

Visual Programming CS202

Chapter One

INTRODUCTION To Visual Basic

Objective

Course objective – This course focuses on Windows programming using visual Basic to create graphical user interfaces. The course examines forms, controls, graphical controls, loops, control arrays, database and traditional file processing, and OOP.

- Be able to demonstrate knowledge of interface design principles and be able to apply them in a visual programming environment.
- Have a working knowledge of object oriented concepts and how to implement them in a visual programming environment.
- Be able to apply database linkages.
- Create a windows application in a visual programming environment.
- Create an Internet (web) application in a visual programming environment.

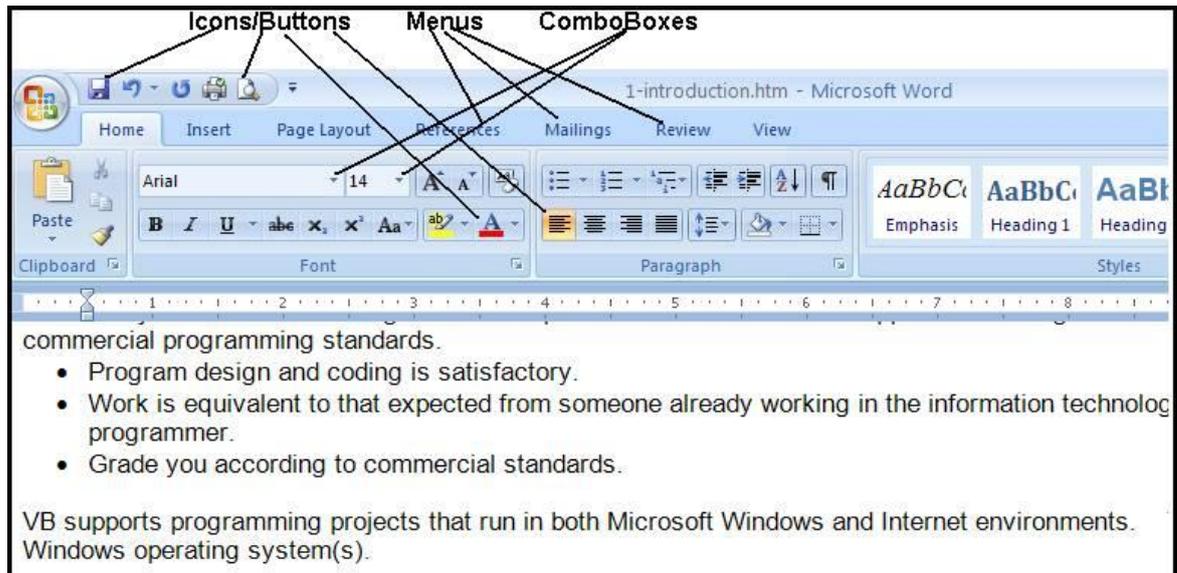
VB supports programming projects that run in both:

- Microsoft Windows and,
- Web (Internet) environments.

Commercial GUI Design/Development Approach

Modern programs run in a **Graphical User Interface (GUI)** environment. You can open up any desktop program or web browser and examine the **GUI**. This interface includes:

- Buttons (some with Icons and Graphical Images) and Menus on which you point and click.
- TextBox controls in which you type information.
- Graphical images to guide a system user in using the program.
- Other components.
- A GUI for a version of Microsoft Word is shown in this figure.



Microsoft Visual Studio and Visual Basic (.NET Version)

Programming Languages

Microsoft's **Visual Studio** (also called **Visual Studio.NET**) includes several different programming languages:

- **Visual Basic**,
- **Visual C#** (C sharp),
- **Visual C++** (cee-plus-plus),
- **Visual F#** (F sharp),
- **JScript** and
- Web Development (called **ASP.NET**).
- It also includes the **.NET 4 Framework** upon which these languages operate.

All of these languages **compile**.

- This means they are translated from **human readable-form** to **machine readable-form** to the same **Microsoft Intermediate Language (MSIL)**.
- MSIL run within the **Common Language Runtime (CLR)** – a component of the .NET Framework.

Visual Basic (also termed Visual Basic.NET or VB) is a major revision of earlier Microsoft VB products.

- This version is Version 10 of VB.
- This is an upgrade of earlier **.NET** versions.
- The current and earlier versions are not completely compatible—a program coded with an earlier version that is upgraded to the current version cannot be opened again with the earlier version software.
- Version 6 and earlier are not **.NET-compatible** – programs created with these versions are completely incompatible with **.NET** applications.

VB is available in several editions including the free **Express Edition** that you can download from Microsoft. Other editions (**Professional**, **Premium**, and **Ultimate**) are used within industry.

- Our classroom has the **Visual Basic 2010 Express Edition** installed.

Visual Studio Components

When Visual Studio is installed on a computer, there are two mandatory components to the installation and an optional third component.

- **.NET Framework Class Library**. This is a library of predefined class objects. It enables you to quickly build a computer application through the use of predefined objects such as forms, text boxes, labels, buttons, drop-down list controls, and others (mandatory).
- **Common Language Runtime (CLR)**. This component manages the execution of a programming project written in any of the languages that are included within Visual Studio including Visual Basic as a language (mandatory). This component is installed as part of the .NET Framework.
- **MSDN (Help)**. This is the help component and provides access to a help reference library. It is an optional, but highly recommended component.

Object-Oriented Programming Terminology

VB is an *object-oriented programming language*.

- Means you work with **objects** in building an application.
- Examples: Form objects, Button objects, TextBox objects, Label objects, ListBox objects, PictureBox objects, and more.

VB is also termed an *event-driven programming language* because you will write program code that responds to events that are controlled by the system user. Example events include:

- Clicking a button or menu.
- Opening or Closing a form.
- Moving the mouse over the top of an object such as a text box.
- Moving from one text box to another.

In order to work with VB, you need to understand "**object**" terminology as defined in Table 1.

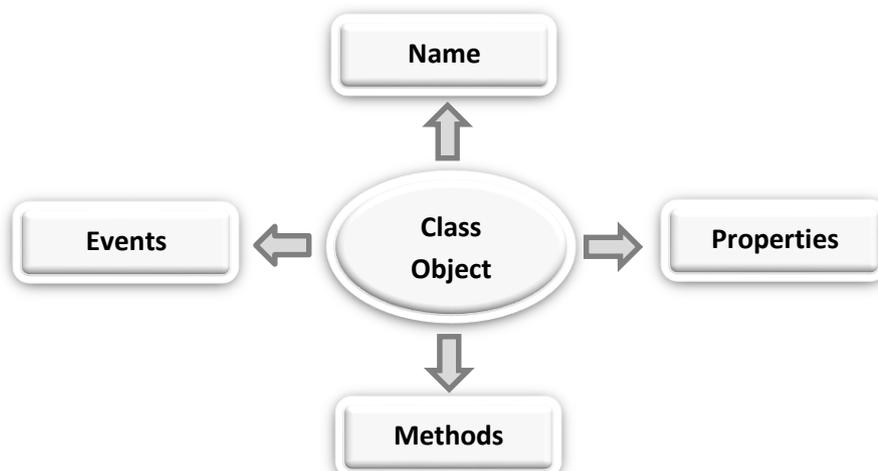


Table 1	
Terminology	Definition
Object	A thing – like a noun in English. Examples include forms and controls you place on forms such as buttons, text boxes, and icons .
Property	Objects have properties – like adjectives in English. Properties describe object behaviors. Examples of properties include Text, Name, BackColor, Font, and Size . Refer to a property by the notation ObjectName.PropertyName (use the .dot notation) – example: TotalDueTextBox.Text or AccountLabel.ForeColor .
Method	Like a verb in English – these are the actions that objects exhibit. Examples include methods to Show and Hide forms and methods to Print and Close forms. Refer to a method with the notation ObjectName.MethodName – example Me.Close will close the current form.
Event	Events are actions usually triggered by the system user such as clicking a button; however, events can also be triggered by the actions of objects. For example, closing a form can trigger an event.
Class	This is a really abstract term – it is a sort of template for an object. For example, all forms belong to the Form class of object. All buttons belong to the Button class of object. Classes include definitions for object properties, methods, and associated events. Each class is assigned an identifying namespace within the .NET Framework Class Library. Each new object you create is defined based on its class – the new object is called a class instance .

Getting Started with Visual Studio

You will use the VB component of Visual Studio to create and test projects.

- The programming applications you will design and develop are called **solutions** in VB.
- A **solution** can actually contain more than one **project**, but in this course we will focus on just creating a single project in a solution.
- Each solution is stored in a folder identified by the solution name.
- This is covered in more detail later in this set of notes.

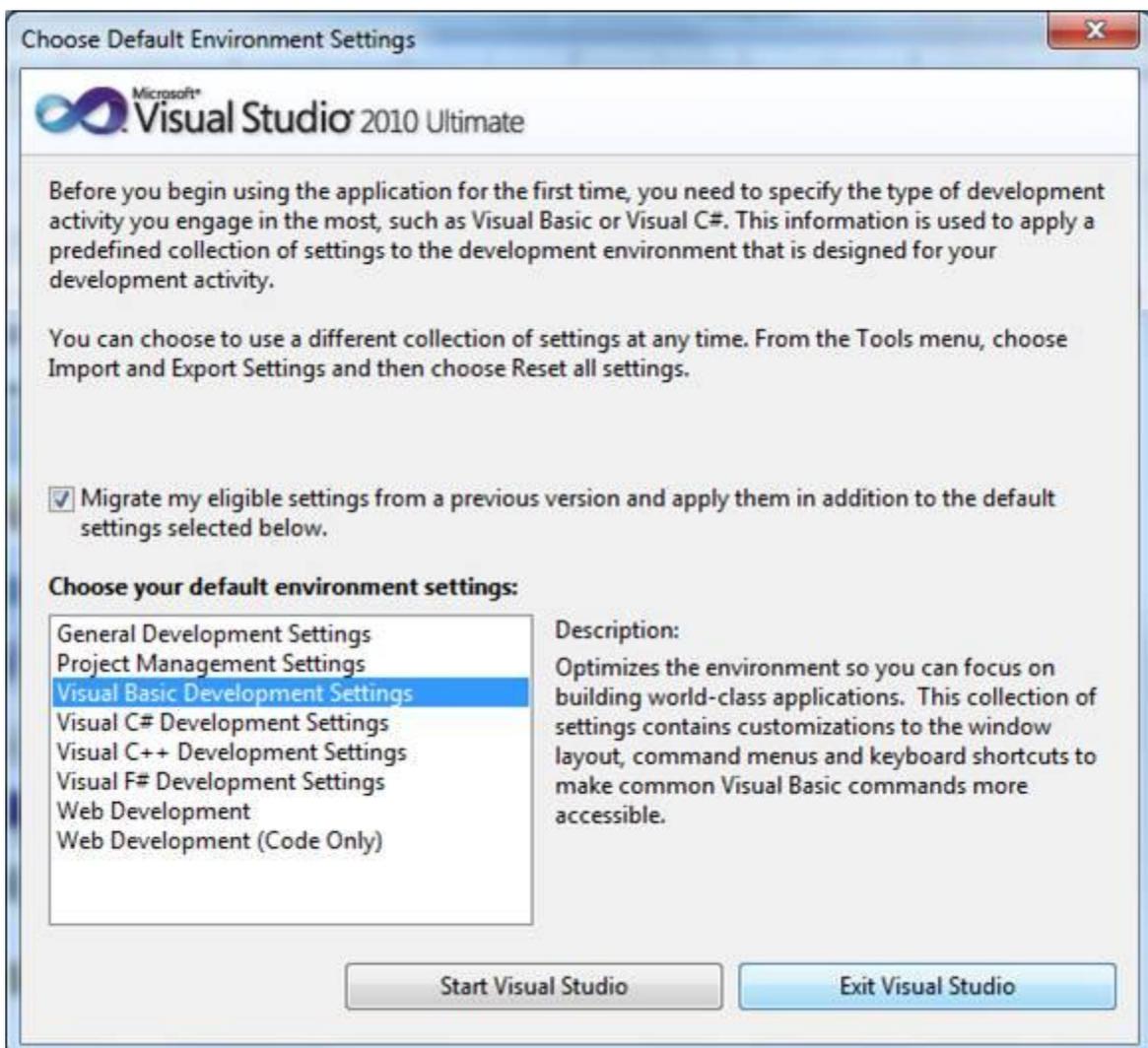
Launching Visual Basic

Start VB – Click the Windows **Start** button and locate **Microsoft Visual Studio 2010** – launch the program by clicking. The option will look something like this.



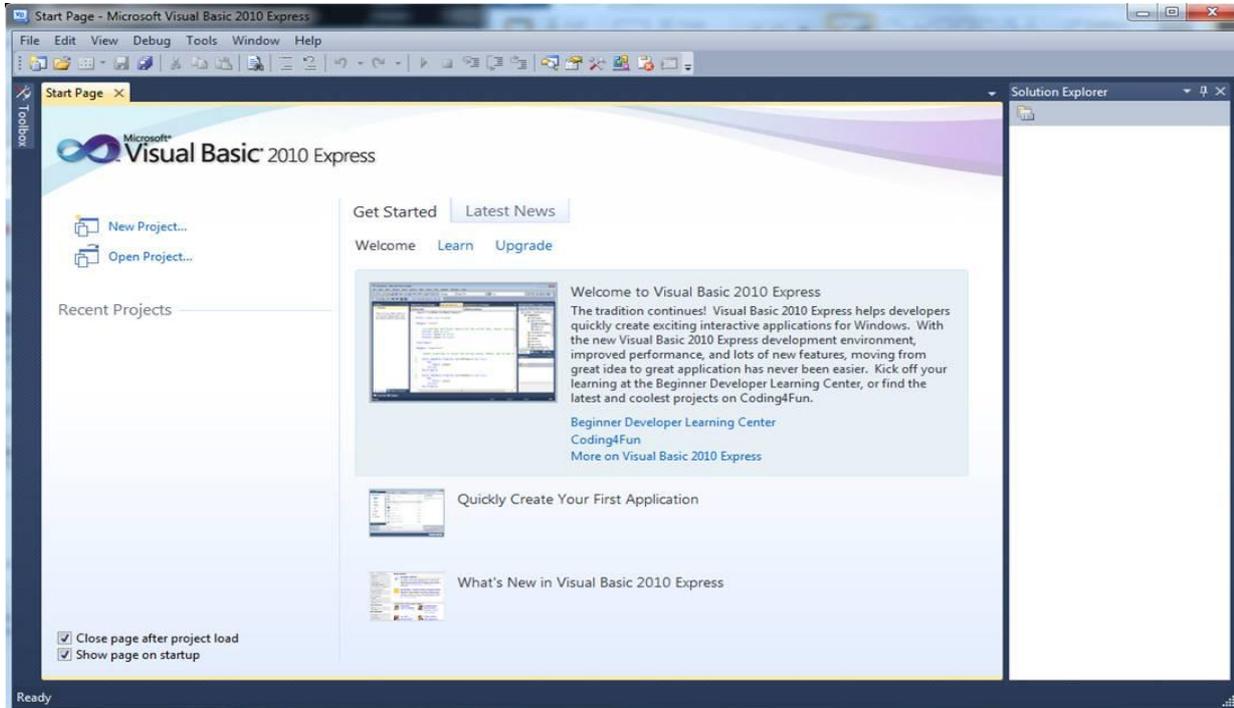
The first screen to display may require you to choose your **Default Environment Settings**.

- If this screen displays, click the **Visual Basic Development Settings** option in the list box and then click the **Start Visual Studio** button

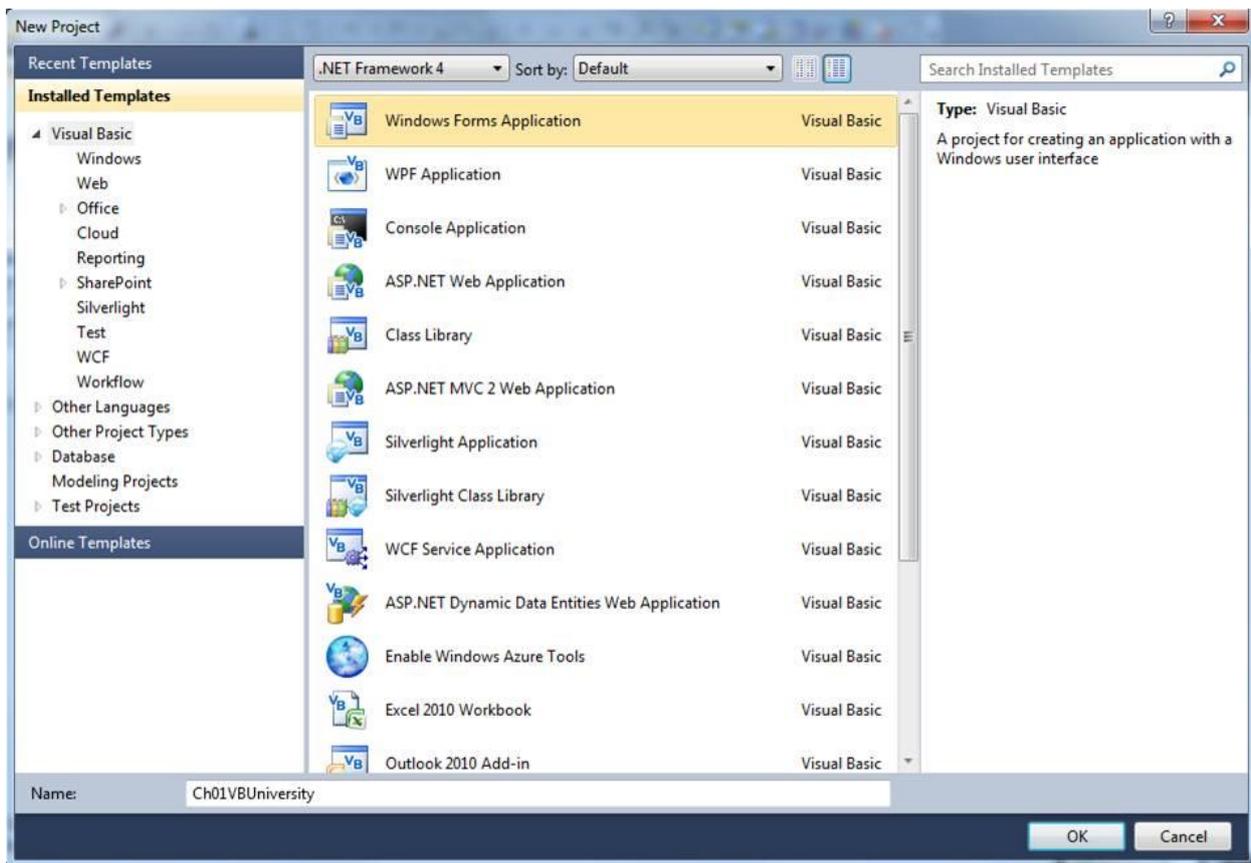


The Visual Studio **Start Page** is shown in the figure below. The start page is slightly different for Visual Studio 2010 (Professional or Ultimate Edition) and Visual Basic 2010 Express Edition.

- This is the Start Page for **Visual Basic 2010 Express Edition**.



Click the **New Project** link shown in the figure above to create a new project. This opens the **New Project** dialog box shown below.



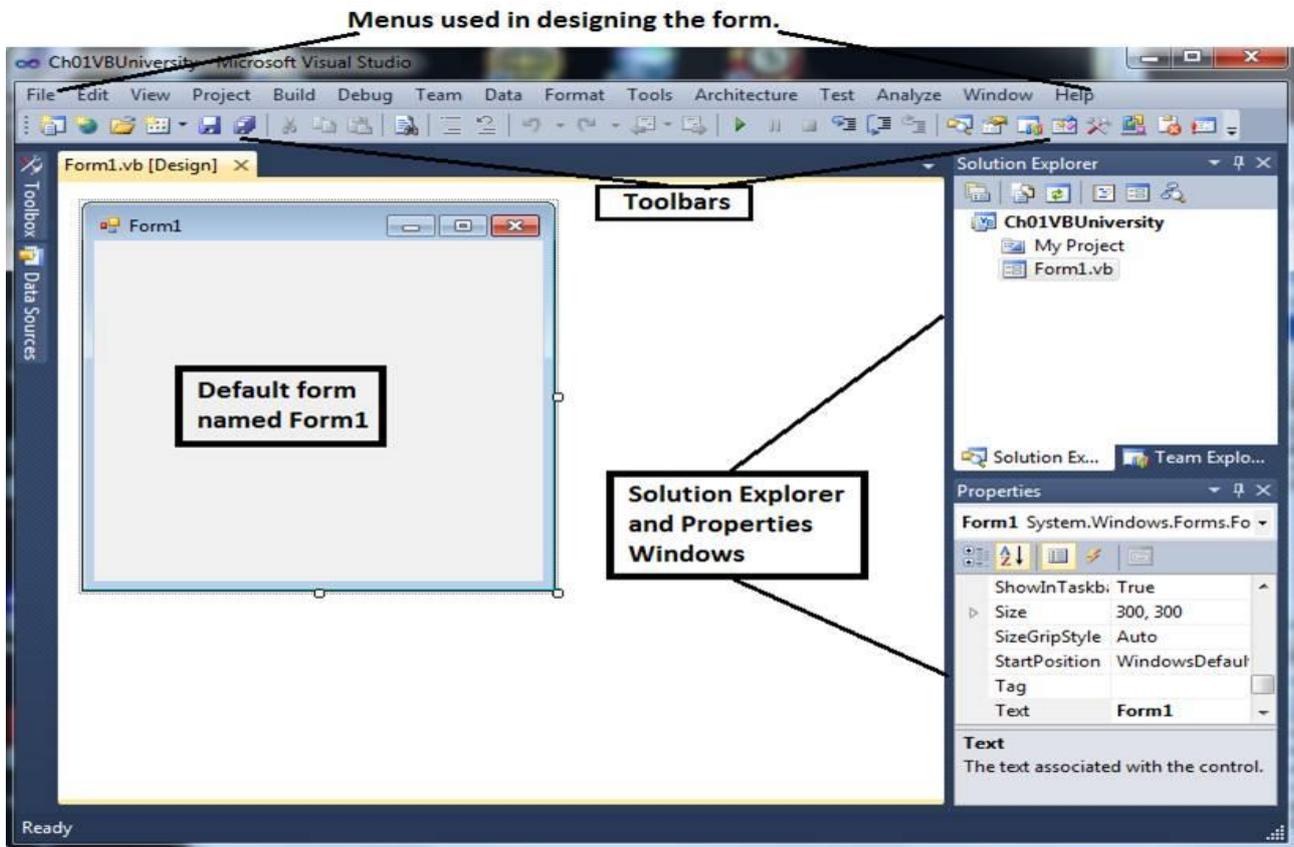
- Your first project will be a **Visual Basic Project** using a **Windows Forms Application** template as shown in the figures above.
- The default name is **WindowsApplication1** -- not a very useful name. Change the project name to: **Ch01VBUiversity**.
- Click the **OK** button – it takes several seconds to create the project files from the template.

The Integrated Development Environment

Familiarize yourself with the **Integrated Development Environment (IDE)**.

- Across the top are menus with different options used in the designing of an application.
- **Toolbars** with shortcut icons are shown below the menus.
- **Form Designer** (also termed the **Document Window**).
 - o Displays open documents such as the **Form1** shown in the figure below.
 - o Tabs along the top are used to allow you to switch rapidly between open documents.
 - o Every project starts with a default form named **Form1** – the form you modify to build your business application.
 - o The form can be resized by using the sizing handles.
 - o The form can be renamed (you'll do this later in an exercise).
- **Solution Explorer Window** – displays filenames for files that comprise a project. The solution name is also shown here (**Ch01VBUiversity**).

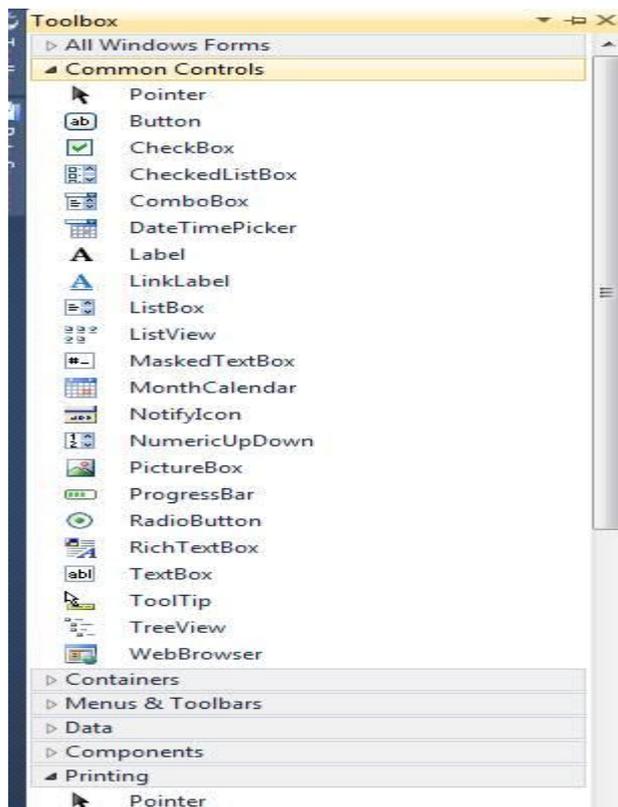
- **Properties Window** – displays properties of the currently selected object – in the figure the properties displayed are those of the Form object.
- **ToolBox Window** – this is shown along the left edge of the figure in collapsed display.



The Toolbox

The **Toolbox**:

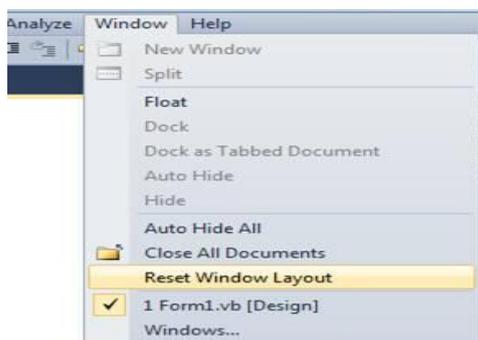
- Contains controls that are used to build forms.
- Can be expanded or collapsed with the **Pin/Unpin** icon.
- Parts of the Toolbox can be expanded/collapsed with the **white and black pointer** icons to the left of the groupings.
- Used to add controls (tools) to a form by either double-clicking or dragging/dropping (your option on which to use).



Resetting the Default IDE Layout

Sometimes you startup VB and the IDE layout is not what you expect.

- Windows you want such as Solution Explorer, Toolbox, or Properties may be closed.
- The quick way to reset the layout is through the **Window** menu, **Reset Window Layout** submenu option as shown in this figure.



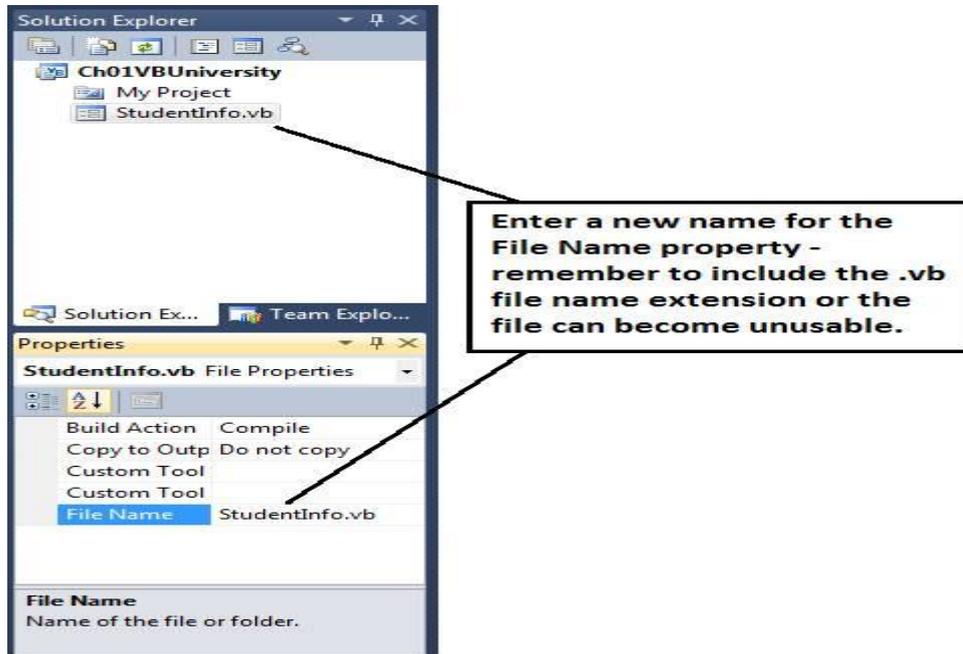
Building the Ch01VBUniversity Application -- Project Specifications

Setup the Project and Form

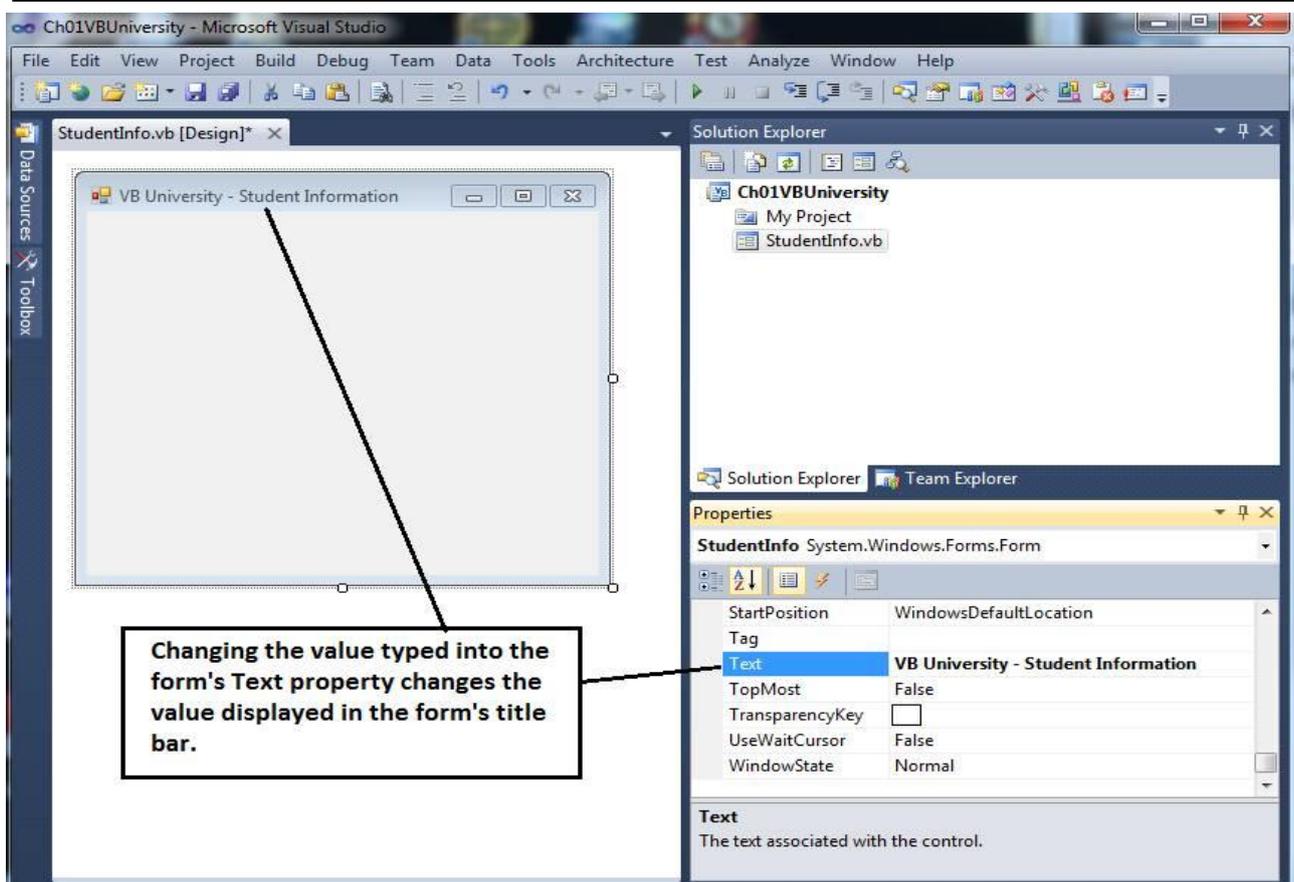
There are several actions you need to take for every project.

- Change the form's **FileName** property.
 - Click the **FileName** in the **Solution Explorer** window – change the **FileName** property in the **Properties** window.
 - A FileName should always end in the **.vb** file name extension.

- Changing the form's **FileName** also changes the **Name** property of the form – select the **Form** and examine the **Name** property (the easiest way to select the form is to single-click the title bar).



- Change the **Title Bar** value of a form by typing a new value into the **Text** property for the form.
 - o Select the Form.
 - o Click the **Alphabetic icon** shown in the figure below to list the properties alphabetically.
 - o Type the new value **VB University – Student Information** in the **Text** property of the Properties window.



- Size the form as needed by clicking the white squares around the form and dragging/dropping.
- Set the form's **StartPosition** property to **CenterScreen**. This will cause the project when it runs to display in the center of the computer monitor screen.
- Set the form's **Font** property – usually you'll leave this at about an **8.25 point font**, but in class I use a **10 point bold font** so everyone in the back of the room can see the overhead display.

Naming Rules and Conventions

Now you are ready to build a form – you will be placing controls on a form and you need to know the naming rules and conventions that you will follow this term in naming controls.

Visual Basic automatically assigns a value to the **Name** property of each control, for example, **Label1**, **Label2**, **Label3**, or **TextBox1**, **TextBox2**, **TextBox3**, or **Button1**, **Button2**, **Button3**. However, it is difficult to remember the difference between Label1 and Label2 so:

- if you are going to later refer to the controls, it is best to rename them to a more meaningful name,
- if you are not going to refer to the controls later, then just use the assigned default name such as **Label1**.

When you name an object such as a Label or TextBox or Button, you must follow these rules:

- An object name can begin with an alphabetic letter or the special “underscore” character.
- An object name can include letters, digits, and underscores.
- An object name **CANNOT** include a space or a punctuation mark.
- An object name **CANNOT** be a VB reserved word such as **Button**, **Close**, or **TextBox**.
- An object name can contain a VB reserved word – object names such as **PrintButton**, **CloseButton**, **NameTextBox**, and **MajorTextBox** are legal names.

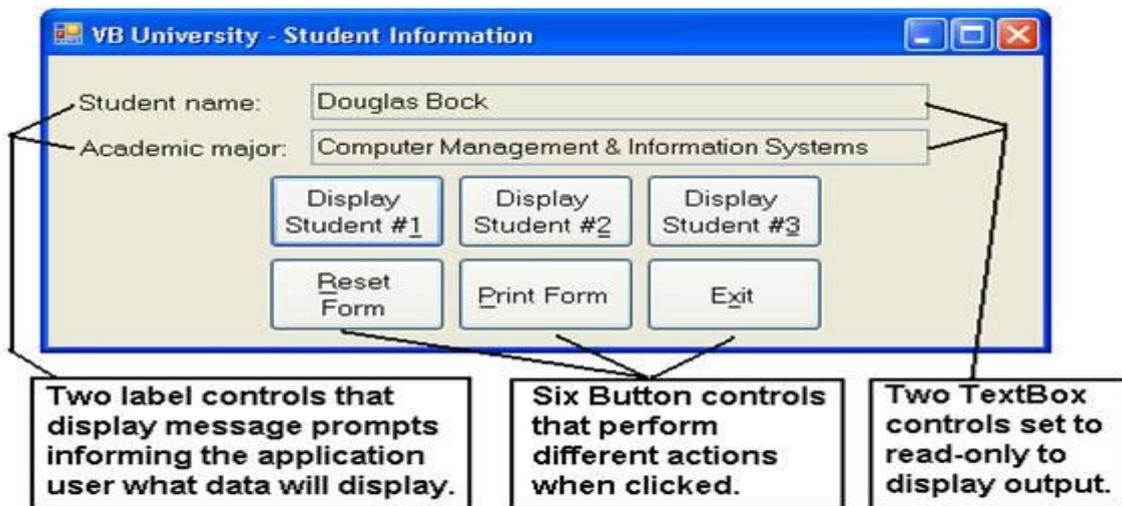
Several naming conventions exist within industry – the common ones are the **Hungarian naming convention** and the **Pascal naming Convention**.

Naming conventions are simply guidelines to help other programmers read your code more easily. We will use the **Pascal naming convention**. The rules are:

- Begin an object name with an uppercase alphabetic character.
- Capitalize each word that is part of an object name.
- Select object names that are meaningful.
- Append the full name of the control class to the end of the name.
- Avoid abbreviations unless they are standard abbreviations such as SSN (social security number).
- Examples of valid names: **MajorTextBox**, **ResetButton**, **MessageLabel**, **TotalDueTextBox**, and **CloseButton**.

This table gives examples naming conventions.

Hungarian Naming Conventions		
Form	Frm	FrmMain, FrmStudent, FrmTeachers
TextBox	Txt	txtName, txtMajor
Button	Btn	btnShipping, btnExit, btnReset
Label	Lbl	lblName, lblOutput
Listbox	Lst	LstName
Image	Img	ImgBox
Option Button	Opt	OptSelect
Note: Label controls are often not renamed – they are not referred to later in writing computer code so the default assigned name is unchanged.		

Build a Form Like This One

1. Open the **Toolbox** – select the **Common Controls** node.
2. Place a **Label** control on the form by either double-clicking the **Label** tool or by drag/dropping a label to the form.
 - o Notice that when you select the Toolbox's **label** control, the mouse pointer on the form becomes a **crosshair** showing you where the upper left corner of the label will be displayed.
 - o Drag the new label to where you want it in the upper left corner of the form.
 - o Change the label's **Text** property to **Student Name:**
 - o The label also has a **Name** property – the first label is named **Label1**. This is a satisfactory name and you will not refer to the label when you later write programming code so leave it named **Label1**.
3. Add a second label control on the form. Set the **Text** property to **Academic Major:** – this label is named **Label2**.
4. Add two text box controls as shown.
 - o Name the first text box control **NameTextBox**.
 - o Name the second text box control **MajorTextBox**.
 - o These **TextBox** controls will only display output so set the properties **ReadOnly = True** and **TabStop = False**.
5. Add six button controls as shown.
 - o Name the button controls **Display1Button**, **Display2Button**, **Display3Button**, **ResetButton**, **PrintButton**, and **ExitButton**.
 - o Set the **Text** property of the buttons to display the text shown in the figure above. Use the ampersand (&) symbol as part of the text in order to create button hot keys (hot keys work just like clicking a button when you press the Alt key + the hot key simultaneously).
 - o Examples:

- For the **ResetButton** type the following: **&Reset**
- For the **PrintButton** type: **&Print**
- For the **ExitButton** type: **E&xit**
- For **Display1Button** type: **Display Student #&1**

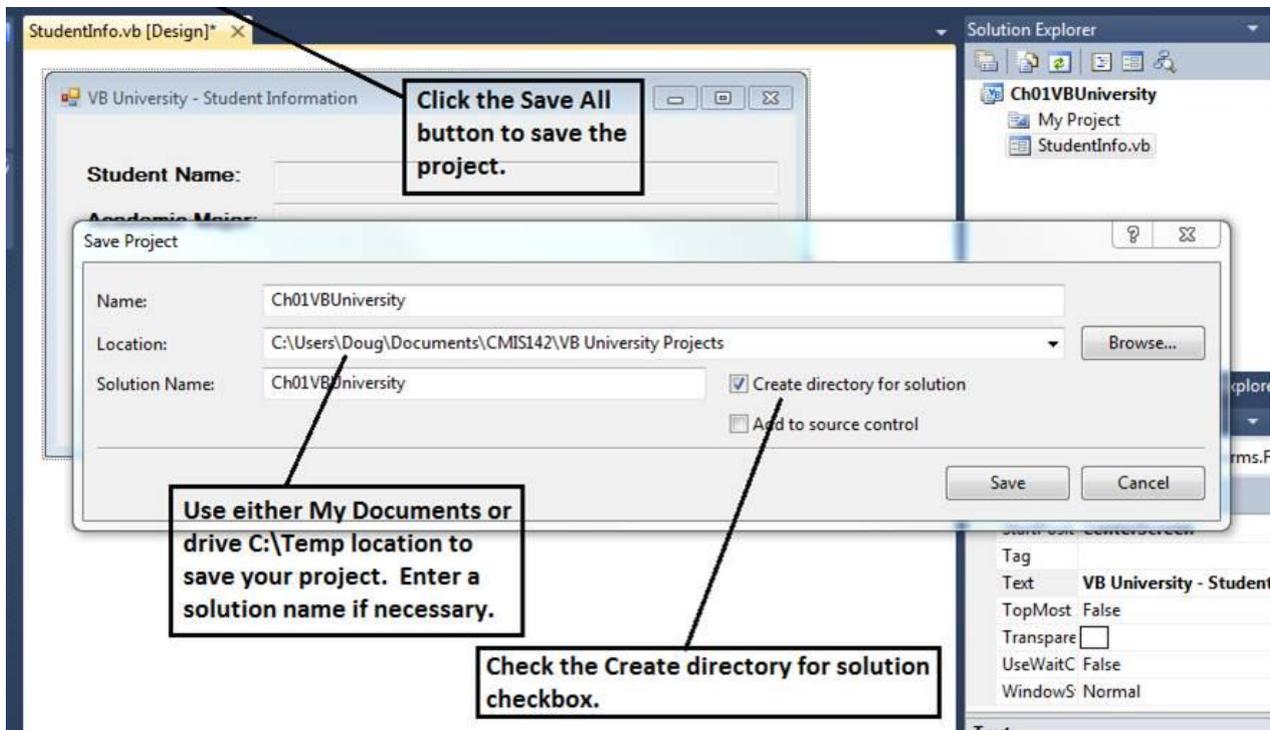
If you accidentally place a control on a form and discover that you don't need the control, simply delete it.

Resize each control to an appropriate size.

Saving and Running a Project

To save a project, use the **File-Save All** menu –**AVOID** using **File Save As** menu options – Repeat, **DO NOT USE File Save As** menu options. You can click the **Save All** button shown in the figure below.

- This will save all files in the project.
- You can specify the location for saving the project. For a location select either:
 - o **My Documents** folder
 - o The **Temp** folder on drive C: (C:\Temp).



There are several ways to run a project to test it.

- Use the **Debug** menu, **Start Debugging** option, or

- Press the **F5** function key to run the project, or
- Click the shortcut **green arrow** on the shortcut toolbar.



Run the project. Confirm that the project starts up centered on the console, and the alignment of controls is professional in appearance.

Name and Text Properties

As you have seen already, forms, labels, text boxes, and buttons have numerous properties. We will focus on learning just a few of these in this chapter.

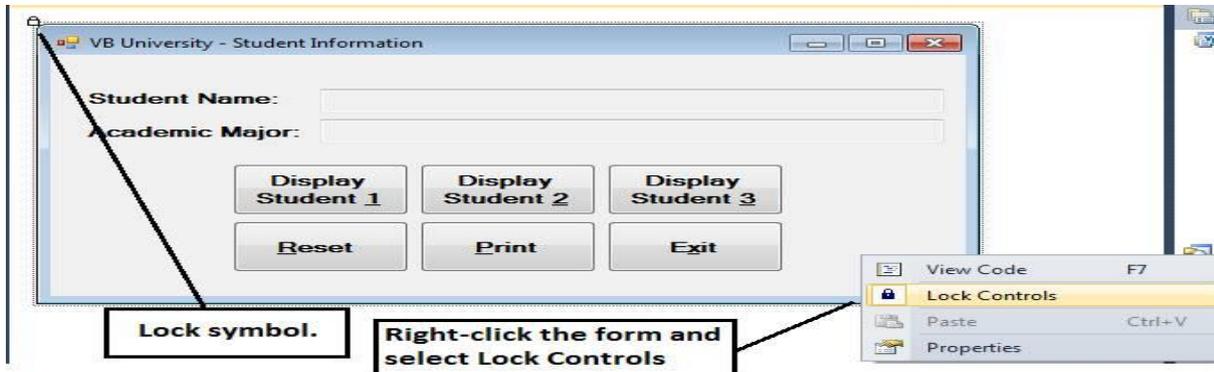
- **Name** property – used to name a control for your reference. For example, we will name the buttons with a name such as **DisplayButton** or **ExitButton**.
 - o Notice that the name begins with a word that describes the control – often you will use the text on a control such as a button as part of the control’s name.
 - o Next the type of control such as **Button** or **TextBox** is added to the name to create a unique name.
 - o Text boxes may be named **NameTextBox** and **MajorTextBox**, or some other meaningful name.
 - o If you're not going to reference a control in the project, you can just use the default name assigned by VB.
- **Text** property – used to display text values on a control.
 - o The **Text** property for a form displays the words that are displayed in the form's Title Bar.
 - o The **Text** property of a button displays the words on the button, such as **Display** or **Exit**. Using the **ampersand** causes a letter on a button to be underlined.
 - o The **Text** property for a text box control displays the actual words typed into the text box.

Locking Controls

After you finish sizing control on a form, you can lock the controls to prevent accidental movement of the controls.

- Right-click any control and select the **Lock Controls** option in the context menu that displays as shown in the figure below.
- Right-click again and select **Lock Controls** to unlock the controls.

You'll notice a small "padlock" icon on the upper left corner of the locked controls.



Program Coding

Programming or writing program code is the means used to cause something to happen when an event, such as a button click occurs. There are six events to be coded for this project – the first three events are for the three display buttons.

- **Click event** for the button named **Display1Button**, **Display2Button**, and **Display3Button**.
- **Click event** for the button named **ResetButton**.
- **Click event** for the button named **PrintButton**.
- **Click event** for the button named **ExitButton**.

You need to master the following coding statements and concepts in this section:

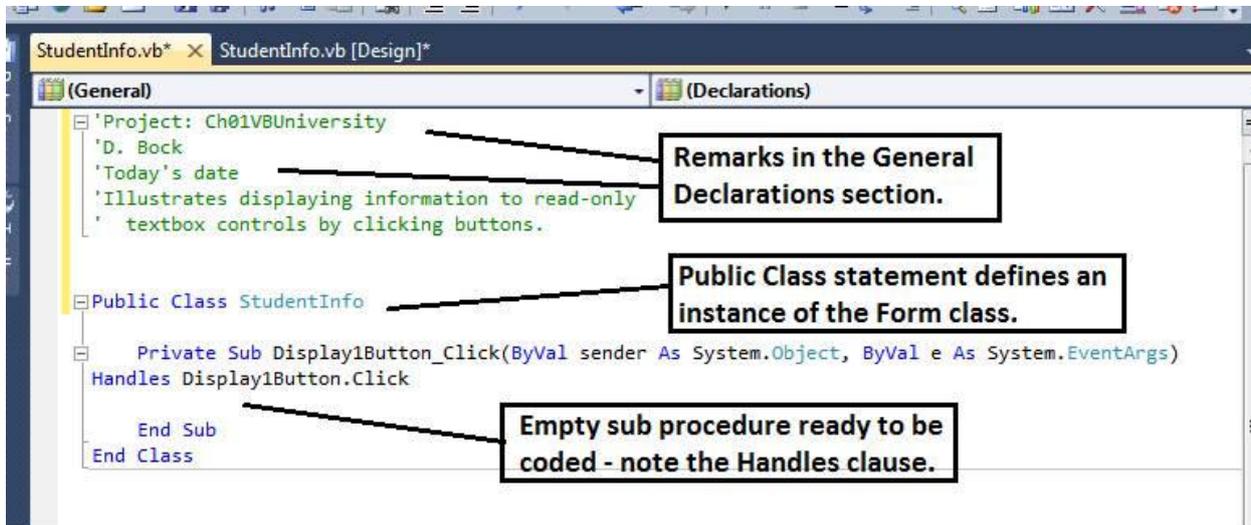
- How to access an event procedure.
- The **Remark** statement.
- Switching between the **View Designer** and **View Code** windows.
- Accessing **Intellisense**.
- The **Assignment** statement.
- The **Clear** method.
- The **Me** keyword and the **Close** method.

Accessing an Event Procedure

Double-click the **Display1Button** control.

- This opens the coding window shown in the figure below.

- o The window is called the **View Code** window.
- o You can also access the code view from the **View** menu or the **View** icon in the Solution Explorer window.
- o The programmer has typed remarks above the **Public Class** statement – remarks begin with a single quote mark and display in green print – remarks are for humans to read – the computer ignores remarks.



Double-clicking the **Display1Button** control causes VB to generate an event sub procedure named **Display1Button_Click** to handle the Click event of the Display1Button control.

- The **Handles** clause indicates that this sub procedure handles the **Click** event of the button named **Display1Button**.
- The sub procedure ends with the line of code **End Sub**.
- Each sub procedure must have a unique name – VB generates a name by combining the name of the control + an underscore + the name of the event, e.g., **Display1Button_Click**.
- Each sub procedure has two **parameters** inside of the parentheses that accompany the event name – these are named **sender** and **e**. For now you can ignore these parameters. Later in the course we will learn to refer to them whenever we need to in order to write more complex code.

The form named **StudentInfo** is created as an instance of the general **Form** object defined in the .NET Framework class library.

- It is declared by the **Public Class StudentInfo** statement.
- The **End Class** statement marks the end of the form's class definition.

The Remark Statement

A **Remark** statement is a line of code shown in green.

- **Remark** statements are non-executable.
- A **Remark** is only read by human beings – VB ignores remarks when running a project. The **Remark** statement is used to document your program code.
- **Remarks** are very useful for large programs. They can be used to embed comments that only programmers will read into programs for later reference.
- Each sub procedure should contain remarks that identify the purpose of the sub procedure.
- A **Remark** can also be placed in-line at the end of a line of programming code.
- **Remarks** are entered in the general declarations area of the program by placing the mouse cursor just before the **Public Class** statement, then pressing the **Enter** key several times to make room to type remarks.
- Here are some example **Remark** statements.

```
'Project: Ch01VBUniversity  
'D. Bock  
'Today's date  
'Illustrates displaying information to read-only  
' textbox controls by clicking buttons.
```

Switching Between the View Designer and View Code Windows

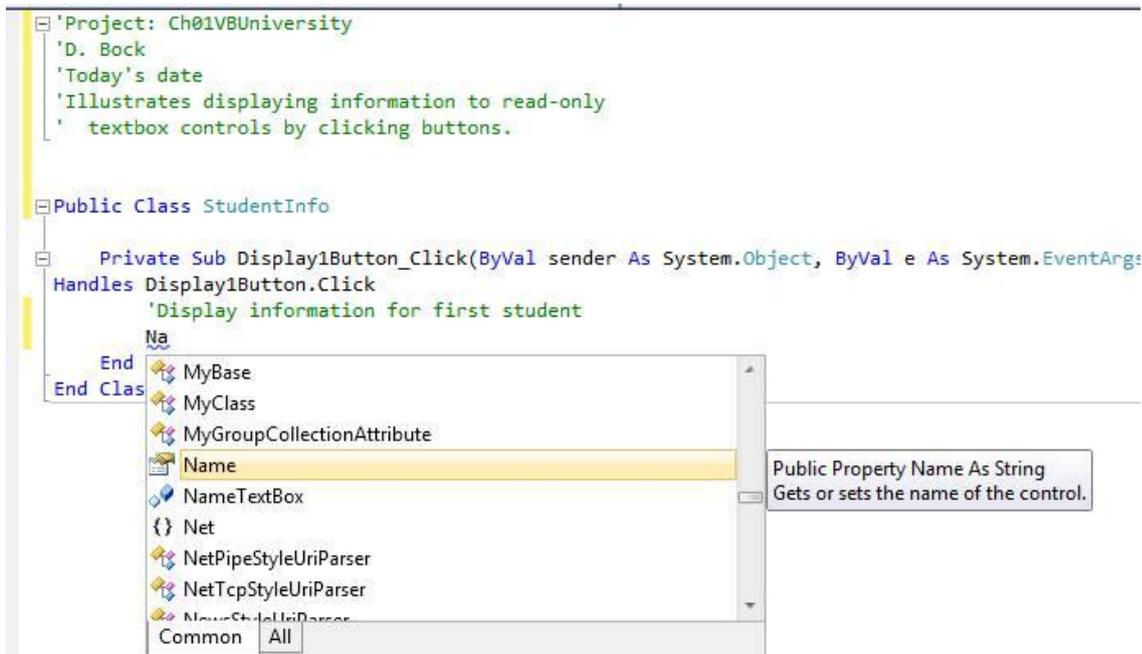
Often you will need to switch between the **View Designer** and **View Code** windows. The different ways to do this are shown in the figure below.

- Click one of the tabs to display either the **View Designer** or **View Code** windows.
- Click the **View Code** icon in the Solution Explorer to open the coding window.
- Click the **View Designer** icon in the Solution Explorer to display the form for additional design layout.
- Click the **View menu** and select either the **Code** or **Designer** submenu options.

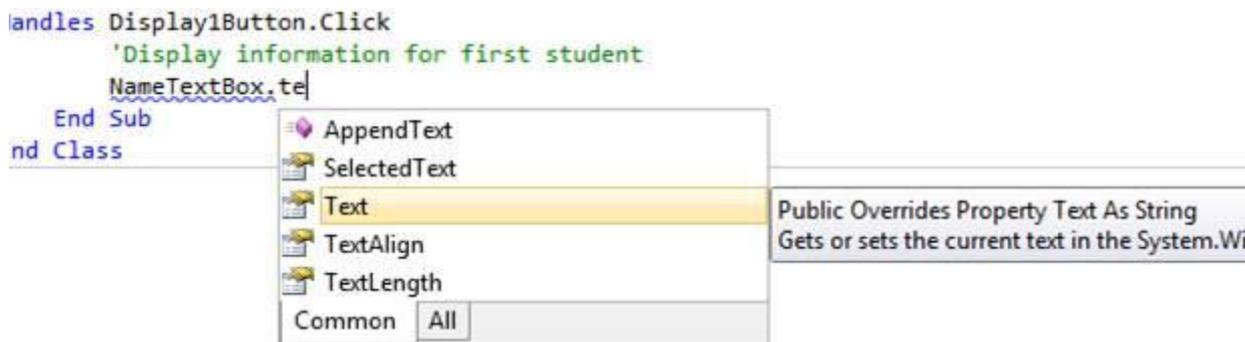
Accessing Intellisense

VB's **Intellisense** feature makes it easier for you to type programming statements.

To access Intellisense simply begin to type the name of an object such as the **NameTextBox** control shown in this figure – VB will pop up a window that displays possible selections – this makes it easier for you to type code and leads to fewer typing errors.



- With the Intellisense focus on the **NameTextBox**, just type a **dot** (period) and the Intellisense will next display all of the properties of the **NameTextBox** – type the letters "Te" (see the figure below) and Intellisense will search and find the **Text** property and set the focus to that property for this text box control.



- Now type an equal sign (=) followed by your name inside double-quotes, e.g., "**Ahmed Ali**".
- The statement you've typed should look like this inside the sub procedure – VB will provide the necessary spacing automatically.
- This is called an **Assignment Statement** and is discussed in detail in the next section.

```

Private Sub Display1Button_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles Display1Button.Click
    'Display information for the first student
    NameTextBox.Text = "Ahmed Ali"

End Sub
    
```

The Assignment Statement

The **Assignment Statement** is used to assign values to a property of an object such as a control.

- The general form of the assignment statement is shown here.
- The value on the right side of the equal sign is assigned to the property of the object on the left side of the equal sign.

```
Object.Property = Value
```

Assign a student name to the **Text** property of the TextBox control named **NameTextBox** and a student's major to the TextBox control named **MajorTextBox**. The assignment statements to do this are:

```
Private Sub Display1Button_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Display1Button.Click
    'Display information for the first student
    NameTextBox.Text = "Ahmed Ali"
    MajorTextBox.Text = "Computer Science & Information Systems"
End Sub
```

Notice that the value is enclosed within **double-quote** marks.

- This indicates the value is a string of characters and only string data is stored to the **Text** property of a TextBox control.
- The value is displayed in brown text.

The Clear Method

The **Clear** method is used to clear the contents of a TextBox control. The general way to execute a method is shown here:

```
Object.Method()
```

Return to the design view and double-click the **Reset Form** button control – this will generate a **Click** event sub procedure for this button. Type a remark and two statements to clear the contents of the two TextBox controls as shown here.

```
Private Sub ResetButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ResetButton.Click
    'Reset the form by clearing the TextBox controls
    NameTextBox.Clear()
    MajorTextBox.Clear()
End Sub
```

The Close Method

The **Close** method is used to close a form. To close a form use the keyword **Me** to refer to the form.

Return to the design view and double-click the **Exit** button control – this will generate a **Click** event sub procedure for this button. Type a remark and a statement to close the form as shown here.

```

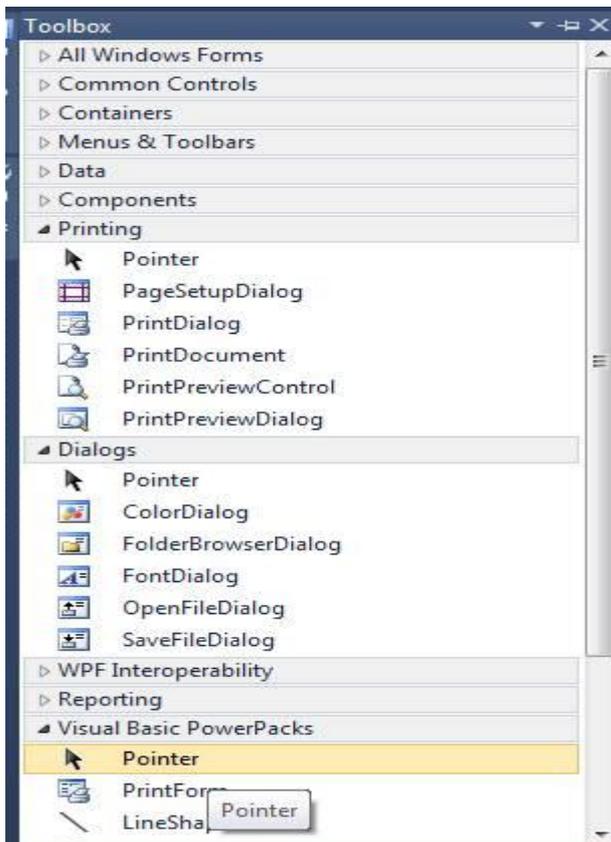
Private Sub ExitButton_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles ExitButton.Click
    'Exit the application by closing it
    Me.Close()
End Sub

```

Printing a Form

Visual Basic includes a **PrintForm** component – this is not a visible control on the form when added to a form – instead it displays in the component tray of the project.

- Return to the form's **Design View** (the Toolbox does not display in code view).
- As shown in this figure, locate the **PrintForm** component in the **Visual Basic PowerPacks** section of the Toolbox.
- Double-click the **PrintForm** component – you will see it display in the component tray of the IDE below the form you are designing. It will be named **PrintForm1**.



The printing **PrintAction** options when printing a form are: (1) Print to Printer, (2) Print to File, and (3) Print to Preview – this third option saves paper so we will use it.

Double-click the Print Form button control and then enter remarks and two coding statements as shown here.

```

Private Sub PrintButton_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles PrintButton.Click
    'Print the form to a print preview window

```

```

    'Assign a PrintAction option
    PrintForm1.PrintAction = Printing.PrintAction.PrintToPreview

    'Print the form
    PrintForm1.Print()
End Sub

```

Finish Coding the Application

To finish the application you need to write code for two buttons named Display2Button and Display3Button. The code for these buttons is shown here. Notice that these buttons are coded like Display1Button, however, different student and major information is displayed.

```

Private Sub Display2Button_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles Display2Button.Click
    'Display information for the second student
    NameTextBox.Text = "Jalal Hassan"
    MajorTextBox.Text = "Computer Science "
End Sub

Private Sub Display3Button_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles Display3Button.Click
    'Display information for the third student
    NameTextBox.Text = "Hussain Mohammed"
    MajorTextBox.Text = "Information Systems"
End Sub

```

Save the program and execute it to test it.

If you have not done so already, add **Remark** statements before the **Public Class** statement. These remarks identify the project, you as the programmer, and the date the program was written and must ALWAYS be included in the project.

- Place the **mouse cursor** in front of the Public Class statement.
- Click the **Enter** key several times to add blank lines above the Public Class statement.
- Type additional remarks similar to those shown here.

```

'Ch01VBUniversity
'Ali M.
'Today's Date
'Illustrates displaying information to read-only TextBox
'controls by clicking Button controls

```

Managing Projects

Copying Your Project to a Flash Drive

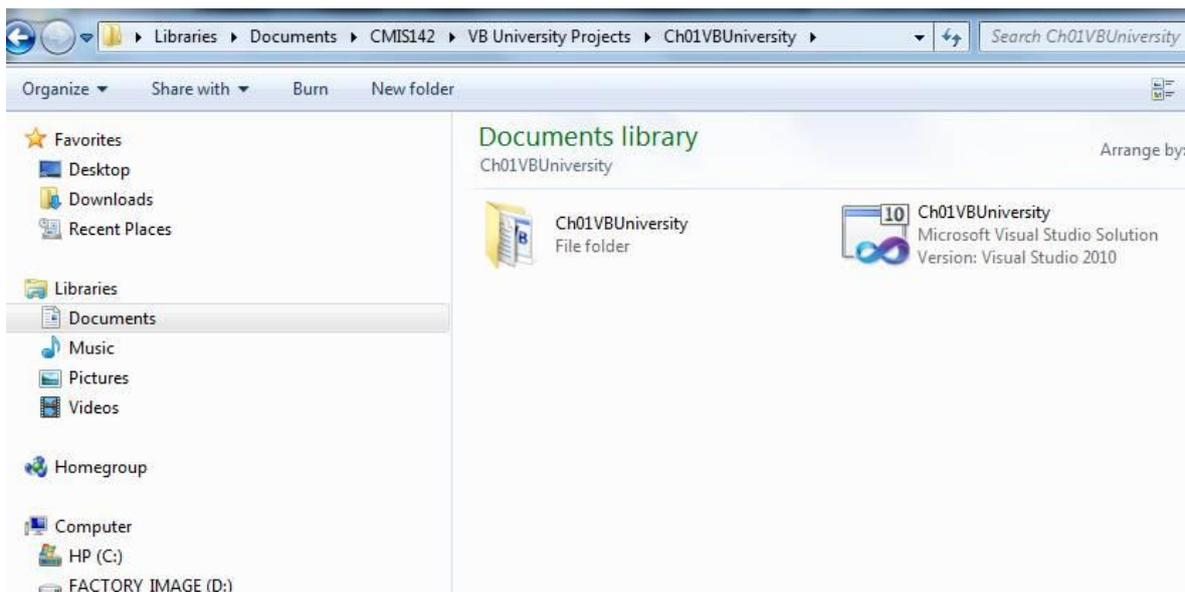
You need to be able to save your project and copy your work to a backup device, such as a flash drive.

- VB will save your project files every time you build or execute a project after your initial save.

- VB projects consist of many different files and folders within folders.
- Save files as you work by clicking the **Save All** button on the button toolbar. **DO NOT USE** the **File-Save As** menu at any time to try to save the project – if you do, you will likely only save an individual file, not the entire project, and you will not have a complete project saved.

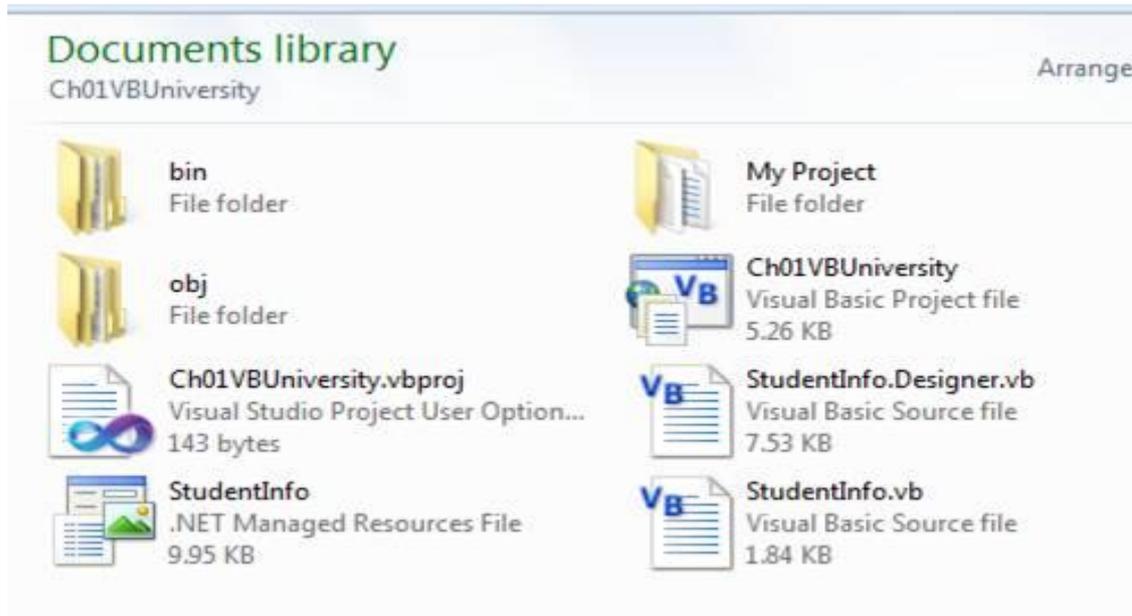
To save a VB project and copy it to a flash drive:

- Click the **Save All** button.
- Exit VB by clicking the **File** menu, **Exit** menu option. If you have not saved all of your program modifications, VB will ask you if you wish to save your changes. It is important to exit VB prior to trying to backup your project.
- Once you have exited VB, use the Microsoft Windows **My Computer** to navigate to drive where your project is stored on either **My Documents** or **C:\Temp** folder. You should be able to locate a folder named **Ch01VBUniversity** – this is the solution folder for the project you developed while studying these notes.
- To make a backup to a flash drive. Copy the entire folder from its current location to your flash drive,
- This figure shows the project folder opened up.



- You may wish to examine the project folder. This figure above shows the folder open. Notice that the original project folder named **Ch01VBUniversity** has another folder inside of it with an identical name. It also has the **Ch01VBUniversity.sln** solution file.
- If you click the "inner" **Ch01VBUniversity** folder, you will find folders named **bin**, **My Project**, and **obj**. You will also find at five files with names like **Ch01VBUniversity.vbproj**, **Ch01VBUniversity.vbproject.user**, **StudentInfo.designer.vb**, **StudentInfo.resx**, and **StudentInfo.vb** as shown in the figure below.

- You should confirm that the backup copy of the project folder on your flash drive contains folders and files that are identical to the folders and files in your original project folder.



Opening an Existing Project

Now you've returned from lunch and are ready to resume work on your project.

- Use the Windows Start button, then find Microsoft Visual Studio.Net!
- Locate the **Ch01VBUniversity** project listed under existing projects.
- Click on **Ch01VBUniversity.sln** to launch the project – this will enable you to revise or continue work on an existing project.

Program Errors

From time to time you will make errors when entering program code. VB expects you to follow specific rules for syntax, punctuation, and spelling.

VB has a smart editor that attempts to find and fix most errors.

- If VB cannot understand some of your code, it will display a blue squiggly line under the part of the line; that cannot be interpreted.
- In the figure below the **blue squiggly line** indicates that VB does not understand what the **textName** or **textMajor** objects are.
- Most likely, you named these two objects **NameTextBox** and **MajorTextBox** when designing the form, but then used the wrong name when typing code in the sub procedure.

```

Private Sub btnDisplay_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnDisplay.Click
    'Display name and major to the two textbox controls
    Me.textName.Text = "Douglas Bock"
    Me.textMajor.Text = "Computer Management and Information Systems"
End Sub

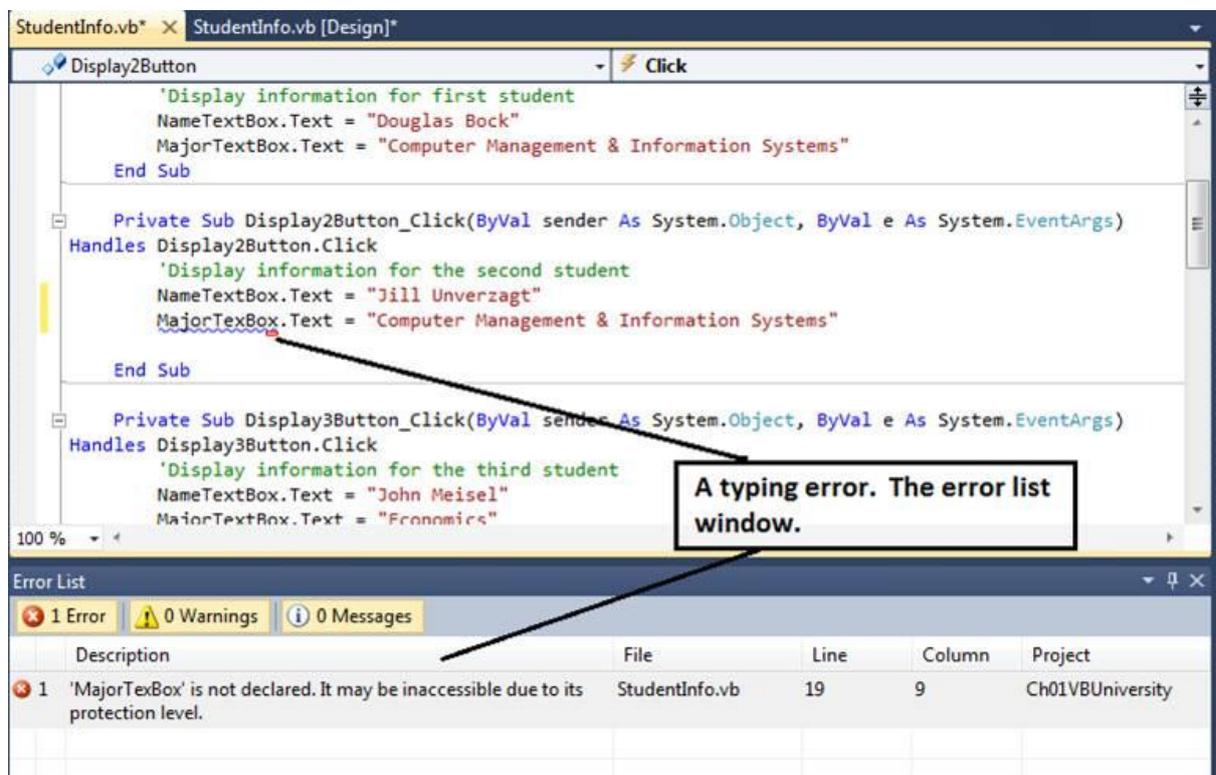
```

The blue squiggly line indicates some sort of error -- here the name of each of the text box controls is misspelled -- does not match the name specified during form design.

Typical **syntax errors** include:

- Referring to an object such as a label or button control by the wrong name – either you forgot to name the control or you made a mistake when typing the name.
- Forgetting to use the syntax: **object.property**.
 - o Forgetting the **dot** between the object and property names will cause an error.
 - o Typing an incorrect or invalid property name will also cause an error.

VB lists errors in the **View menu** -> **Error List** window that opens when you run your VB project. The figure shown below displays an error message corresponding to the error that occurs when you don't type the name of a control correctly.



There are two other types of errors that we will study in later chapters:

- **Run-Time error** – an error or exception that occurs while running a program.
- **Logic error** – a program error that causes the display of erroneous output.

A Clean Compile

After you correct errors in your code, access the **Build** menu and select the **Rebuild** option to recompile the program into a new MSIL version.

Compiling means to convert the program from human-readable form to machine-readable form – all referenced classes and components are combined by VB into an executable (.exe) file.

Design Time, Run Time, Break Time

There are some additional terms for you to learn about designing and executing a VB program.

- **Design Time** – when you are designing a project using the IDE, you are in what is termed **Design Time**.
- **Run Time** – when you execute or run a project, you are in **Run Time**.
- **Break Time** – occurs whenever VB encounters some type of processing error that prevents program execution from continuing. We will cover error-handling throughout your studies of VB.

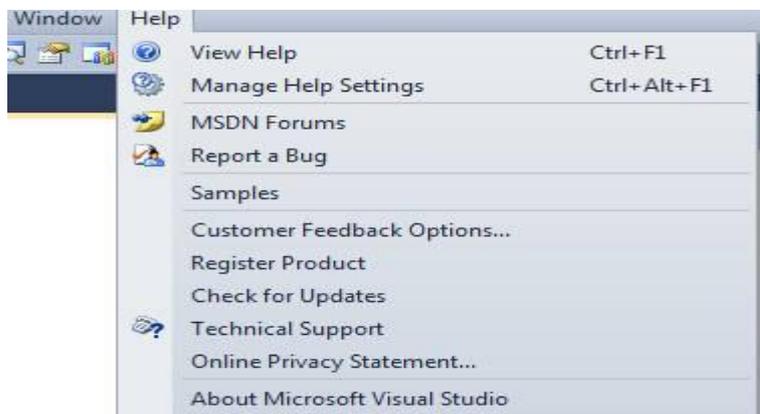
VB Help

MSDN

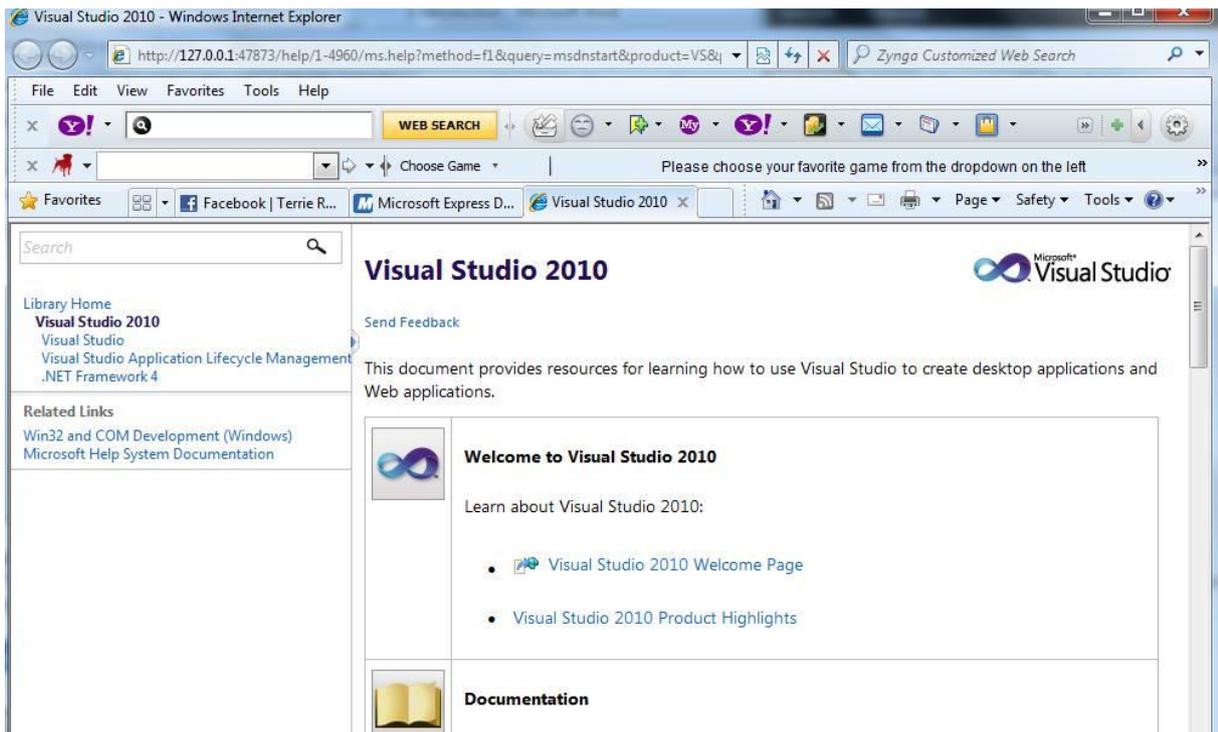
The Microsoft Developer Network (**MSDN**) library contains books and technical articles to help you answer questions that will arise as you learn the VB language. This must be installed on your computer in order for it to be available from the **Help** menu within VB.

- MSDN can be access from a hard drive, network drive, or the Web (requires an Internet connection).
- Hard drive installation is fastest and most convenient.
- Web access is from <http://msdn.microsoft.com>.

This figure shows the **Help** menu.



You may opt to view Help on the Internet or use local Help installed on the PC.



You can also obtain **context-sensitive help** on an object by placing the insertion point in a word in the coding editor or by selecting a VB object and pressing the **F1** function key.

A good way to start learning about Help is to access the Help menu Index option, then inside that window, access the Help menu and look for **Help on Help** as a submenu – this explains how to look up topics in Help.

You're now ready to complete your first computer programming laboratory assignment. You may wish to review these notes prior to starting your practice computer lab assignment.

Solution to In-Class Exercise

```
'Ch01VBUniversity
'D. Bock
'Today's Date
'Illustrates displaying information to read-only TextBox
'controls by clicking Button controls
```

```
Public Class StudentInfo
```

```
    Private Sub Display1Button_Click(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles Display1Button.Click
        'Display information for the first student
        NameTextBox.Text = "Douglas Bock"
        MajorTextBox.Text = "Computer Management & Information
Systems"
    End Sub
```

```
Private Sub Display2Button_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Display2Button.Click
    'Display information for the second student
    NameTextBox.Text = "Jill Unverzagt"
    MajorTextBox.Text = "Computer Management & Information
Systems"
End Sub

Private Sub Display3Button_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Display3Button.Click
    'Display information for the third student
    NameTextBox.Text = "John Meisel"
    MajorTextBox.Text = "Economics"
End Sub

Private Sub ResetButton_Click(ByVal sender As System.Object, By
Val e As System.EventArgs) Handles ResetButton.Click
    'Reset the form by clearing the TextBox controls
    NameTextBox.Clear()
    MajorTextBox.Clear()
End Sub

Private Sub PrintButton_Click(ByVal sender As System.Object, By
Val e As System.EventArgs) Handles PrintButton.Click
    'Print the form to a print preview window
    'Assign a PrintAction option
    PrintForm1.PrintAction =
Printing.PrintAction.PrintToPreview

    'Print the form
    PrintForm1.Print()
End Sub

Private Sub ExitButton_Click(ByVal sender As System.Object, ByV
al e As System.EventArgs) Handles ExitButton.Click
    'Exit the application by closing it
    Me.Close()
End Sub

End Class
```