

# Visual Basic Lecture 1

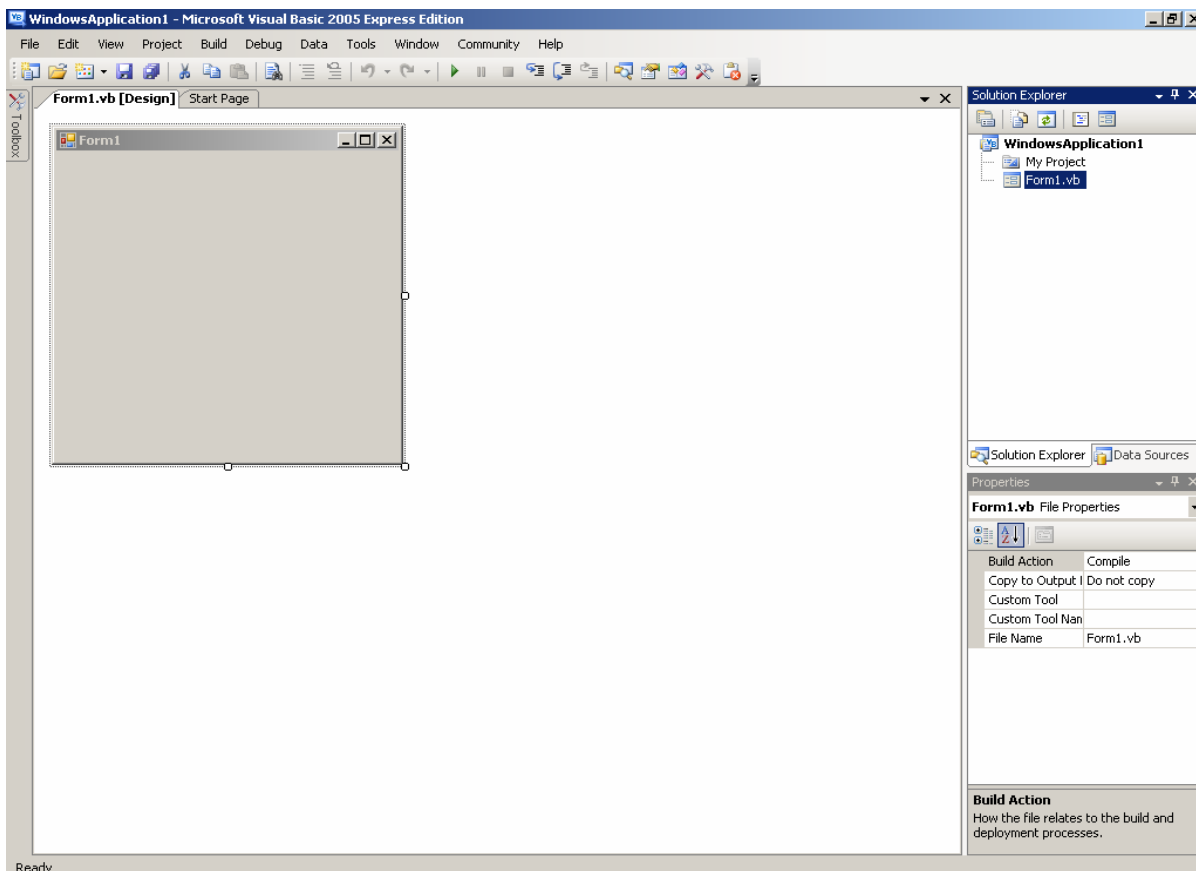
The duration of this class will focus on writing Windows-based applications using Microsoft Visual Basic. If you are interested in computer programming in other languages, software is available at: <http://msdn.microsoft.com/vstudio/express/downloads/>. Here you will be able to download software development tools from a variety of programming languages to suit your needs. And best of all: it's completely free! If you wanted to pay for all of these together, it could easily cost you \$1,200 for the full versions.

So what is MSVB? VB is a programming language that not only allows you to write code for programs, but also allows you to visually design what the final program will look like. It provides a simple way for you to see and develop your program simultaneously.

## Getting Started:

When you begin the VB 2005 Express program, you will have the option to click on several links that will allow you to see what's new in the world of Microsoft development. Also, in the lower left-hand box, you will notice a link that says "How do I ..." This can link you to both Microsoft's help page and utilize the built-in help that comes with the program.

In the upper left box, you will have options to open or create projects. Clicking on create will bring up a screen asking what type of application you want to create. Select **Windows Application**, type in the name of the project (if you have one), and press Ok. You should now see this:



You are currently looking at the design screen of your VB program. Let's go through what you are seeing piece-by-piece.

In the upper right-hand corner is the window for the **Solution Explorer**. Below the words Solution Explorer are five buttons. The first is for the **Properties** window which is below the Solution Explorer. The second is **Show All Files**, which allows you to see all of the files that VB is using to create your application. The third is a **Refresh** button. The fourth button allows you to **View Code** so you can tell your program what to do. The fifth button is **View Designer**, which will return you to this screen and allow you to add new features to your program.

In the Solution Explorer window, you can right-click on any of the objects to modify them. For instance, you can add additional Forms by right-clicking on the project name and clicking **Add ->Windows Form**. I'll let you explore those other options on your own.

Select **Form1** in the Solution Explorer or click on the Form in the design window and direct your attention to the **Properties Window**. Here you will see all of the options available to you for how you can modify this program. There are a lot of options available, but I want you to pay attention to a few. First, you can re-name your program. Whatever name is displayed here is not what the user will see when they run the program. In order to change the name on the form itself, go down to the **Text** property and change the name to whatever you desire. Notice that you also have the option to add background colors and images, and you can modify the icons, fonts, etc.

Direct your attention to the **Toolbox** located at the left side of the screen. Moving the mouse over the Toolbox will expand a menu with collapsible sections. I recommend that you expand the section for **Common Controls** and leave the others collapsed.

The Toolbox contains all of the items which you will be able to place onto your program. These are the objects which the user will be able to see and interact with. You are probably familiar with many of them already and may not realize it. Now you'll get to see how the things which you use so much actually work. Let's go through some of the common controls to make sure we understand what they are. If something does not look familiar to you, click on it, then move the mouse over the form, and click again. This will deposit one of these items onto your program so you can use it.

**Button:** causes an event to occur when the user clicks on the button

**CheckBox:** allows the user to select or de-select the option

**CheckedListBox:** allows the user to select or clear a variety of choices

**ComboBox:** contains a drop-down list for the user to select an item from

**Label:** displays text that is typically non-interactive

**LinkLabel:** a label with an internet hyperlink

**ListBox:** displays a list of items from which the user can make a selection

**MaskedTextBox:** a text box in which data of only one type can be entered

**ProgressBar:** displays a bar that fills to indicate progress on an operation

**RadioButton:** allows the user to select only a single choice out of several choices

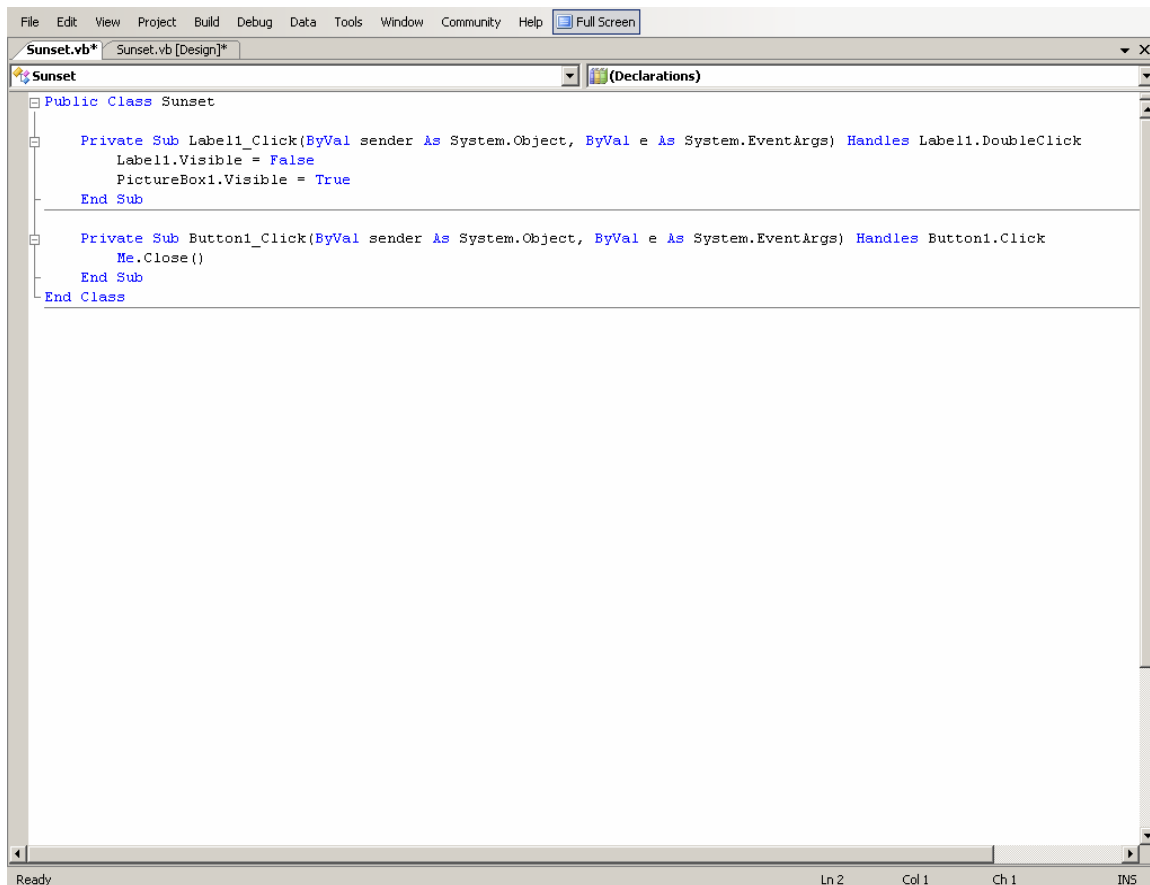
You'll notice that if you place one of these items on your form, the Properties Window changes. This window will now reflect all available properties of the object you have selected, not the Form itself. If you want to place a Label on the program, then select it from the Toolbox, click on the Form, and then look at the Properties Window. Notice that the default name of the Label is **Label1**. VB will always name object by type and number by default. You can change the name of this label if you want. Notice that changing the name does not change what is displayed. The name of the object will be important when modifying its properties in the code. If you want to change the text, go to **Text** on the Properties Window and change the text. Notice that if you delete all of the text from a label, it will be very difficult to find. It is better to type in some text initially, but set the property **Visible** to false if you don't want the user to see it when the program loads. These properties can also be modified in the code. For instance, if an event

occurs (the user clicks a button, checks a box, etc.), you might want some of these properties to change.

Let's try to write a basic program in VB. We want this program to display a line of text that tells the user to double click on the text in order to see a sunset. On the properties window, name the form **Sunset**. From the Toolbox, select **Label** and then click on the Form. A label should appear with the name **Label1**. On the properties of this label, go to **Text**, and change the text to "Double-click here to see a sunset". Once you move the mouse off of that line of the properties window, you'll notice that the text on the form has changed. Reposition the text so it is in the center of the Form. Now import a **Picture-box** from the toolbox and size it so that it takes up most of the Form. In the properties, go to **Image**, and click the button with the ellipses (...). You should now have a window prompting you to select a resource. Select **Local Resource, Import**, and select the file **Sunset** from C:\ Documents and Settings\ All Users \ Shared Documents \ Shared Pictures \ Sample Pictures. Once you have selected the file Sunset, click open, and the picture-box should now display the picture of the sunset. Also, add a **Button** and change the text on it to "Close".

Think about what this program is supposed to do. We want the user to double-click on the text in order to see the sunset. That means that the sunset must be invisible when the form loads. So go to the properties of the picture box, and go to the property **Visible**, and change it to **False**. The visibility of the Label should be set to True, since the user needs to see the text in order to click on it.

At this point, we've visually designed the program, but we haven't written any code to tell the computer what to do. The main event of this program will occur if the user double-clicks on the label, so this is where we must focus our attention. Double-click on the label, and VB will begin generating a sub-routine for you. Below is the completed code for this program.



```
File Edit View Project Build Debug Data Tools Window Community Help Full Screen
Sunset.vb* Sunset.vb [Design]*
Sunset (Declarations)
Public Class Sunset
    Private Sub Label1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Label1.DoubleClick
        Label1.Visible = False
        PictureBox1.Visible = True
    End Sub
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Me.Close()
    End Sub
End Class
Ready Ln 2 Col 1 Ch 1 INS
```

Notice that the code begins with **Public Class Sunset**. It is public in the way that the global variables were public. Other programs could conceivably call this program if I added this functionality to it. The term **class** means a logical grouping of procedures and data that simplify program organization. It is called Sunset because we changed the name of this form to Sunset. The drop-down menu on the left will list for us all of the objects contained in this class. The menu on the right will list all actions that can be performed on or by those objects.

The next line of code says:

```
Private Sub Label1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Label1.Click
```

This tells us that this is a **Private** (as opposed to **Public**) **sub-routine** (or **sub-procedure**) that will treat events (of type **Click**) on the object Label1. Specifically, this sub-routine will tell the computer what to do if the user clicks once on the label.

We said we wanted the user to double-click on the text. Change Label1.Click into Label1.DoubleClick. Notice that this can be done several ways. You can manually type it in; or you can also select the event **Double-click** from the drop-down menu.

We need this sub-procedure to hide the text in the label and make the picture appear. We do this by altering the properties of these objects. The properties of an object can be accessed by typing in its name followed by a period. When you hit the period, a drop-down menu will appear listing all of the properties of that object. Type in the following:

```
Label1.Visible = False  
PictureBox1.Visible = True
```

Now the sub-routine will tell the program that a double-click on the text will cause the text to become invisible and make the picture appear.

Also, the button “Close” that we created needs to close the program when it is clicked. Go back to the Designer view, double-click on the button “Close”, and the code for a sub-procedure will appear beneath the sub-procedure for the Label1.

If some code needs to access a property of the program, it can do so through the use of **Me**. Type in the following:

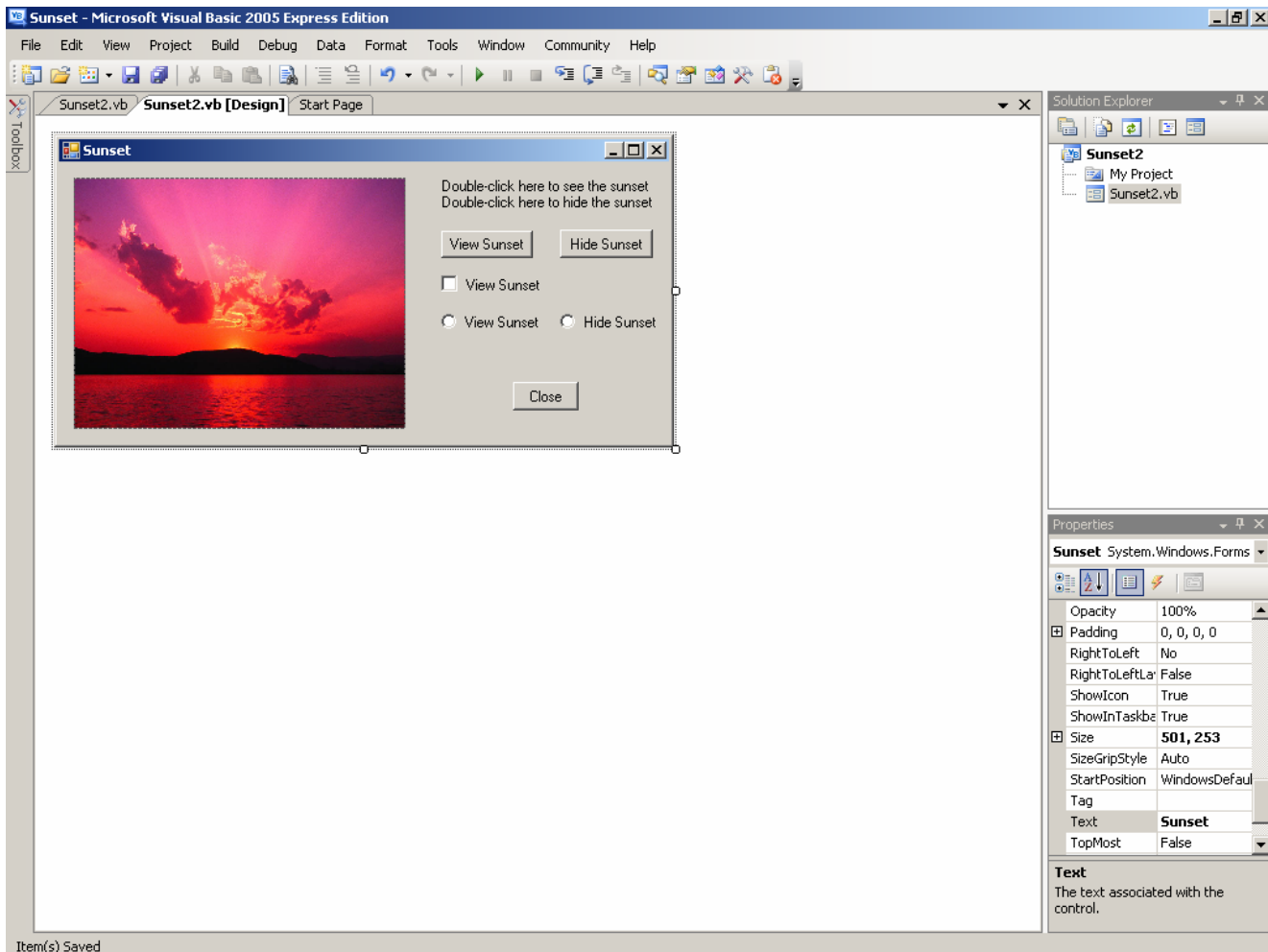
```
Me.close
```

Now that we’ve created the code for our program, let’s test it. Before compiling the program into an executable file that can be distributed, you always want to **Debug** the program first. In order to do this, click the **Debug** menu, and then click **Start Debugging**. VB will now create a temporary compilation of your program so you can test the code out. You can check that the program works by double-clicking on the text, verify that the text disappears and the picture appears, and then hit the close button.

Once you have checked the program and cleared out any bugs, you can create an executable (.exe) file that you can use on any Windows operating system. This can be done through the **Build** menu at the top of the screen. Before we do that, let’s add a little more functionality to this program.

Right now, the only way to make the picture appear is to double-click on the text. Normally, operations in Windows programs do not involve clicking on text. Also, we have not made a way to remove the picture once it has appeared. You normally use buttons, check-boxes, radio-buttons, etc. Let’s add that functionality to the program.

Go back to the design view and expand the size of the form to make room for additional objects. Since we already have a text line that accomplishes something, let's leave it in the program, but create another Label from the Toolbox, and add a Checkbox, two Radio Buttons, and two Buttons. Arrange it so your form looks something like this:



Change the text of the objects appropriately so it looks similar to what you see here. Now we need to add some functionality to this program. Let's think about what we want each of these objects to do.

The new Label, the Hide Sunset Button, and the Hide Sunset RadioButton need to make the picture disappear. So all of them will have the code:

```
PictureBox1.Visible = False
```

The Label you just added needs to make the picture disappear when the user double-clicks on it. But should this Label even be visible when the picture is not displayed? If we don't want the user having this option, we might want to make Label2 invisible at the start of the program. The same can be said of the Hide Sunset button. Go back to the designer view and change the Visible property to false for both of these. Also, the RadioButtons need to reflect the status of the program when it loads. When you begin the program, the picture is not visible; so why should the RadioButton labeled View Sunset be checked? Click on the RadioButton labeled Hide Sunset and change the Property **Checked** to true (or you could click on the RadioButton

labeled View sunset and change the Property **Checked** to false). Remember, any properties that you want loaded when the program starts should be set in the Designer View.

Now we need to make sure that all of these objects do everything that they are supposed to. Let's consider Label1, the label we originally worked with. When this label is double-clicked, it needs to disappear, make the picture appear, and also make Label2 appear. Also, it needs to make the appropriate changes to the other objects. Here's what the code might look like:

```
Private Sub Label1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Label1.DoubleClick
    Label1.Visible = False
    Label2.Visible = True
    Button2.Visible = False
    Button3.Visible = True
    RadioButton1.Checked = True
    CheckBox1.Checked = True
    PictureBox1.Visible = True
End Sub
```

**Homework 1:** Make the appropriate changes to the rest of the program. Make sure that each object interacts appropriately with the rest. You might want to make use of an **If...Then** statement for the checkbox and the RadioButtons.

**Extra Credit:** Good computer programmers always strive to make their programs as efficient as possible. One way to measure this is to look at how many lines of code you have. See if you can write this program in fewer than 50 lines of code and still have it retain all of its functionality. You can check how many lines of code you have by clicking on the last line of code and looking at the line number at the bottom of the screen.