C h a p t e r 2

Hello World Program

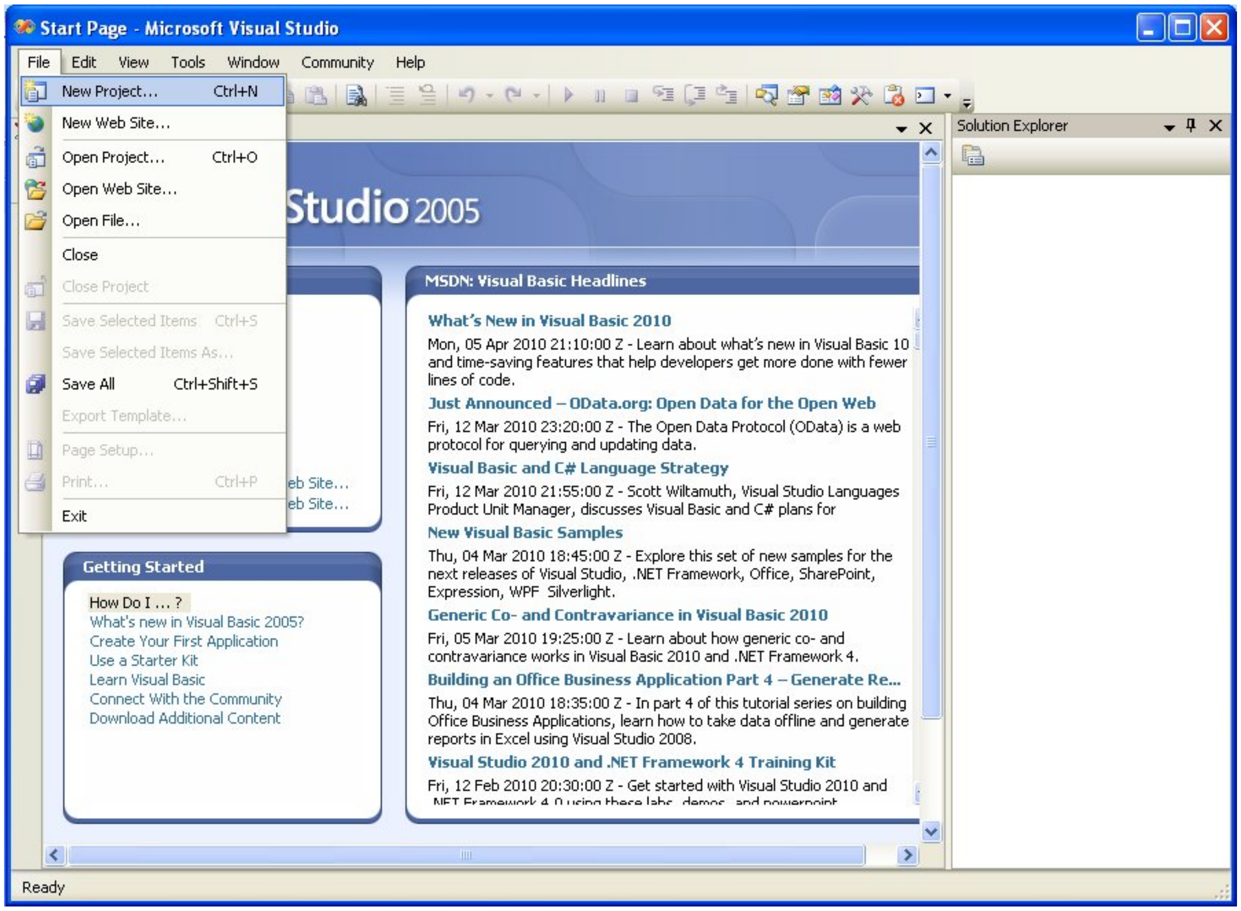
In this chapter, you will learn how to use the following Visual Basic Application functions to World Class standards:

- **Opening Visual Basic Editor**
- **Beginning a New Visual Basic Project**
- **Laying Out a User Input Form in Visual Basic**
- **Insert a Label into a Form**
- **Insert a Textbox into a Form**
- **Insert a Label into a Form to Post an Output**
- **Insert Command Buttons into a Form**
- **Adding a Copyright Statement to a Form**
- **Adding Comments in Visual Basic to Communicate the Copyright**
- **Declaring Variables in a Program with the Dimension Statement**
- **Setting Variables in a Program**
- **Using a Label to Communicate with Variables**
- **Ending the Program**
- **Running the Program**

Open the Visual Basic Editor

To open the Visual Basic Editor in Microsoft Visual Studio is essential to creating the program to automate any process. In this version of the World Class CAD – Visual Basic, we are using Visual Basic 2005.

To open a new project, we select File on the Menu Bar and New Project.



MSDN: Visual Basic Headlines

What's New in Visual Basic 2010

Mon, 05 Apr 2010 21:10:00 Z - Learn about what's new in Visual Basic 10 and time-saving features that help developers get more done with fewer lines of code.

Just Announced – OData.org: Open Data for the Open Web

Fri, 12 Mar 2010 23:20:00 Z - The Open Data Protocol (OData) is a web protocol for querying and updating data.

Visual Basic and C# Language Strategy

Fri, 12 Mar 2010 21:55:00 Z - Scott Wiltamuth, Visual Studio Languages Product Unit Manager, discusses Visual Basic and C# plans for

New Visual Basic Samples

Thu, 04 Mar 2010 18:45:00 Z - Explore this set of new samples for the next releases of Visual Studio, .NET Framework, Office, SharePoint, Expression, WPF Silverlight.

Generic Co- and Contravariance in Visual Basic 2010

Fri, 05 Mar 2010 19:25:00 Z - Learn about how generic co- and contravariance works in Visual Basic 2010 and .NET Framework 4.

Building an Office Business Application Part 4 – Generate Re…

Thu, 04 Mar 2010 18:35:00 Z - In part 4 of this tutorial series on building Office Business Applications, learn how to take data offline and generate reports in Excel using Visual Studio 2008.

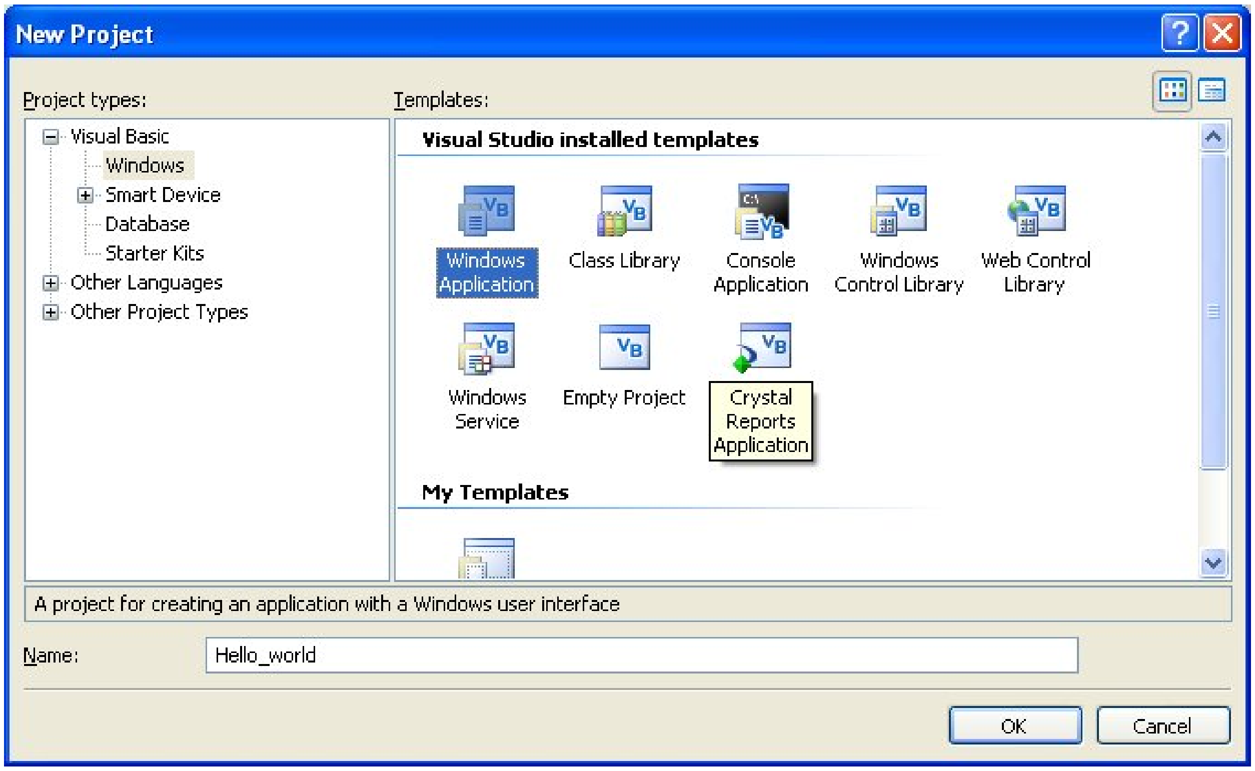
Visual Studio 2010 and .NET Framework 4 Training Kit

Fri, 12 Feb 2010 20:30:00 Z - Get started with Visual Studio 2010 and .NET Framework 4.0 using these labs, demos, and powerpoint

Figure 2.1 – The Start Page

We start a new Windows Application by picking the Windows Application icon on the New Project window.

The New Project window will appear and we will choose Window Applications from the list of installed templates.

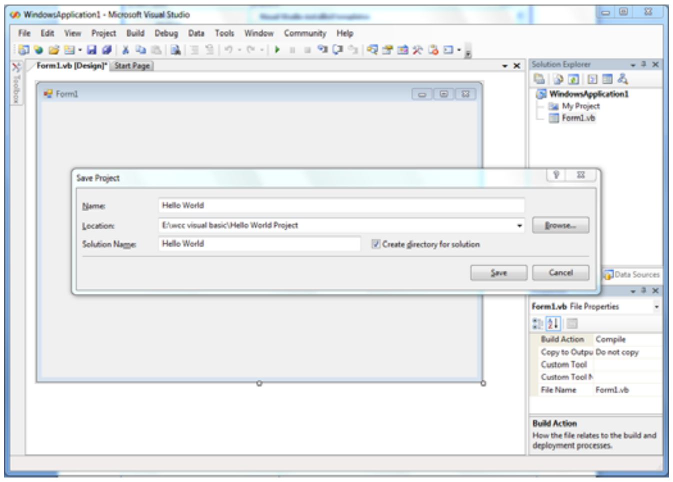


A project for creating an application with a Windows user interface

Name: Hello_world

Figure 2.2 – New Project

With the Project open, select **File** on the Menu Bar and select **Save All**. We will create a folder on either the desktop or the My Documents folder called "Visual Basic Projects". After creating the folder, name the project "Hello World". A folder called "Hello World" will be made and all the files for the program will be located in the folder.


WindowsApplication1 - Microsoft Visual Studio
File Edit View Project Build Debug Data Tools Window Community Help
Form1.vb [Design]* Start Page
Form1
Solution Explorer
WindowsApplication1
My Project
Form1.vb
Save Project
Browse...
Save
Cancel
Data Sources
Form1.vb File Properties
Build Action Compile
Copy to Outpu Do not copy
Custom Tool
Custom Tool N
File Name Form1.vb
Build Action
How the file relates to the build and deployment processes.

Name: Hello World
Location: E:\wcc visual basic\Hello World Project
Solution Name: Hello World
☑ Create directory for solution

Figure 2.3 – Saving the Hello World Program

Beginning a New Visual Basic Application

Remember, that all programming projects begin with one or more sketches. The sketch will show labels, textboxes, and command buttons. In our first project, we will name the input form, **Hello World**. We will place a textbox close to the top of the form to type a name. Over the top of the textbox, we will insert the label, “**Type your name**”. We will have three command buttons, **Hello**, **Reset** and **Exit**. On the bottom of the form, we will write the copyright statement using another label. On this presentation, we can help ourselves by being as accurate as possible, by displaying sizes, fonts, colors and any other specific details which will enable us to quickly create the form. From the beginning of inserting the form into the project, we need to refer to our sketch.

We should train new programmers initially in the art of form building. When using the editor, we insert and size the form, and selecting the Controls Toolbox, we will place all the various input tools and properly label them. Whenever we place an input tool, the properties window will display a list of every attribute associated with the tool, and we will take every effort to arrange the tool by performing such actions as naming, labeling and sizing the visual input device.

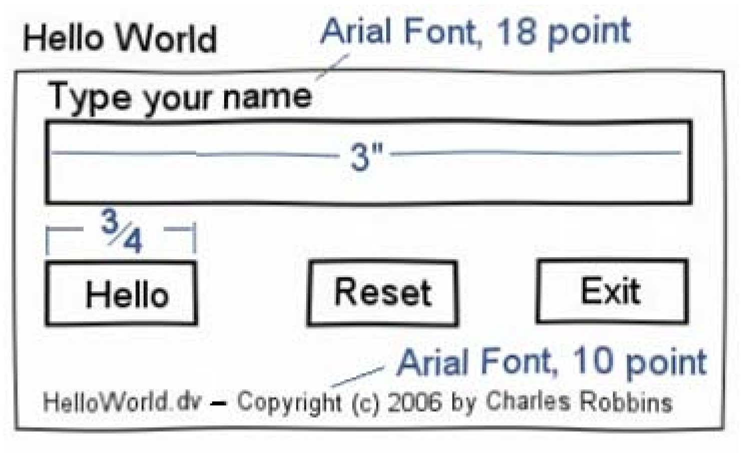
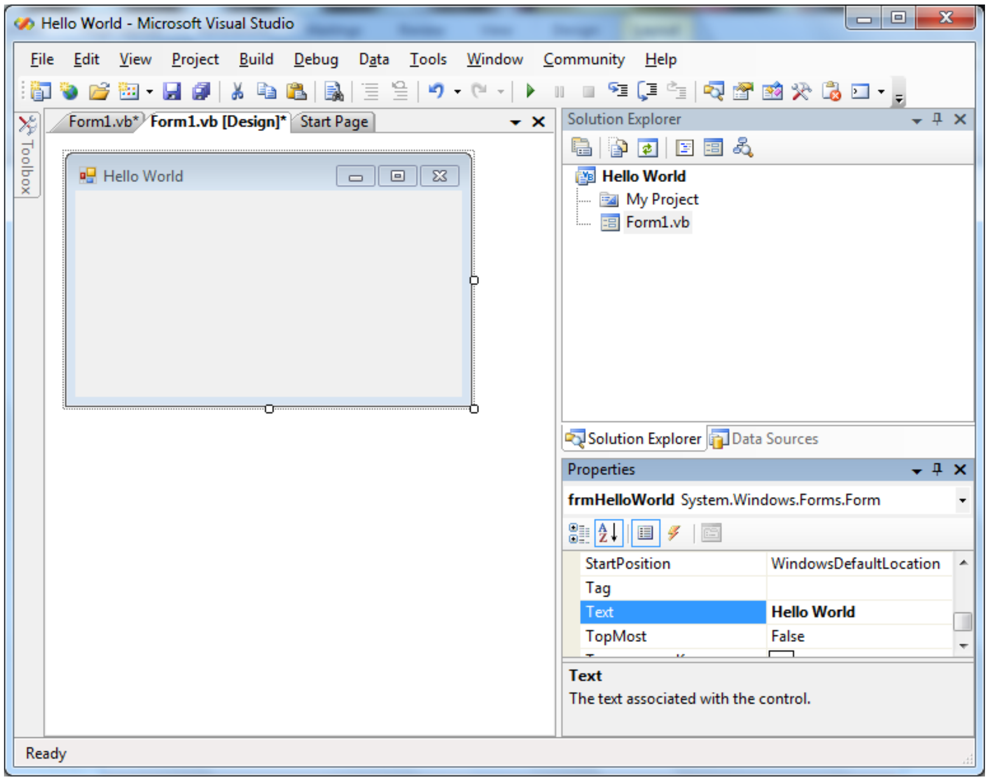


Figure 2.4 – Sketch of the Hello World Form


Hello World - Microsoft Visual Studio
File Edit View Project Build Debug Data Tools Window Community Help
Toolbox
Form1.vb* Form1.vb [Design]* Start Page
Hello World
Solution Explorer
Hello World
My Project
Form1.vb
Solution Explorer Data Sources
Properties
Ready

frmHelloWorld System.Windows.Forms.Form

StartPosition	WindowsDefaultLocation
Tag	
Text	Hello World
TopMost	False

Text
The text associated with the control.

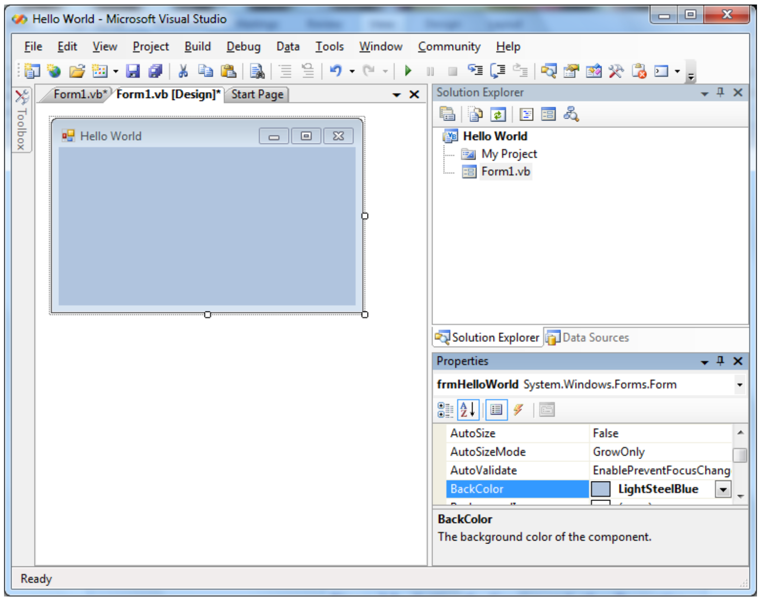
Figure 2.5 – Designing the Hello World Form in Visual Basic

Laying Out a User Input Form in Visual Basic

We will change the **Text** in the Properties pane to Hello World to agree with the sketch in Figure 2.4. Go ahead and change the form in two other aspects, BackColor and Size.

Alphabetic	
BackColor	Light Steel Blue
Size	320, 220

The first number is the width and the second number is the height. The form will change in shape to the size measurement.


Hello World - Microsoft Visual Studio
File Edit View Project Build Debug Data Tools Window Community Help
Form1.vb* Form1.vb [Design]* Start Page
Toolbox
Hello World
Solution Explorer
Hello World
My Project
Form1.vb
Solution Explorer Data Sources
Properties
Ready

frmHelloWorld System.Windows.Forms.Form

AutoSize	False
AutoSizeMode	GrowOnly
AutoValidate	EnablePreventFocusChang
BackColor	LightSteelBlue

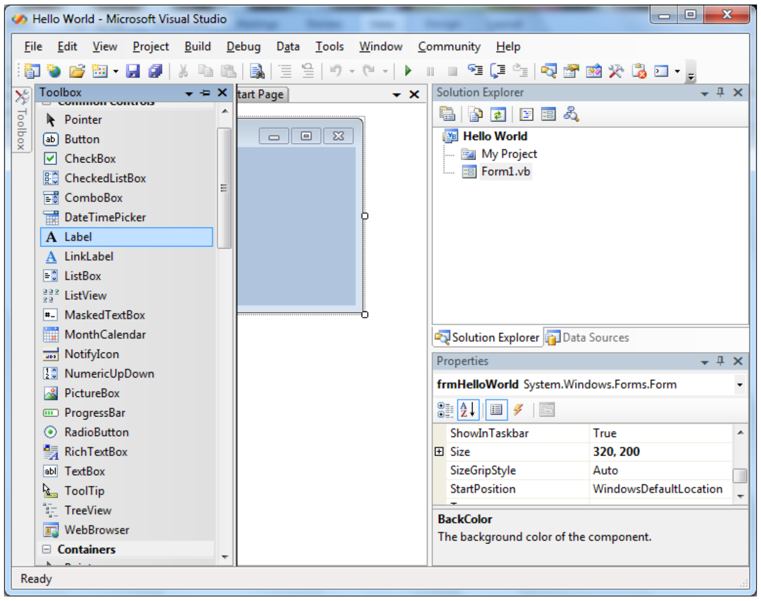
BackColor
The background color of the component.

Figure 2.6 – Setting BackColor and other Properties

The background color will change to a light blue. There are many more attributes in the Properties pane that we will use on future projects.

Inserting a Label into a Form

A good form is easy to figure out by the user, so when we are attempting to provide information on the window that will run in Windows; we add labels to textboxes to explain our intent. Press the Label (A) button on the Control Toolbar to add a label. To size the label area, click on the upper left area of the form and hold down on the left mouse button, draw the dotted label box.



frmHelloWorld System.Windows.Forms.Form

ShowInTaskbar	True
Size	320, 200
SizeGripStyle	Auto
StartPosition	WindowsDefaultLocation

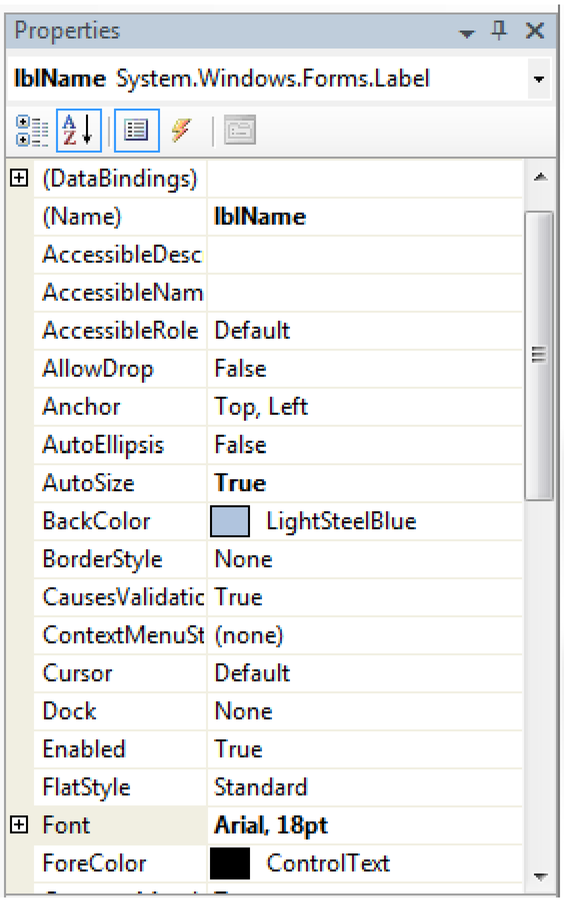
BackColor
The background color of the component.

Figure 2.7 – Placing a Label on the Form

We will name the Label using a common Visual Basic naming convention where the programming object is a three letter prefix followed by the name or phrase of the tool. For our first label, the name is **lblName.**

Alphabetic	
(Name)	lblName
BackColor	Light Steel Blue
Text	Type your name:
Font	Arial, 18 pt

On the sketch, the label's caption is "**Type your name:**" The font on the sketch is 18 point, Arial. When highlighting the row for Font, a small command button with three small dots appears to the right of the default font name of Microsoft San Serif. Click on the three dotted button to open the Visual Basic Font window.



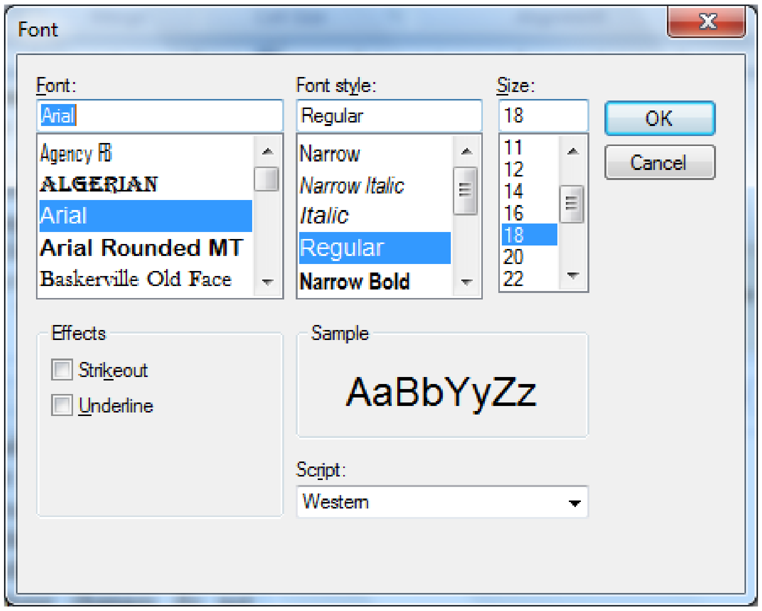
Properties

lblName System.Windows.Forms.Label

(DataBindings)	
(Name)	**lblName**
AccessibleDesc	
AccessibleNam	
AccessibleRole	Default
AllowDrop	False
Anchor	Top, Left
AutoEllipsis	False
AutoSize	**True**
BackColor	LightSteelBlue
BorderStyle	None
CausesValidatic	True
ContextMenuSt	(none)
Cursor	Default
Dock	None
Enabled	True
FlatStyle	Standard
Font	**Arial, 18pt**
ForeColor	ControlText

Figure 2.8 – Changing the Font Property

We will select the Arial font, Regular font style and 18 size for this project to agree with the initial sketch if the user input form. When we adjust the attributes for the label, these changes do not alter globally for the other objects on the form. If we wish to underline the text or phrase in the label, add a check to the Underline checkbox in the Effects section of the Font window. When we finish making changes to the font property, select the OK command button to return to the work area.



Font

Font: Arial
Font style: Regular
Size: 18

Script: Western

Figure 2.9 – The Font Window in Visual Basic

When the first label is done, the background color of the label matches the background color of the form. In many cases that effect is visually pleasing to the eye, versus introducing another color. Both color and shape will direct the user in completing the form along with the explanation we place on the window to guide the designer in using the automated programs. Use colors and shape strategically to communicate well.

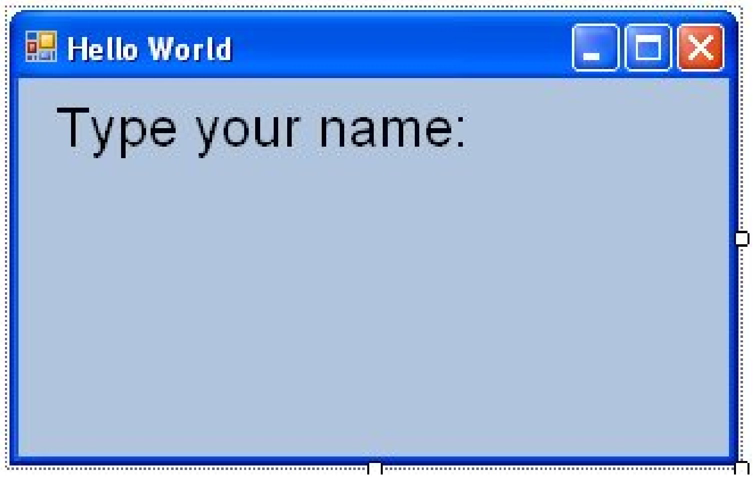


Figure 2.10 – The Finished Label on the Form

Inserting a Textbox into a Form

A textbox is used so that a user of the computer program can input data in the form of words, numbers or a mixture of both. Press the TextBox (ab) button on the Control Toolbar to add a textbox. To size the label area, click on the upper left area of the form and hold down on the left mouse button, draw the dotted textbox.

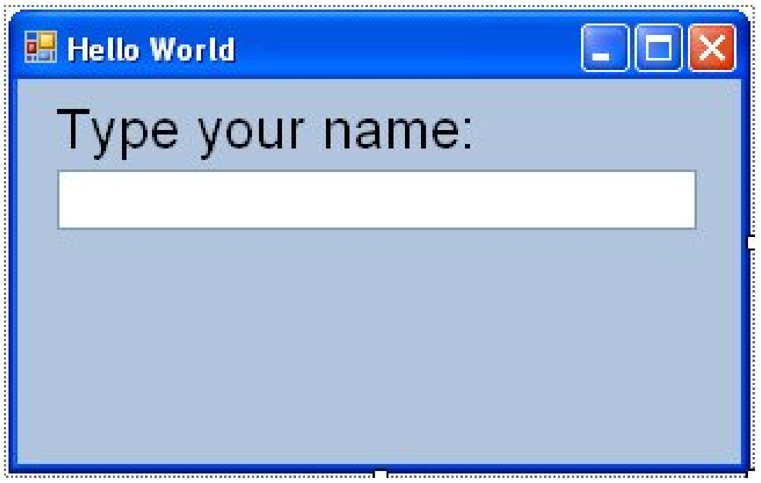
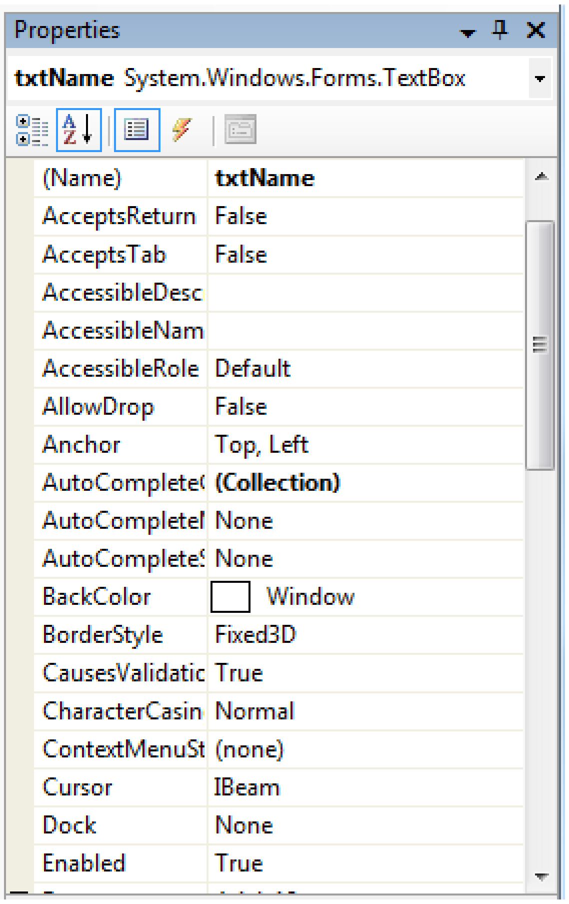


Figure 2.11 – Placing a TextBox on the Form

We will name the TextBox using the three letter prefix followed by the name or phrase of the tool. For our first textbox, the name is **txtName.**

Alphabetic	
(Name)	txtName
Font	Arial, 18 pt
Size	276, 35

The font on the sketch is 18 point, Arial. When highlighting the row for Font, a small command button with three small dots appears to the right of the default font name of Microsoft San Serif. Click on the three dotted button to open the Visual Basic Font window. Make the changes like we did on the Label and press OK to save the property.


Properties

txtName System.Windows.Forms.TextBox

(Name)	**txtName**
AcceptsReturn	False
AcceptsTab	False
AccessibleDesc	
AccessibleNam	
AccessibleRole	Default
AllowDrop	False
Anchor	Top, Left
AutoComplete(	**(Collection)**
AutoCompleteI	None
AutoComplete!	None
BackColor	⬜ Window
BorderStyle	Fixed3D
CausesValidatic	True
CharacterCasin	Normal
ContextMenuSt	(none)
Cursor	IBeam
Dock	None
Enabled	True

Figure 2.12 – Changing the (Name) to txtName

Inserting a Label into a Form to Post the Output

Some labels on a form are in a position to display an answer after the user inputs data and they press the command button to execute the application. To add this label, press the Label (A) button on the Control Toolbar to add a label. To size the label area, click on the upper left area of the form and hold down on the left mouse button, draw the dotted label box.

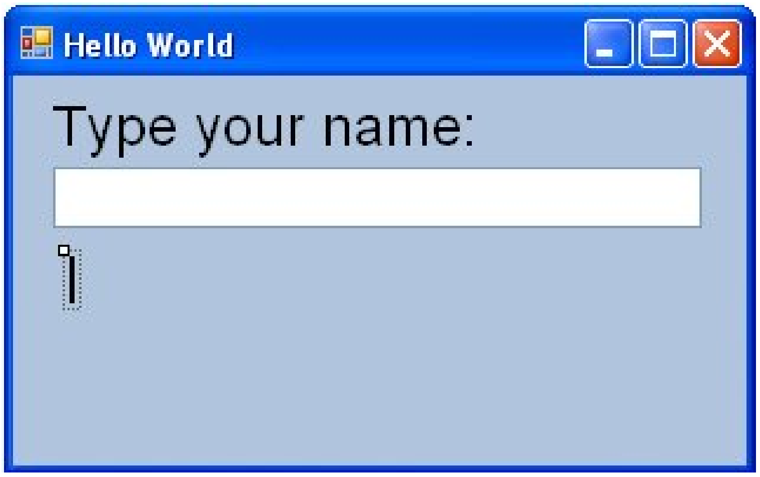
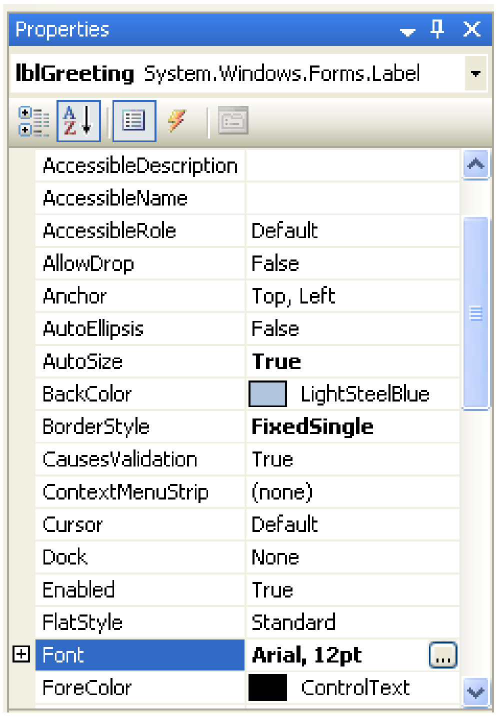


Figure 2.13 – Placing another Label on the Form

We will name the Label using the name is **lblGreeting.**

Alphabetic	
(Name)	lblGreeting
BorderStyle	FixedSingle
Font	Arial, 12 pt
Size	2,20

The font on the sketch is 12 point, Arial. When highlighting the row for Font, a small command button with three small dots appears to the right of the default font name of Microsoft San Serif. Click on the three dotted button to open the Visual Basic Font window. Make the changes as we did before and press OK to save the property.



Properties

lblGreeting System.Windows.Forms.Label

AccessibleDescription	
AccessibleName	
AccessibleRole	Default
AllowDrop	False
Anchor	Top, Left
AutoEllipsis	False
AutoSize	**True**
BackColor	LightSteelBlue
BorderStyle	**FixedSingle**
CausesValidation	True
ContextMenuStrip	(none)
Cursor	Default
Dock	None
Enabled	True
FlatStyle	Standard
Font	**Arial, 12pt**
ForeColor	ControlText

Figure 2.14 – Changing the (Name) to lblGreeting

Inserting a Command Buttons into a Form

A command button is used so that a user will execute the application. Press the Command button on the Control Toolbar to add a command button. To size the label area, click on the upper left area of the form and hold down on the left mouse button, draw the command button as shown in Figure 2.15.

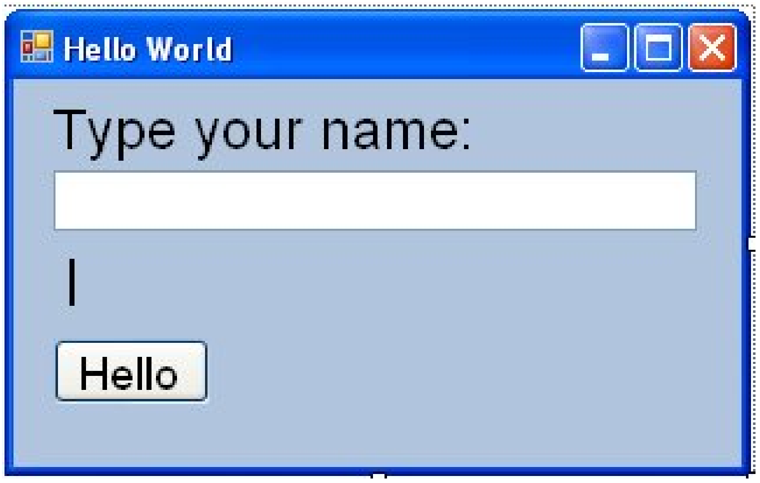
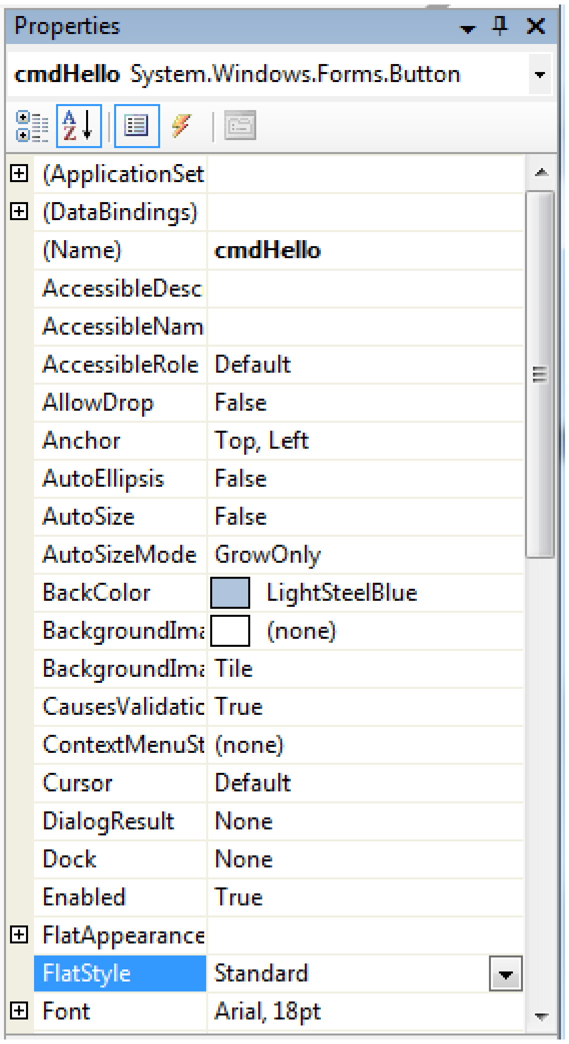


Figure 2.15 – Insert a Command Button onto a Form

We will name the command button using the name is **cmdHello.**

Alphabetic	
(Name)	cmdHello
Caption	Hello
Font	Arial, 18 pt
Size	80, 38

The font on the sketch is 18 point, Arial. When highlighting the row for Font, a small command button with three small dots appears to the right of the default font name of Microsoft San Serif. Click on the three dotted button to open the Visual Basic Font window. Make the changes as we did before and press OK to save the property.



Properties

cmdHello System.Windows.Forms.Button

(ApplicationSet	
(DataBindings)	
(Name)	cmdHello
AccessibleDesc	
AccessibleNam	
AccessibleRole	Default
AllowDrop	False
Anchor	Top, Left
AutoEllipsis	False
AutoSize	False
AutoSizeMode	GrowOnly
BackColor	LightSteelBlue
BackgroundIma	(none)
BackgroundIma	Tile
CausesValidatic	True
ContextMenuSt	(none)
Cursor	Default
DialogResult	None
Dock	None
Enabled	True
FlatAppearance	
FlatStyle	Standard
Font	Arial, 18pt

Figure 2.16 – Changing the (Name) to cmdHello

Add a second Command button, named cmdReset is for clearing the txtName and lblGreeting objects. The third command button is to exit the program. When the user presses the Exit command button, the application closes. Notice the equal spacing between the command buttons gives a visually friendly appearance.

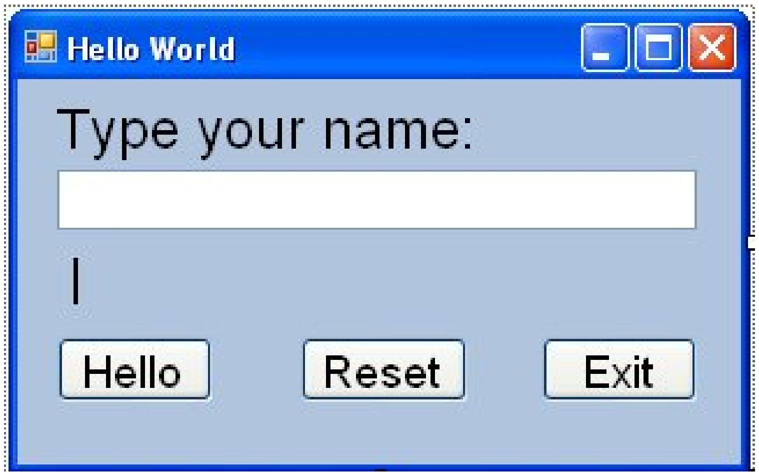


Figure 2.17 – Insert Two More Command Buttons

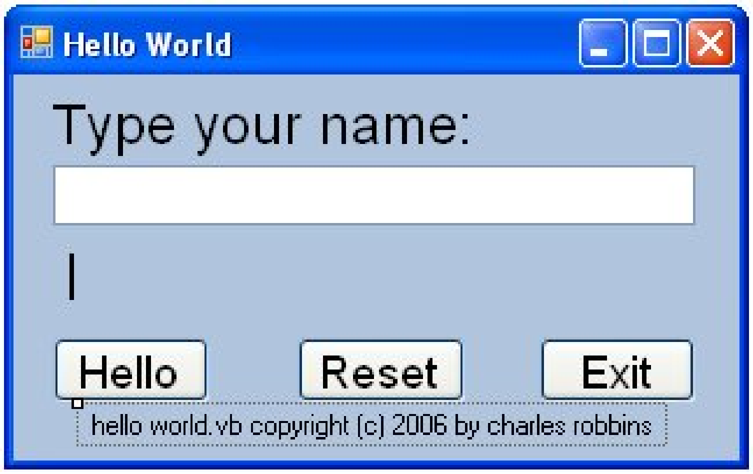
Adding a Copyright Statement to a Form

At the beginning of a new program, we will expect to see an explanation or any special instructions in the form of comments such as copyright, permissions or other legal notices to inform programmers what are the rules dealing with running the code. Comments at the opening of the code could help an individual determine whether the program is right for their application or is legal to use. The message box is a great tool when properly utilized to inform someone if they are breaking a copyright law when running the code.

Finish the form with the following copyright information.

'hello world.dvb copyright (c) 2006 by charles robbins

If there are special rules or instructions that the user needs to know, place that information on the bottom of the form.



hello world.vb copyright (c) 2006 by charles robbins

Figure 2.18 – Adding a Copyright Statement

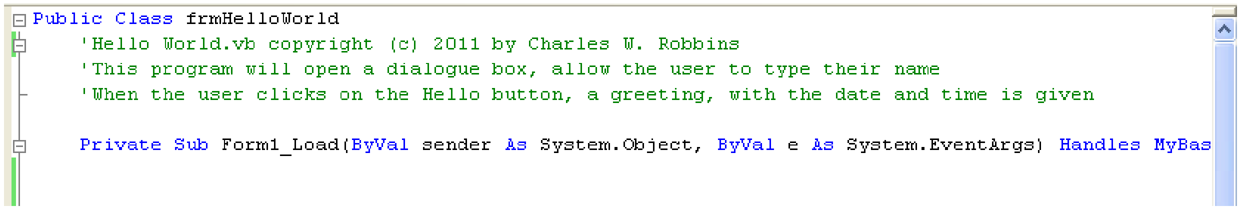
Adding Comments in Visual Basic to Communicate the Copyright

The comments we placed in the first three lines of the program will inform the individual opening and reading the code, but those user that may run the application without checking, the label on the bottom of the form with the copyright information is a great tool to alert the client to the rules of the program and what will the application do.

To begin the actual coding of the program, double click on the Hello command button. At the top of the program and before the line of code with Private Sub cmdHello_Click (), place the following comments with the single quote (‘) character. Remember, the single quote character (‘) will precede a comment and when the code is compiled, comments are ignored.

Type the following line of code:

```
'Hello World.vb copyright (c) 2006 by Charles W. Robbins
'This program will open a dialogue box, allow the user to type their name
'When the user clicks on the Hello button, a greeting, with the date and time is given
```



```
Public Class frmHelloWorld
    'Hello World.vb copyright (c) 2011 by Charles W. Robbins
    'This program will open a dialogue box, allow the user to type their name
    'When the user clicks on the Hello button, a greeting, with the date and time is given

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBas
```

Figure 2.19 – Adding a Copyright Statement

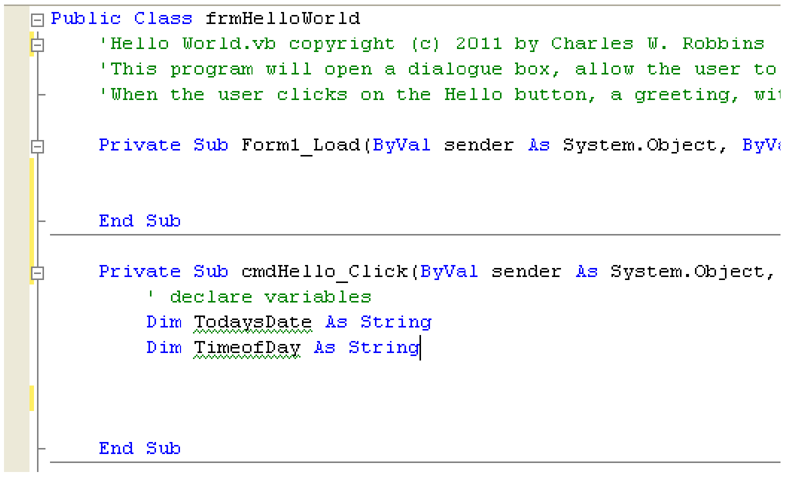
Declaring Variables in a Program with the Dimension Statement

When we are going to use a number, text string or object that may change throughout the life of the code, we create a variable to hold the value of that changing entity. In Visual Basic, the dimension statement is one of the ways to declare a variable at the procedure level. The other two ways are the Private and Public statements, which we will use in later chapters.

In our program, we will retrieve the date and time from the personal computer running the application and place the values in variables called **TodaysDate** and **TimeofDay**.

Type the following code:

```
' declare variables
    Dim TodaysDate as String
    Dim TimeofDay as String
```



```
Public Class frmHelloWorld
    'Hello World.vb copyright (c) 2011 by Charles W. Robbins
    'This program will open a dialogue box, allow the user to
    'When the user clicks on the Hello button, a greeting, wi

    Private Sub Form1_Load(ByVal sender As System.Object, ByV


    End Sub

    Private Sub cmdHello_Click(ByVal sender As System.Object,
        ' declare variables
        Dim TodaysDate As String
        Dim TimeofDay As String



    End Sub
```

Figure 2.20 – Declaring Variables with Dim Statements

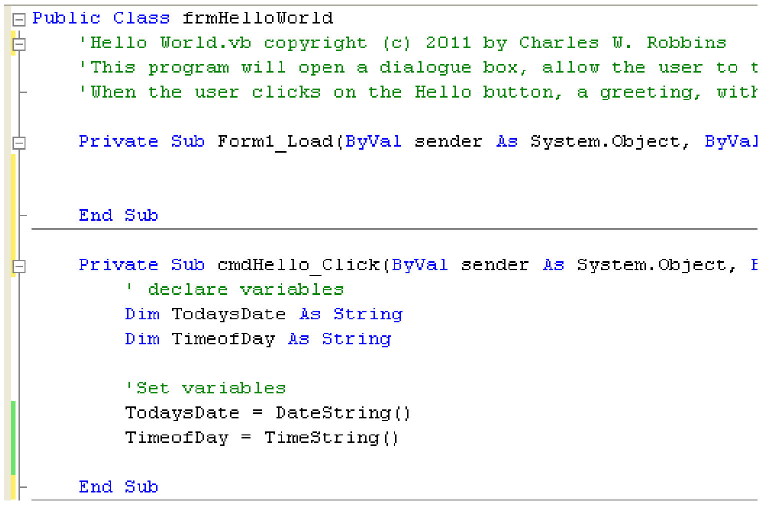
Notice that the variable name should be a word or a phrase without spaces that represents the value that the variable contains. If we want to hold a value of one's date of birth, we can call the variable, DateofBirth. The keywords Date and Birth are in sentence case with the first letter capitalized. There are no spaces in the name. Some programmers use the underscore character (_) to separate words in phrases. This is acceptable, but a double underscore (__) can cause errors if we do not detect the repeated character.

Setting Variables in a Program

Next, we will set the variables using the equal function. First TodaysDate will equal the DateString on the calendar in the personal computer. Second, the TimeofDay will equal the TimeString on the clock on the personal computer.

Type the following code:

```
'Set variables
     TodaysDate = DateString()
     TimeofDay = TimeString()
```



```
Public Class frmHelloWorld
    'Hello World.vb copyright (c) 2011 by Charles W. Robbins
    'This program will open a dialogue box, allow the user to t
    'When the user clicks on the Hello button, a greeting, with

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal


    End Sub

    Private Sub cmdHello_Click(ByVal sender As System.Object, B
        ' declare variables
        Dim TodaysDate As String
        Dim TimeofDay As String

        'Set variables
        TodaysDate = DateString()
        TimeofDay = TimeString()

    End Sub
```

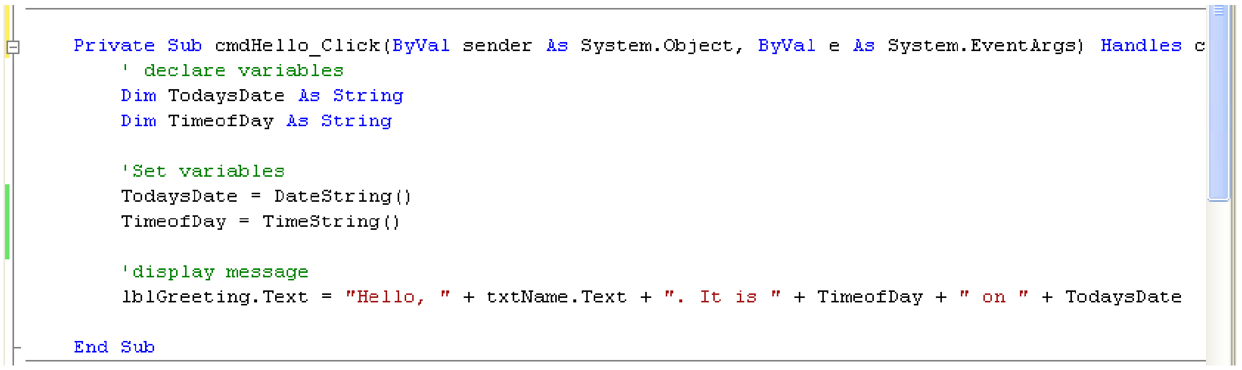
Figure 2.21 – Setting the Variables in the VBA Code

Using a Label to Communicate with Variables

The second message box is more difficult than the first, since we will incorporate the text strings in quotes with the variables. To bring text together or to concatenate the sentence, we will use the & character. When we want a space, we place a space in quotes “ ” or a period “.”

Go ahead and type the following code:

```
'display message
     lblGreeting.Caption = "Hello, " + txtName.Text + ". It is " + TimeofDay + " on " + TodaysDate
```



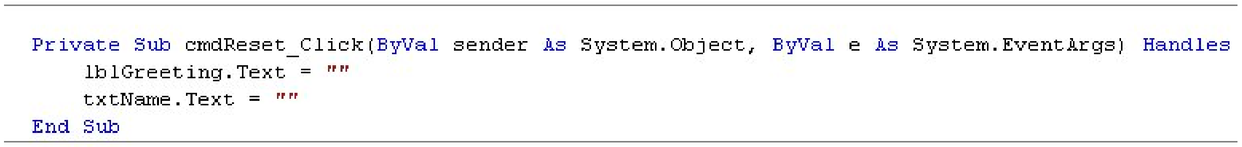
```
    Private Sub cmdHello_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles c
        ' declare variables
        Dim TodaysDate As String
        Dim TimeofDay As String

        'Set variables
        TodaysDate = DateString()
        TimeofDay = TimeString()

        'display message
        lblGreeting.Text = "Hello, " + txtName.Text + ". It is " + TimeofDay + " on " + TodaysDate

    End Sub
```

Figure 2.22 – Computing the Greeting with String Concatenation

The “Hello, ” + txtName.Text is concatenated with the + sign. We can use multiple + signs to add text strings and variables containing text together.

Resetting the Data

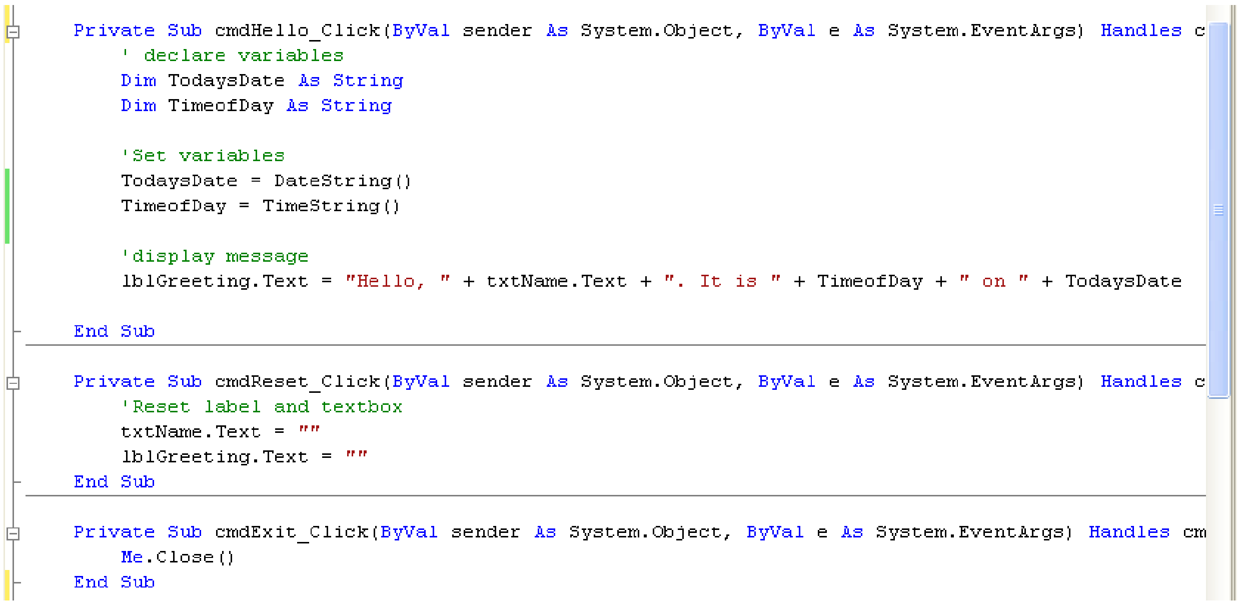
To clear the textbox or label containing the greeting, we will set the textbox for Name, txtName.text property to a blank entry by using the equal sign “=” and the null string “”. This makes the property blank. We will set the label for Greeting, lblGreeting.text property to a black entry by using the equal sign “=” and the null string “”, and this will make that property blank, also. Notice that after the control object name the dot (.) separates the suffix which is the name of the property for that object.



```
Private Sub cmdReset_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
    lblGreeting.Text = ""
    txtName.Text = ""
End Sub
```

Figure 2.23 – Computing the Reset Button by Clearing a Textbox and Label Caption

Exiting the Program



```
Private Sub cmdHello_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles c
    ' declare variables
    Dim TodaysDate As String
    Dim TimeofDay As String

    'Set variables
    TodaysDate = DateString()
    TimeofDay = TimeString()

    'display message
    lblGreeting.Text = "Hello, " + txtName.Text + ". It is " + TimeofDay + " on " + TodaysDate

End Sub

Private Sub cmdReset_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles c
    'Reset label and textbox
    txtName.Text = ""
    lblGreeting.Text = ""
End Sub

Private Sub cmdExit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cm
    Me.Close()
End Sub
```

Figure 2.24 – Exiting the Program

To exit this program, we will unload the application and end the program.
Type the following code:

```
“Unload and exit the program
   Me.Close()
```

Running the Program

After noting that the program is saved, press the F5 to run the Hello World application. The Hello World window will appear on the graphical display as shown in Figure 2.25. Notice the professional appearance and presentation of information in a clean dialogue box. The color of the background is neither black nor white, which would match the normal graphical display colors used by designers.

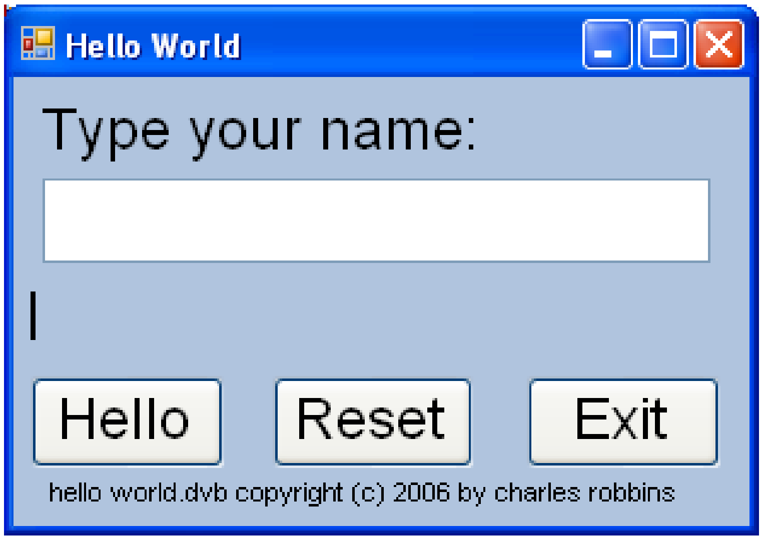
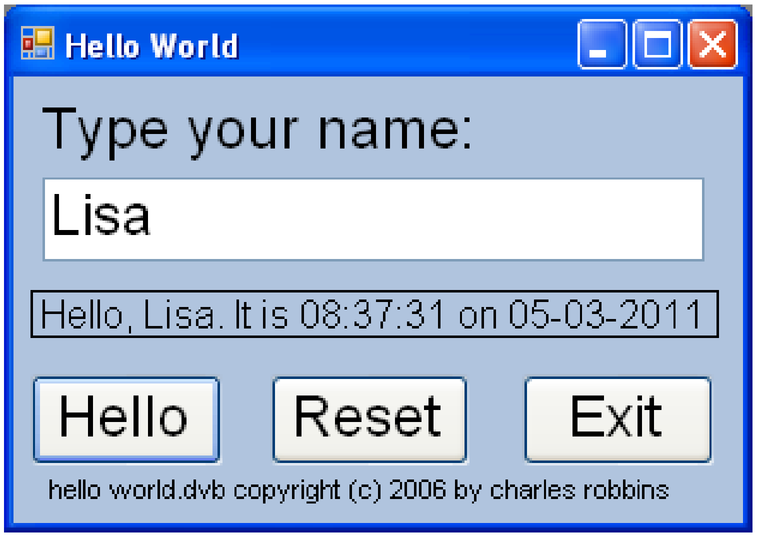


Figure 2.25 – Launching the Program

Type your name in the textbox just as we typed the name "Lisa" as shown in Figure 2.26. If we make a mistake, we can type over the text entry or press the Reset command button to clear the textbox. Press the Hello command button and a greeting is displayed like "Hello, Lisa. It is 8:37:31 on 05-03-2011 shown in Figure 2.26. After experimenting with our program, press the Exit command button to exit the application.



Lisa

Hello, Lisa. It is 08:37:31 on 05-03-2011

Figure 2.26 – Running the Program

If our program does not function correctly, go back to the code and check the syntax against the program shown in Figure 2.24. Repeat any processes to check or Beta test the program. When the program is working perfectly, save and close the project.

There are many variations of this Visual Basic Application we can practice and obtain information from a personal computer. While we are practicing with forms, we can learn how to use variables, strings and comments. These are skills that we want to commit to memory.

*** World Class CAD Challenge 90-1 * - Write a Visual Basic Application that displays a single input form, allow the user to type in their name, and when executed, the program will greet the user with information obtained from the computer.**

Continue this drill four times using some other form designs, each time completing the Visual Basic Project in less than 1 hour to maintain your World Class ranking.