INTRODUCTION

Microsoft's **Visual Studio** (**Visual Studio.NET**) is a suite of products that includes the .NET Framework and the Integrated Development Environment (IDE). The .NET Framework supports several different programming languages, which includes: **Visual Basic**, **Visual C#** (C sharp), **C++** (cee-plus-plus), **Visual F#** (F sharp) and Web Development (**ASP.NET**). These languages **compile** i.e. 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.
VB is available in several editions including the free **Express Edition** that is downloaded from Microsoft. Other editions are **Professional**, **Premium**, and **Ultimate**.
VB supports programming projects that run in both:

- Console Applications
- Windows Applications
- Web-based, Client-Server Applications

Microsoft .NET Framework

Provides a common set of services that can be used when programming in any supported language and enables programmers to write programs that run on any operating system on any hardware platform
The *.NET Framework* encompasses the following:

- *A new way to expose operating system and other APIs.* For years, the set of Windows functionality that was available to developers and the way that functionality was invoked were dependent on the language environment being used. For example, the Windows operating system provides the ability to create windows. Yet, the way this feature was invoked from a C++ program was dramatically different from the way it was invoked from a Visual Basic program. With .NET, the way that operating system services are invoked is uniform across all languages. This portion of .NET is commonly referred to as the *.NET Framework class library*.
- *A new infrastructure for managing application execution.* To provide a number of sophisticated new operating-system services—including code-level security, cross-language class inheritance, cross-language type compatibility, and hardware and operating-system independence, among others—Microsoft developed a new runtime environment known as the Common Language Runtime (CLR). The CLR includes the Common Type System (CTS) for cross-language type compatibility and the Common Language Specification (CLS) for ensuring that third-party libraries can be used from all .NET-enabled languages. To support hardware and operating-system independence, Microsoft developed the Microsoft Intermediate Language (MSIL or IL). IL is a CPU-independent machine language-style instruction set into which .NET Framework programs are compiled. IL programs are compiled to the actual machine language on the target platform prior to execution (known as *just-in-time*, or JIT, compiling) i.e. IL is never interpreted.
- *A new web server paradigm.* To support high-capacity web sites, Microsoft replaced its Active Server Pages (ASP) technology with ASP.NET.
- *A new focus on distributed-application architecture.* Visual Studio .NET provides top-notch tools for creating and consuming *web services* -- vendor-independent software services that can be invoked over the Internet.

The .NET Framework is designed top to bottom with the Internet in mind. For example, ADO.NET, the next step in the evolution of Microsoft's vision of "universal data access," assumes that applications will work with disconnected data by default. In addition, the DOT.NET classes provide sophisticated XML capabilities, further increasing their usefulness in a distributed environment.

Visual Studio

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

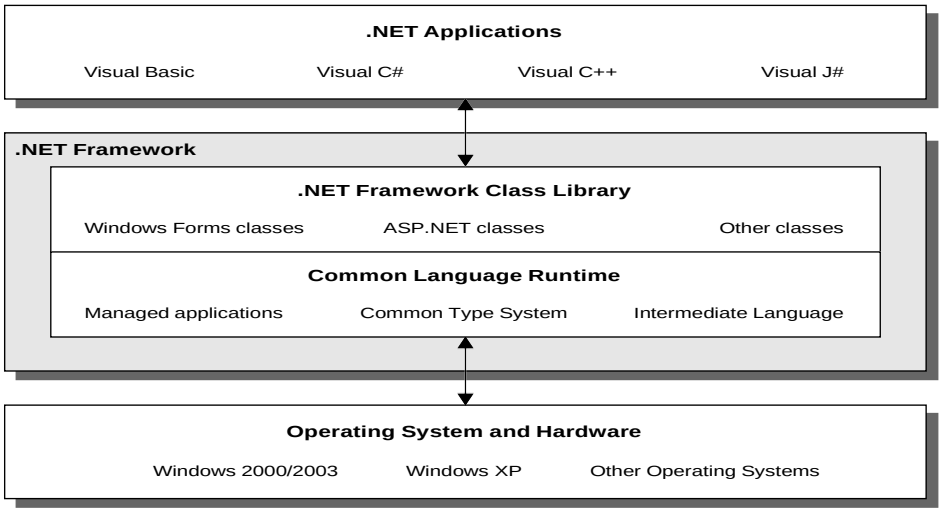
- The programming applications you will design and develop are called ***solutions*** i.e. each Application that you create is organized by Visual Studio into a Solution

- A solution is a container for all your work on an application. A solution contains several folders that define an application's structure. Each solution files have a file suffix of .sln
- A solution contains one or more projects
    - The project file is used to create an executable application
    - A project file has a suffix of .vbproj

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 (FCL)**. Contains files of pre-written code organized as classes. The classes themselves are organized (grouped) into Java-like *packages* called *namespaces.* The groupings are done according to the functionality of the classes. 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. This component is installed as part of the .NET Framework (mandatory).
- **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 i.e. you work with objects in building an application, e.g., Form objects, Button objects, TextBox objects, Label objects, ListBox 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 or Moving the mouse over the top of an object such as a text box.

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

Terminology	Definition
Object	An entity – 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**, and **Size**. Refer to a property by the notation **ObjectName.PropertyName** example: **TotalDueTextBox.Text**

Method	Like a verb in English – these are the actions that objects exhibit. Examples include methods to **Show**, **Hide** and **Close** forms. Refer to a method with the notation **ObjectName.MethodName** – example **Me.Close**
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 an 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**.

The Integrated Development Environment

The IDE is the interface between the programmer and the .NET tools. It is used to develop applications in any of the supported programming languages (Console, Web, and Windows development, etc). It includes an editor for all .NET languages as well as XML and HTML as well as a comprehensive set of tools for forms design and code organization

The following are some of the components of Visual Basic IDE:

1). **Form Designer** (also termed the **Document Window**).
Displays open documents such as the **Form1.** Tabs along the top are used to allow you to switch rapidly between open documents. The form can be resized by using the sizing handles.

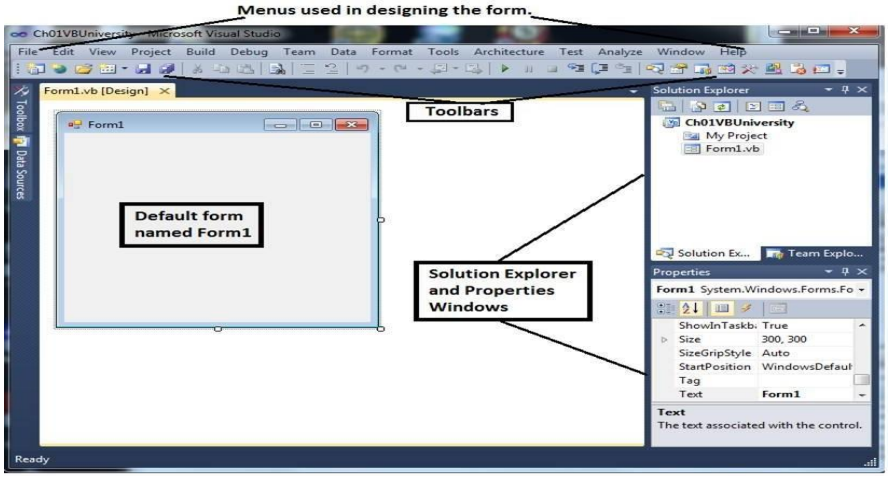
2). **Solution Explorer Window** – displays filenames for files that comprise a project. It is used to manage the elements of a solution.
- The View Code button displays the Code Editor for a module
- The View Designer button displays a visual designer

3). **Properties Window** – displays properties of the currently selected object

4). **ToolBox Window** – this is shown along the left edge of the IDE.
- Contains controls that are used to build forms.
- Can be expanded or collapsed with the **Pin/Unpin** icon.
- Used to add controls (tools) to a form by either double-clicking or dragging/dropping (your option on which to use).

5). The Error List window displays syntax errors

6). The Output window displays information as a project is compiled



Creating an Application

There are three steps for creating an application:

i) Designing the interface:
   - Form object is drawn as the container
   - Appropriate objects/controls are added to the form object from the toolbox

ii) Setting the Property values for forms and controls:
   - Design time:
     a) Select the object/control
     b) Using the properties window choose the object properties and settings
   - Run time:
     a) Access the code window
     b) Select the object/control and append code to the code event procedure

iii) Writing the program code:
   - Instructions to be executed in response to the objects/controls events is written in the appropriate event procedure within the code window

Creating a New Project

- In VS, you begin with the application you wish to create, and VS organizes everything you do into a Solution (with one or more Projects)
- The Application, and everything related to it is encapsulated in a Visual Studio solution
  - The solution takes on the name of the first Project created

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.
- Changing the form's **FileName** also changes the **Name** property of the form – select the **Form** and examine the **Name** property.
- Change the **Title Bar** value of a form by typing a new value into the **Text** property for the form.
  - 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.

Naming Rules and Conventions

Visual Basic automatically assigns a value to the **Name** property of each control, for example, **Label1**, **Label2**, or **TextBox1** or **Button1**,etc. However, it is difficult to remember the difference between Label1 and Label2

- 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. The **Pascal naming convention** will be used. 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).

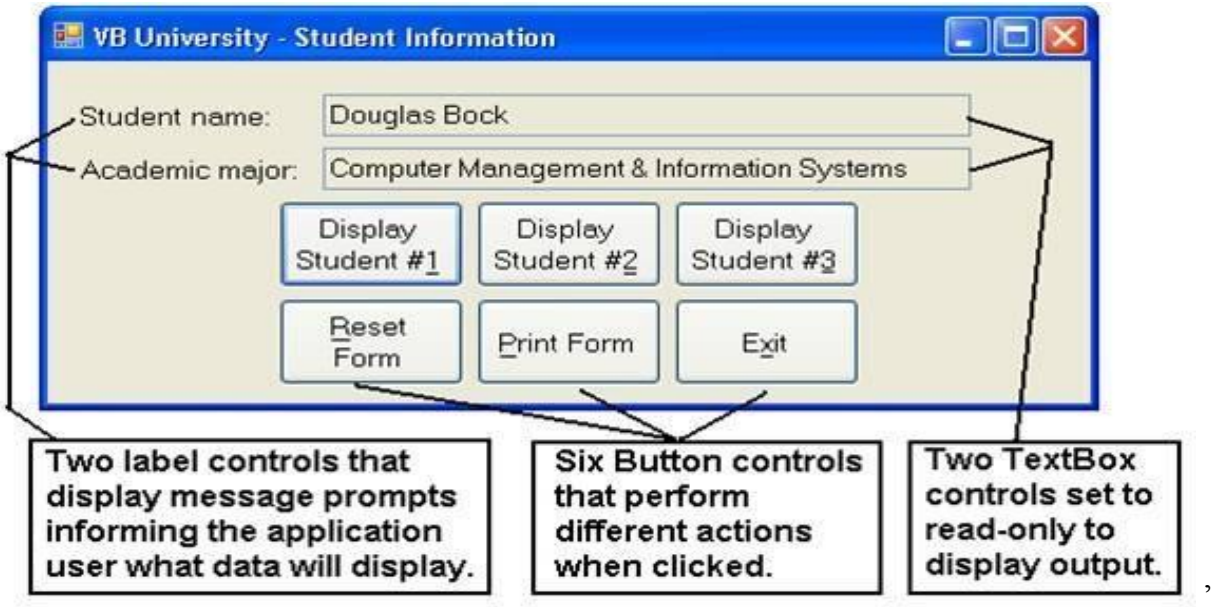
This table gives examples of both the Camel Casing and Hungarian naming conventions.

Camel Casing (Pascal) and Hungarian Naming Conventions

Control Type and Camel Casing Naming Suffix	**Example Camel Casing Control Names**	**Hungarian Naming Prefix**	**Example Hungarian Control Names**
TextBox	NameTextBox, MajorTextBox	txt	txtName, txtMajor
Button	ShippingButton, ExitButton, ResetButton	btn	btnShipping, btnExit, btnReset
Label	NameLabel, OutputLabel	lbl	lblName, lblOutput

Note: Label controls are often not renamed – they are not referred to later in writing computer code so the default assigned name is unchanged.

Sample Application
Design and code the following sample application



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.

- Drag the new label to where you want it in the upper left corner of the form.
- Change the label's **Text** property to **Student Name:**
- 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**

4. Add two text box controls as shown.
    - Name the first text box control **NameTextBox**.
    - Name the second text box control **MajorTextBox**.
    - These TextBox controls will only display output so set the properties **ReadOnly** = **True** and **TabStop** = **False**.
5. Add six button controls as shown.
    - Name the button controls **Display1Button**, **Display2Button**, **Display3Button**, **ResetButton**, **PrintButton**, and **ExitButton**.
    - 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

Saving and Running a Project

To save a project, use the **File-Save All** menu –**AVOID** using **File Save As** menu options .

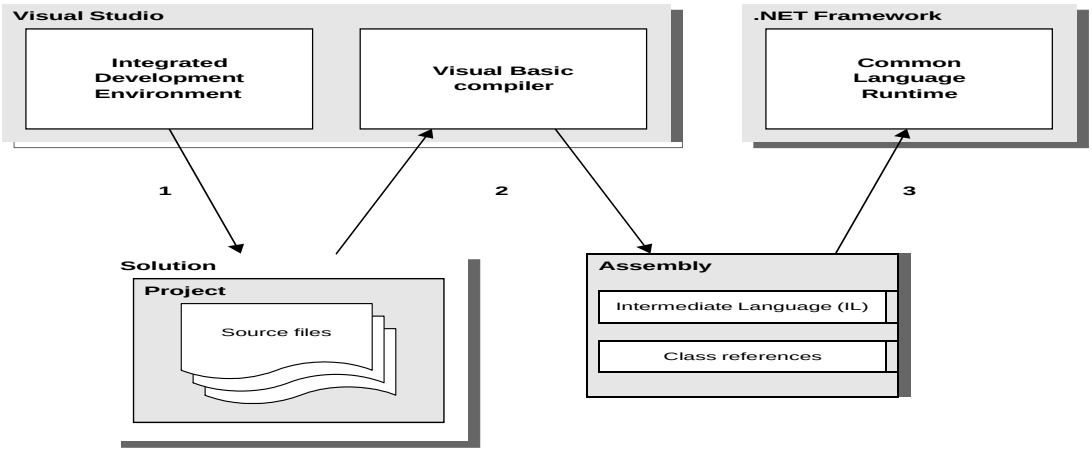
- Specify the solution name and the folder where the solution will be created. By default, a new folder is created for a new solution

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.

Compiling and Running Code

The following diagram illustrates how compilation and execution of solutions is carried in Visual Studio



Program Coding

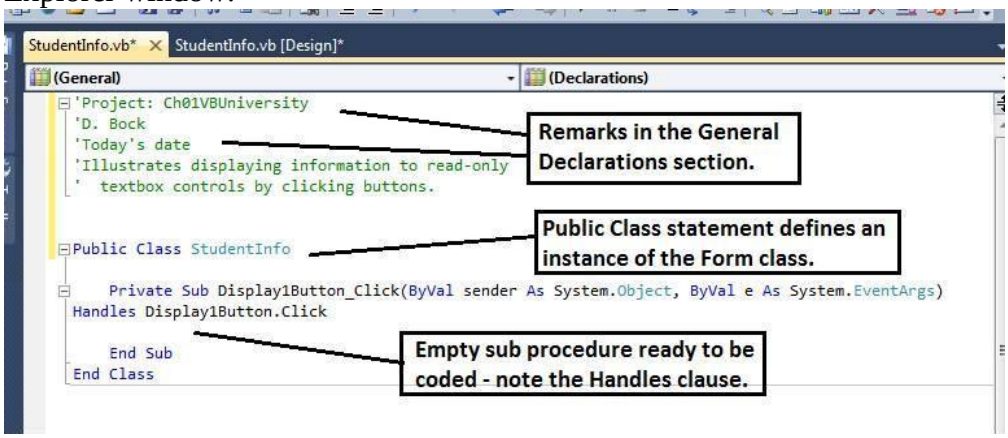
Programming or writing program code is the means used to cause something to happen when an event, such as a button click occurs.

Accessing an Event Procedure

Double-click the **Display1Button** control.

- This opens the coding window.

- The window is called the **View Code** window.
- You can also access the code view from the **View** menu or the **View** icon in the Solution Explorer window.



- 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**. Generally, Visual Studio "wires" an event to a handler using the "handles clause" in the handler header
- 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**.

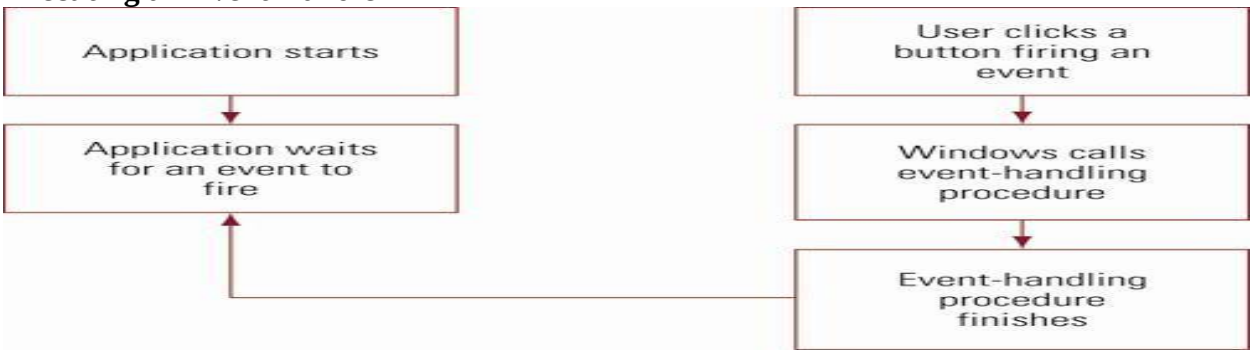
The form named **StudentInfo** is created as an instance of the general **Form** object defined in the .NET Framework class library.

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

Event Handlers

- An **event handler** is a procedure containing statements that execute when the end user performs an action. Activity happens when an event is fired.
  - Windows executes different event handlers as different events fire
  - Windows fires a **`Click`** event when the end user clicks a button
  - Different types of controls support different events
- Notice that when an event is fired we get another process running – the event handler

Executing an Event Handler



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

- **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.

Note: Remark statements before the **Public Class** statement are used to identify the project, the programmer, and the date the program was written and must ALWAYS be included in the project.

```
'VB University
'Author name
'Today's Date
'Illustrates displaying information to read-only TextBox
'controls by clicking Button controls
```

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** – 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.

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 below.

```
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 = "Douglas Bock"
      MajorTextBox.Text = "Computer Management & Information Systems"
End Sub
```

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()
```

Example

```
Private Sub ResetButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles ResetButton.Click
          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.

```
Private Sub ExitButton_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles ExitButton.Click
         Me.Close()
End Sub
```

Program Errors
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.
Typical **syntax errors** include:

- Referring to an object such as a label or button control by the wrong name.
- Forgetting to use the syntax: **object.property**.
    - Forgetting the **dot** between the object and property names will cause an error.
    - 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.
There are two other types of errors:

- **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.

Design Time, Run Time, Break Time
There are some additional terms for you to learn about designing and executing a VB program.

- **Design Time** – This is when you are designing a project using the IDE
- **Run Time** – This is when you execute or run a project.
- **Break Time** – occurs whenever VB encounters some type of processing error that prevents program execution from continuing.

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.

- 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**.

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.

Complete Solution

```
'Ch01VBUniversity
'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, ByVal e As
System.EventArgs) Handles ResetButton.Click
        'Reset the form by clearing the TextBox controls
        NameTextBox.Clear()
        MajorTextBox.Clear()
    End Sub

    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
End Class
```