *Reset*.

Complete
Checkpoint pg 57

Notes	Activity
<p>Adding Controls</p> <p>The toolbox contains the standard tools to add to forms. In OOP terms, the tools represent a class. When you add an object to the form, we call them controls.</p> <p>There are 4 ways to add a control –</p> <ul style="list-style-type: none"> • Option 1: <ul style="list-style-type: none"> - Double-click a tool in the toolbox • Option 2: <ul style="list-style-type: none"> - Click a tool in the toolbox and drag the pointer to the form. - Release the mouse button when the pointer is on the form. • Option 3: <ul style="list-style-type: none"> - Click the tool and then click the form • Option 4: <ul style="list-style-type: none"> - Click the tool, then press the mouse pointer on the form and press the left mouse button to drag the pointer until the control is the desired size 	<p>Add 3 labels – using each of the 3 methods mentioned</p>
<p>Manipulating Controls</p> <p>Once the control is on the form, it can be moved, deleted, sized, select multiple controls and aligned multiple controls.</p> <ul style="list-style-type: none"> • Moving Controls <ul style="list-style-type: none"> - Place mouse over object - When the pointer changes to a four-headed arrow (move arrow to get it to change), click and drag the object to the new location. • Deleting Controls <ul style="list-style-type: none"> - Click on the control to gain focus and hit the delete button - Or, click on the control and select the <i>Cut</i> toolbutton. 	<p>Move labels</p> <p>Delete a label</p> <p>Add the label again</p> <p>Change the size of each label</p> <p>Select all 3 labels – try each option</p>

Notes	Activity
<ul style="list-style-type: none"> • Sizing Controls <ul style="list-style-type: none"> - Click on the control to select it. - Place the mouse over one of object sizing handles (the small white squares around the middle of each edge of the object). <ul style="list-style-type: none"> ➤ Note: Small handles on the corners are used to adjust both height and width - Drag handles to resize the object. 	<p>Move labels as a group</p> <p>Deselect labels</p> <p>Move labels so they are not aligned</p>
<ul style="list-style-type: none"> • Select Multiple Controls <ul style="list-style-type: none"> - Option 1: <ul style="list-style-type: none"> ➤ Click on the first control, then Ctrl-Click or Shift-Click additional objects to select them. ➤ Ctrl or Shift-Click the object again to deselect it. ➤ NOTE: the first object selected controls the alignment (has white handles) - Option 2: <ul style="list-style-type: none"> ➤ Point to an empty area near the first object ➤ Drag a box that encompasses all the objects you want selected. ➤ Best used when objects are near each other - Option 3: <ul style="list-style-type: none"> ➤ Press Ctrl-A to select all objects on a form 	<p>Now – align labels using the grid</p> <p>Complete Checkpoint pg 53</p>
<ul style="list-style-type: none"> • Aligning Controls <ul style="list-style-type: none"> - When you drag a control to a position on the form, a blue guide/grid line automatically appears when the controls are aligned. - Guide line only appears when controls are in line. - Guide can be used to align vertically and horizontally. - Option 2: <ul style="list-style-type: none"> ➤ Select controls you want to align ➤ Click on <i>Format</i> menu option, select <i>Align</i> and select desired alignment - Option 3: <ul style="list-style-type: none"> ➤ With the <i>Layout</i> toolbar showing (to display – select <i>View / Toolbar / Layout</i>) ➤ Click on toolbar for the type of alignment desired. 	

Notes	Activity
<p>Labels</p> <p>Labels are used to: describe other objects on the form (<i>what should be entered in this text box?</i>) and display information to the user (such as output results or directions/remarks that don't change)</p> <ul style="list-style-type: none"> • Add a label object to the form (see above) • Labels used to describe other objects or display information that does not change – ARE NOT renamed (default name used) because code rarely refers to them. • Labels used to display results will be used in code and should have a descriptive name (lbl prefix). <p><u>Properties:</u></p> <ul style="list-style-type: none"> • <i>AutoSize</i> property of a label is set to True by default. <ul style="list-style-type: none"> ➤ Makes the label adjust to the size of the Text property contents. ➤ Gives the control a dotted-line rectangle called <i>bounding box</i>. This marks the tightest rectangle that contains all of the control and you <u>can not</u> manually change the size of the label. ➤ To change the size, set the <i>AutoSize</i> to False • <i>Text</i> - controls the words that appear on the label • <i>Image</i> - to add a picture to the label • <i>ImageAlign</i> - control where on the label the image appears • <i>TextAlign</i> - control how the <i>text</i> aligns on the label (often in relation to an image location) • <i>Font</i> - change the appearance of the label's text: font, style and size. • <i>ForeColor</i> - change the color of the text on the label • <i>BackColor</i> - change the color of the label itself <ul style="list-style-type: none"> ➤ Labels are often transparent so the form's background shows through. ➤ Note: If a background Image was put on the form, the label takes on the background color and is not transparent. • <i>BorderStyle</i> - set the kind of border the label has around it. <ul style="list-style-type: none"> ➤ Labels describing another control don't have a border ➤ Labels displaying output - set to <i>FixedSingle</i> • <i>Visible</i> - indicates if label can be seen on the form 	<p>Add a title to the form and set it's text properties (<i>This program was written by:</i>)</p> <p>Note: same color as form background</p> <p>Change: Font</p>
<p>• <i>AutoSize</i> property of a label is set to True by default.</p> <ul style="list-style-type: none"> ➤ Makes the label adjust to the size of the Text property contents. ➤ Gives the control a dotted-line rectangle called <i>bounding box</i>. This marks the tightest rectangle that contains all of the control and you <u>can not</u> manually change the size of the label. ➤ To change the size, set the <i>AutoSize</i> to False <p>• <i>Text</i> - controls the words that appear on the label</p> <p>• <i>Image</i> - to add a picture to the label</p> <p>• <i>ImageAlign</i> - control where on the label the image appears</p> <p>• <i>TextAlign</i> - control how the <i>text</i> aligns on the label (often in relation to an image location)</p>	<p>Add another label and put you're name in the text. Place under the title label</p>
<p>• <i>Font</i> - change the appearance of the label's text: font, style and size.</p> <p>• <i>ForeColor</i> - change the color of the text on the label</p> <p>• <i>BackColor</i> - change the color of the label itself</p> <ul style="list-style-type: none"> ➤ Labels are often transparent so the form's background shows through. ➤ Note: If a background Image was put on the form, the label takes on the background color and is not transparent. <p>• <i>BorderStyle</i> - set the kind of border the label has around it.</p> <ul style="list-style-type: none"> ➤ Labels describing another control don't have a border ➤ Labels displaying output - set to <i>FixedSingle</i> <p>• <i>Visible</i> - indicates if label can be seen on the form</p>	<p>Add another label with the course name by copying the first label.</p> <p>Add MSTC Logo.gif as label image. Experiment with alignment</p> <p>Add a border to this label</p>

Note: many objects have the same properties (Font, BackColor, Left, Top, etc). After having been described once in the these notes, they will not be described again.

Notes	Activity
<ul style="list-style-type: none"> • TIP <ul style="list-style-type: none"> - If you need many objects that have the same appearance (e.g. many labels with the same font size) <ul style="list-style-type: none"> ➤ Create one object and set its properties appropriately ➤ Copy the object to the clipboard ➤ Paste a copy of the object onto the form and position it appropriately ➤ Don't forget to rename the copy - Alternatively, after you've finished formatting one object, press the Ctrl key, point to the object and drag it to a new location <ul style="list-style-type: none"> - A small plus sign should appear next the mouse pointer to designate you're adding (copying) a control - The new control will have the same format as the original but will be assigned a default name. - You can also place the objects on the form, select them all and change the properties for all of them at once. 	<p>Experiment with getting the Font properties for the labels with text to be the same. Try doing this with both controls selected.</p>

PictureBox

Pictures improve the appearance of the form when used appropriately.

NOTE: The form itself can have a background image but it can not be resized or repositioned.

- Add a PictureBox to the form

Properties:

- *Name* - use **pic** prefix
- *Image* - designate which image should be displayed
 - You can use animated GIFs in picture boxes. If you want to stop the animation, set *Enabled* to *False*
- *SizeMode*
 - Normal: Picture is aligned top-left, as much of the image as possible displays
 - CenterImage: the image centers itself in the box, as much of the image as possible displays
 - AutoSize: the size of the PictureBox adjusts to accommodate the entire image
 - StretchImage: the image will distort to completely fill the PictureBox
- Size and position as necessary
- Commonly modified properties:
 - [Left](#), [Top](#), [Height](#), [Width](#), [BackColor](#), [BorderStyle](#)

Add a PictureBox to the form

Set the *SizeMode* to *StretchImage*

Assign the *bookImage* to the form

Size the PictureBox to increase the image size

Experiment with size mode

Experiment with the visible property

Notes	Activity
<p>Buttons</p> <p>Buttons allow users to designate when program actions should take place</p> <ul style="list-style-type: none">• Add a button to the form <p><u>Properties:</u></p> <ul style="list-style-type: none">- <i>Name</i> - use btn prefix- <i>Text</i> - change the words that appear on the button <ul style="list-style-type: none">• Size and position as necessary• Commonly modified properties: Left, Top, Height, Width, Font, ForeColor, BackColor, Image, ImageAlign, TextAlign	<p>Add an OK button to the form with the Exit icon and the text OK</p> <p>Experiment with text alignment</p>
<p>Code Window</p> <p>Everything you create using VB.NET generates code.</p> <ul style="list-style-type: none">• A lot of code is generated automatically by VB.NET<ul style="list-style-type: none">- Code to create a form- Code for every object placed on the form• To view the code for a form, double-click on any object on the form or on the form itself.• The event most applicable to the object you double-clicked will be shown. The code template consists of the Sub procedure header line and the End (last) line.	<p>Double-click the form object to open the code window</p>

Notes	Activity
<ul style="list-style-type: none"> • Code Features <ul style="list-style-type: none"> - Works a lot like a word processor <ul style="list-style-type: none"> ➤ Cut, copy and paste ➤ Drag to move or copy (Ctrl-drag) - Color coding to make code more readable <ul style="list-style-type: none"> ➤ Reserved words in blue ➤ Comments in green - Auto-indent <ul style="list-style-type: none"> ➤ VB.NET uses the indentation of the previous line to set the indentation of a new line ➤ Press Tab to indent further, back space to move to the previous indent - Auto-End <ul style="list-style-type: none"> ➤ VB.NET automatically inserts End statements for decision and loop constructs (appropriately indented) - Auto-List Members <ul style="list-style-type: none"> ➤ Whenever you refer to an object or a class, VB.NET shows you the members and methods of that class as soon as you press a period (.) ➤ Select the value using the <i>Tab</i> or <i>Spacebar</i> - Error Highlighting <ul style="list-style-type: none"> ➤ Just like when you make a spelling error in MS Word, VB.NET highlights syntax errors (language errors) by underlining the erroneous code with a blue wavy line. - Unused variable highlighting (discussed later) <ul style="list-style-type: none"> ➤ Green wavy line 	<p>Enter sample code to change the form's background color.</p> <p>Me.backcolor (Note auto member list) (Note color coding)</p> <p>Me.backcolor = Color.red (Note auto member list)</p> <p>Press enter (Note auto indent)</p>

Adding Code to a Control

Objects know how to handle *events*, things that users do to the object.

- The most common type of event is the **Click** event that occurs when a user clicks an object
- To keep our first application simple, we are going to add some simple code to our button's Click event
- To add code to an object's event handler, simply double-click the object
 - The code editor will open in a new tab and VB.NET will create a procedure (a small section of code) header for the most commonly accessed event for this object (often the Click event).
 - Type the appropriate VB.NET statements/commands for this event.

Double-click on the btnOK

Add
Me.Close with a comment

Notes	Activity
Switching Views	Try it

You will be switching from the Code Editor to the Form Designer often.

- Simply click the appropriate tab at the top of the design window ([Design] designates form design)
- If you're in form design, it's often more convenient to simply double-click an object to get to the Code Editor
- You can also use the *Solution Explorer* window. Click the *View Designer* button to switch to the *Design* window. Click the *View Code* button to switch to the *Code* window.

Syntax Rules

Syntax or language syntax, defines the correct way to use a specific programming language. Syntax errors occur when you don't follow the rules – Visual Basic can't understand the command you're trying to give. If a programmer violates Visual Basic syntax, the application will not run until it is corrected.

Most syntax errors are typing errors.

A nice feature of Visual Studio is it checks for syntax errors as you are writing your code. When you move to the next line of code, the previous line is checked. If there is an error, a *blue wavy* line appears under the part of the statement that is in error.

- Position your mouse on the blue squiggly line and an error message appears.
- Fix the error and click off the line to verify the code is correct.

If you do not fix the error and try to run the application, you will not be able to run the application until the errors are fixed. (Refer to the Testing/Running section below)



If you do not have syntax errors but your program does not work as designed, you have **logic errors**. You need to review the code to verify you entered the statements correctly.

In the frmSample_Load event, key in

BackColor
And hit enter

Notice the blue squiggly line.

Put cursor on blue line to see error

Notes	Activity
<p>Testing / Running a Program</p> <p>During the project development process, you will want to test your program to make sure it is working as planned. During the testing process, the code you wrote is compiled to make sure the statements follow the rules of the language called syntax rules.</p> <p>Syntax or language syntax, defines the correct way to use specific programming language. Syntax errors occur when you don't follow the rules – Visual Basic can't understand the command you're trying to give. If a programmer violates Visual Basic syntax, the application will not run until it is corrected.</p> <p>Most syntax errors are typing errors. If you do not have syntax errors but your program does not work as designed, you have logic errors. You need to review the code to verify you entered the statements correctly.</p> <p>To test and run a VB application:</p> <ul style="list-style-type: none"> • Click the <i>Start Debugging</i> button on the toolbar  or press the F5 key • If there are remaining <u>errors</u> in your code (referred to as Build Errors by VB.NET), VB.NET will let you know and you'll have to fix them before you can run the program. <ul style="list-style-type: none"> - An <i>Error List</i> window will display to identify the errors. - Double click on an error and you will be taken to the appropriate line of code. - Evaluate the code and correct the error • To stop a running program <ul style="list-style-type: none"> - close its window by clicking the X in the upper right corner OR - click the Stop button  on the Debug toolbar <ul style="list-style-type: none"> ➤ If the Debug toolbar is not visible, right-click any toolbar and select it from the popup menu - If the Immediate Window remains visible, close it 	<p>Click the Start debugging button</p> <p>Click the Stop debugging button</p> <p>Change the code in the form load to be:</p> <pre>BackColor = red</pre> <p>This will cause a blue squiggly line</p> <p>Click the start debugging – you will get a message box indicating there are errors. Answer no to see the Error list window.</p> <p>Fix errors Run the program and then Close it.</p>

Notes	Activity
-------	----------

Adding Comments to Code

Comments are descriptions of the project as a whole or code in the project, added by the programmer, to further describe what the code does.

- To add a comment in VB.NET, type an apostrophe (') followed by the descriptive comment
- The apostrophe can appear anywhere in a line of code (beginning or middle) but all text after the apostrophe is ignored.
- The **comments will appear in green**

Complete
Checkpoint pg 70
and pg 73

Types of Program Documentation

- *Application* level – describes the Name of the project, programmer name, date, purpose and any changes made to the application.
 - For now, enter the general program documentation in the code for the main form.
 - Enter the comments after the *Public Class* statement
 - Use the following general program documentation for all projects in this course:

Project Name: Sample.vb
Programmer Name: Programming Logic Student
Date: Month day, year
Purpose: Provide a short explanation of the project's purpose. You must use complete sentences.

Change Log:

Date of change1 – description of change 1
Date of change2 – description of change 2

- *Form-level* - form level documentation identifies the procedures (for now – just events) within the form and the purpose of the procedure.
- *Variables and Constants* level – for each variable and constant (described in next unit), provide a purpose for the variable/constant using inline comments.
- *Code* level – for more complex blocks of code, provide internal documentation describing what the logic does.

Notes	Activity
-------	----------

Regions Directive

You have the ability to group your code into regions. This features will collapse and expand code within the region.

- Begin with `#Region " identifier-name"`
- Ends with `#End Region`

```
#Region "DataValidation"  
    ' Insert code for the Data validation logic here.  
#End Region
```

- Consider using regions to contain your application and form documentation

The Final Touch

Hopefully, when you created and customized your form, you assigned an icon to the form. This icon appears in the form's title bar and in the Windows toolbar when the form is active.

However, there is one more place you need to define an icon—the project itself. The project icon displays on the executable file of your program. Each of the project's forms and the project itself can have unique icons.

Note: You'll find your program's executable file in the Bin, Debug folder of your project folder.

To add a project icon:

- Locate the icon you'd like to use for your project using Windows or an external icon management program.
 - There are lots of icons on the I: drive
- Copy the icon to your solution folder
- Double-click My Project in the Solution Explorer
- Drop down the Icon: list.
- Choose the Browse option
- Locate your icon
- The icon should appear in the My Project window

Notes	Activity
-------	----------

Object Oriented terms

This section introduces you to object-oriented terms you will read or hear about.

- *Class* - a pattern or blueprint used to create an object.
 - Every object used in an object-oriented program comes from a class.
- *Instance* – objects created from a class are referred to as an instance of the class and said to be “instantiated” from the class.
 - You can have multiple instances of an object from the same class in your application. (Ex: you can have 3 labels on your form, you can have 2 buttons on your form).
 - Each object is uniquely referenced by its name.
- *Encapsulation* – hiding all the properties (data) and behaviors (methods and events of an object) in one package. By doing this the programmer can use the class without having to know exactly how the class is coded. This is known as *information hiding*. (Example: you can drive a car without having to know how the engine, brakes and exhaust work internally)
- *Inheritance* – you can create one class from another class. The new class inherits the attributes and behaviors of the original class.
- *Polymorphism* - an object-oriented feature that allows the same instructions to be carried out differently depending on the object and the situation. For example: you can open a door but you can also open your eyes, open a jar or open a box.

Notes	Activity
<p>Help System</p> <ul style="list-style-type: none">• The Visual Studio Help System is a valuable tool. Learn to use it!!• <i>Dynamic Help</i> is only available in Visual Studio 2005.<ul style="list-style-type: none">- To display <i>Dynamic Help</i>, select <i>Help</i> menu, then <i>Dynamic Help</i>.- A new window is displayed providing the following Help tools<ul style="list-style-type: none">– How Do I ...– Search– Index– Contents• If you are using Visual Basic Express, you access the <i>Help</i> system through the <i>Help</i> menu and then select from the above Help tools (listed above).<ul style="list-style-type: none">- The Help system is displayed in a separate Window so you can move between your application and the Help information.	<p>Complete Using Visual Basic Help – Tutorial 2-17</p> <p>(Note: Visual Basic Express may differ slightly from Visual Studio)</p> <p>Complete Checkpoint pg 87</p>