

Hello World Program

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

- **Opening Visual C# Editor**
- **Beginning a New Visual C# Project**
- **Laying Out a User Input Form in Visual C#**
- **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 C# to Communicate the Copyright**
- **Declaring Variables in a Program**
- **Setting Variables in a Program**
- **Using a Label to Communicate with Variables**
- **Ending the Program**
- **Running the Program**

Open the Visual C# Editor

To open the Visual C# Editor in Microsoft Visual Studio is essential to creating the program to automate any process. In this version of the World Class CAD – Visual C#, we are using Visual C# 2010.

To open a new project, we select New Project on the left side of the Start Page.

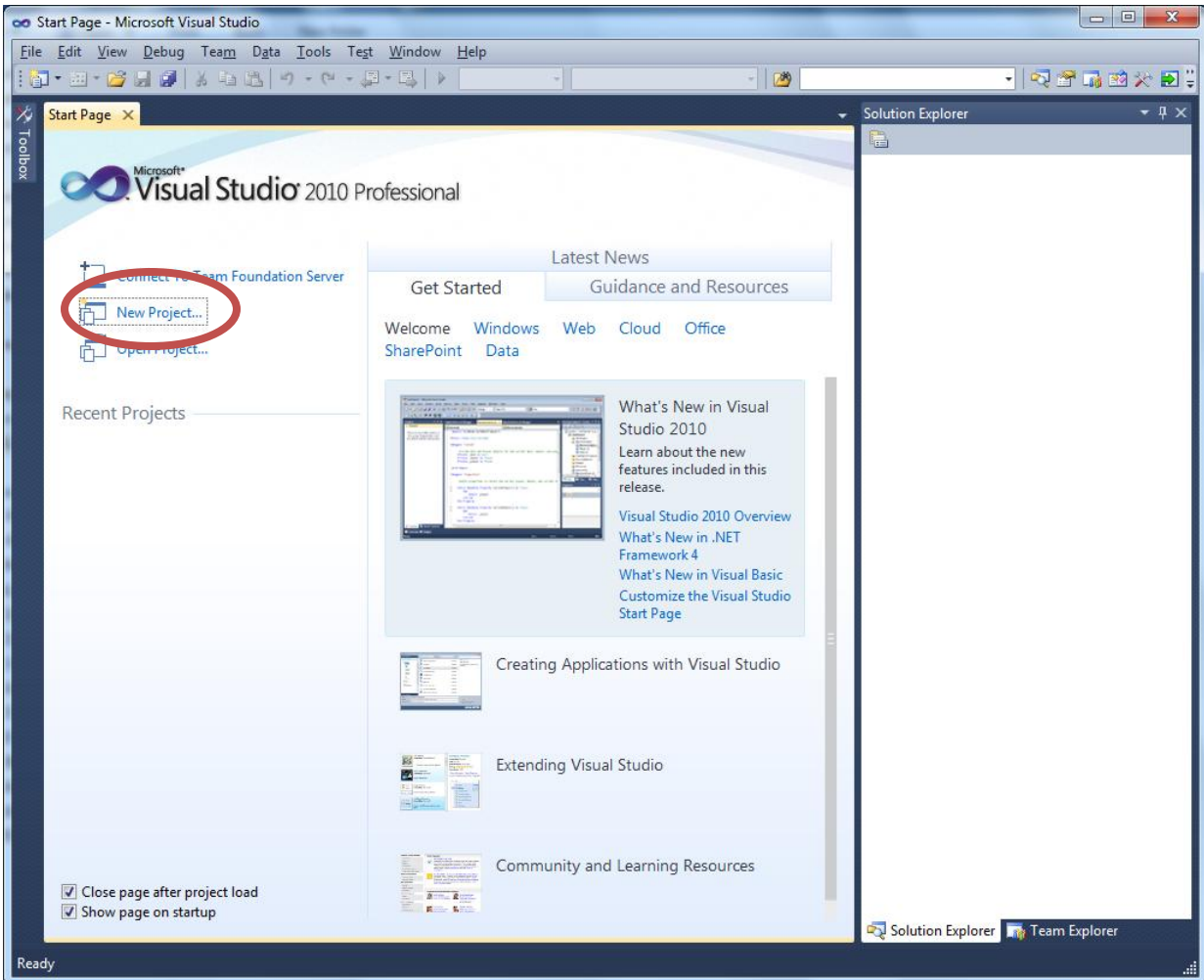


Figure 2.1 – The Start Page

We start a new Windows Application Project by picking the Windows under Visual C # in the left pan of the New Project window. Then we pick Windows Form Application in the center pane.

At the bottom of the Window, we name the project, Hello World. We make a folder for our projects called Visual C Sharp on the desktop, on our flash drive or in the Documents folder. We make another folder inside the first called Hello World Project. On the New Project window, we browse to the Hello World Project location. The solution name is the same as the project name.

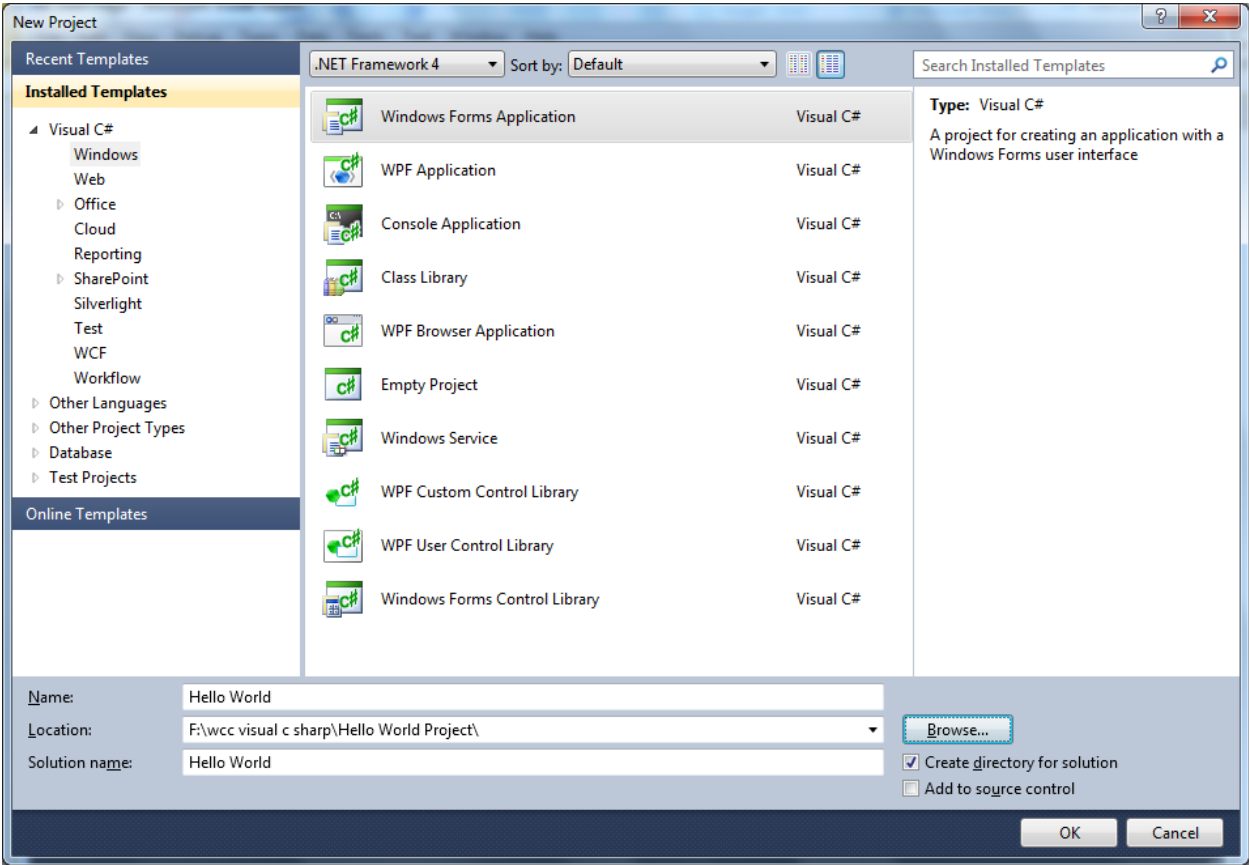


Figure 2.2 – New Project

The Hello World project opens with the form ready to design.

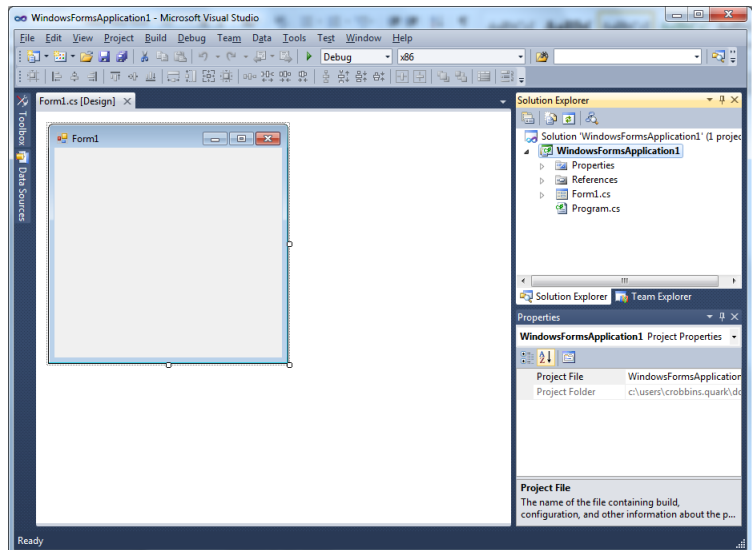


Figure 2.3 –The Hello World Program

Beginning a New Visual C# 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.

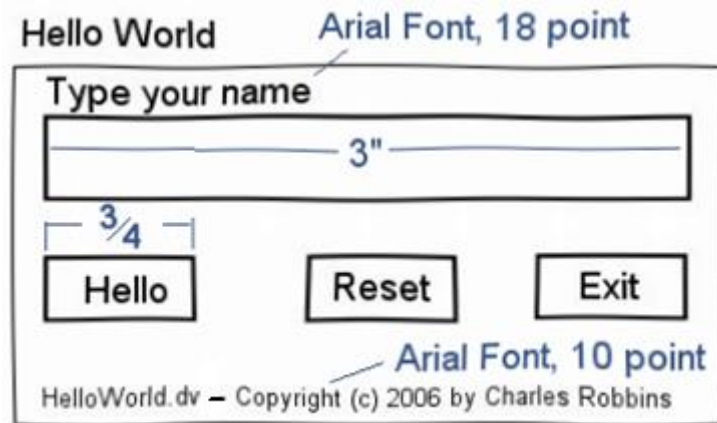


Figure 2.4 – Sketch of the Hello World Form

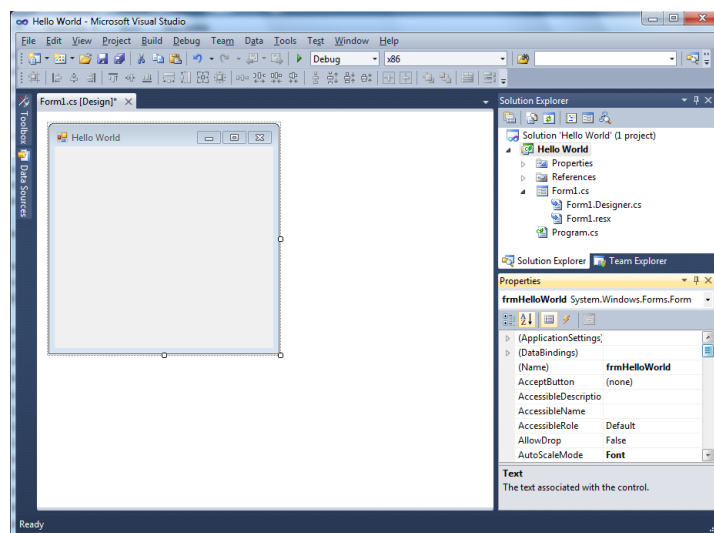


Figure 2.5 – Designing the Hello World Form in Visual C#

Laying Out a User Input Form in Visual C#

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 |

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

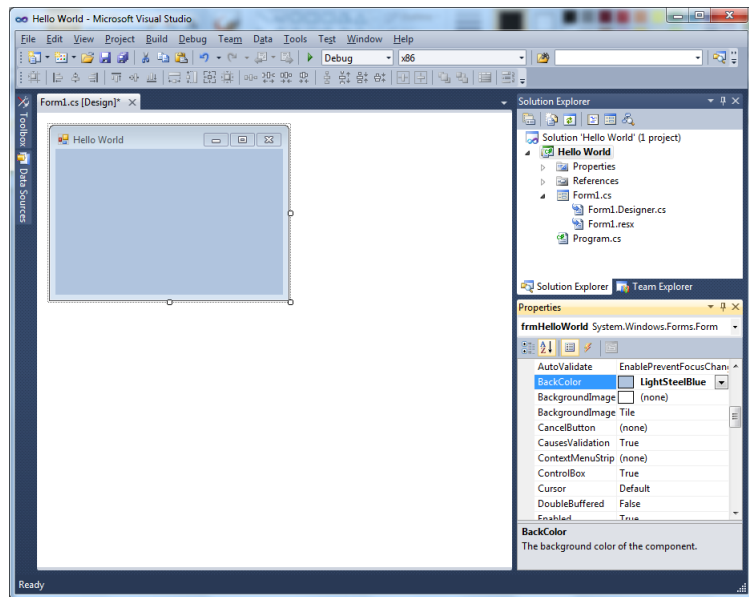


Figure 2.6 – Setting BackColor and other Properties

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.

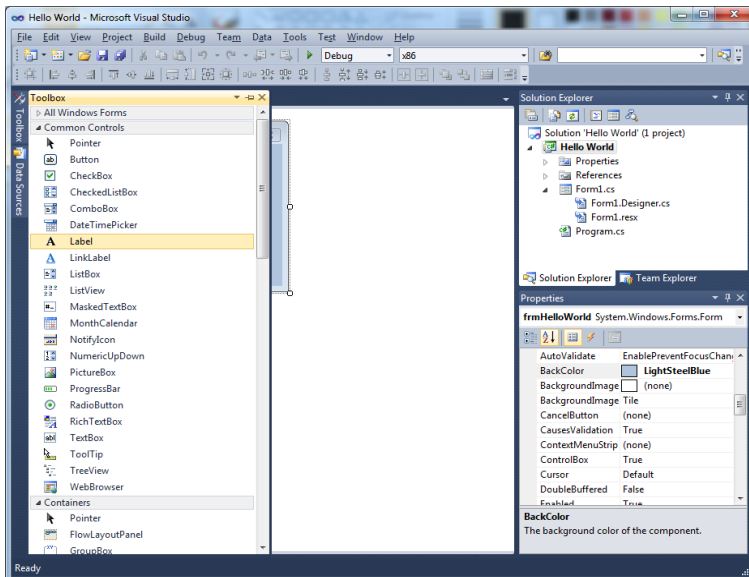


Figure 2.7 – Placing a Label on the Form

We will name the Label using a common Visual C# 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 C# Font window.

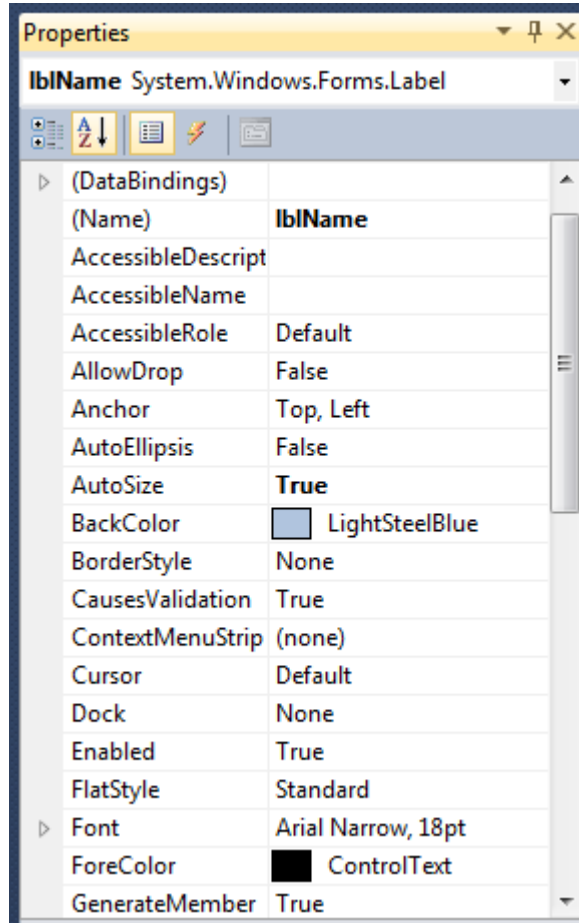


Figure 2.8 – Changing the Font Property

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.

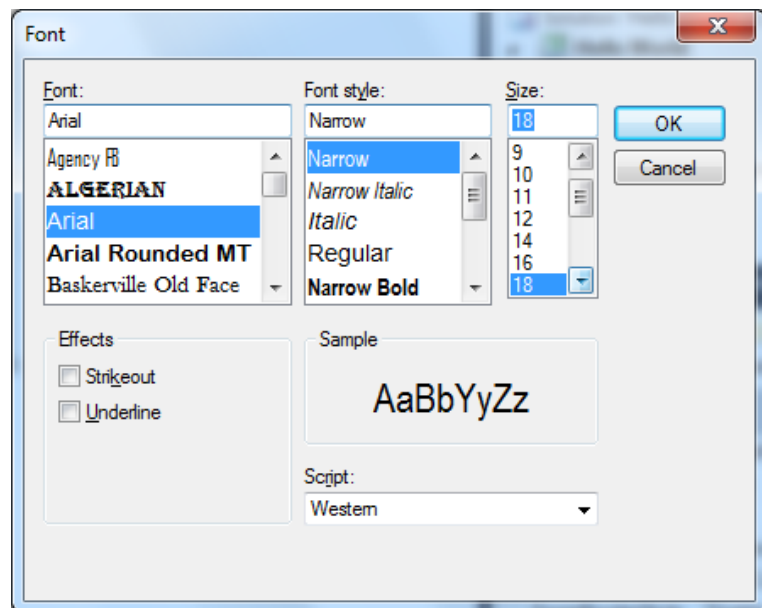


Figure 2.9 – The Font Window in Visual C#

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.

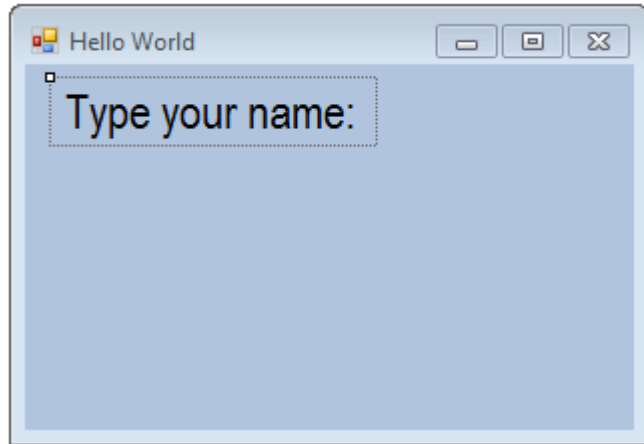


Figure 2.10 – The Finished Label on the Form

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.

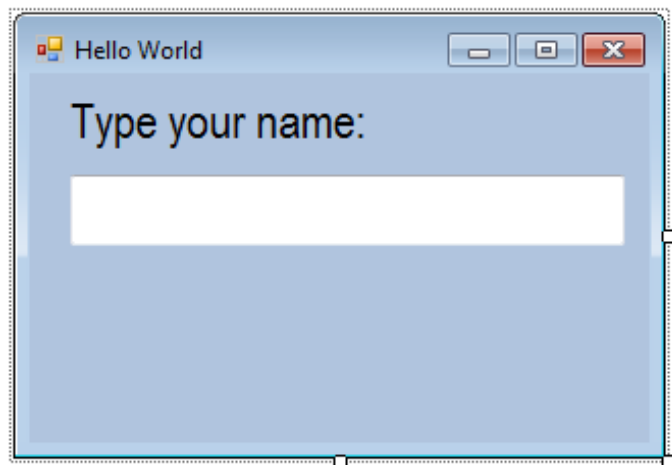


Figure 2.11 – Placing a TextBox on the Form

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 | 272, 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 C# Font window. Make the changes like we did on the Label and press OK to save the property.

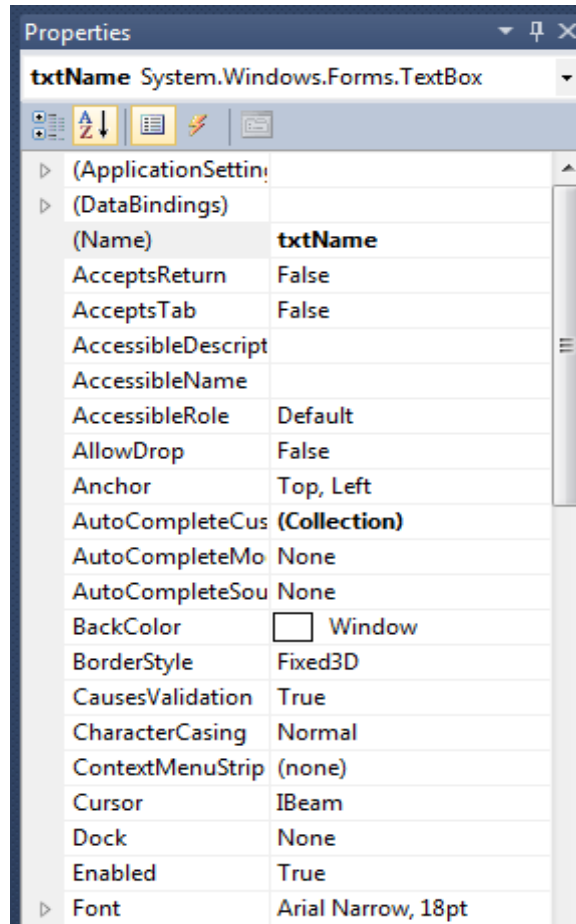


Figure 2.12 – Changing the (Name) to txtName

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.

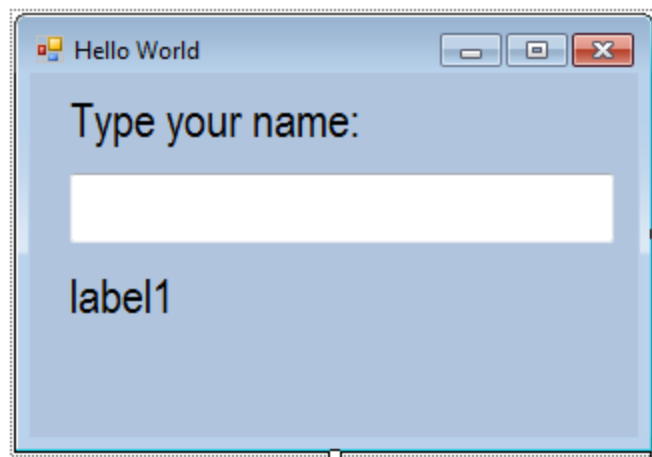


Figure 2.13 – Placing another Label on the Form

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

| | |
|-------------|--------------|
| Alphabetic | |
| (Name) | lblGreeting |
| BorderStyle | None |
| Font | Arial, 12 pt |
| Size | 2,20 |
| Text | (blank) |

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 C# Font window. Make the changes as we did before and press OK to save the property.

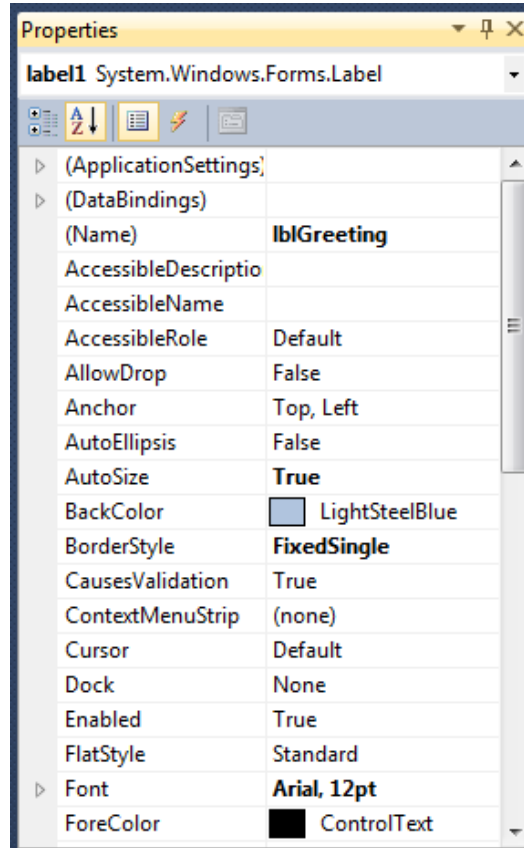


Figure 2.14 – Changing the (Name) to lblGreeting

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 |
| Text | Hello |
| Font | Arial, 12 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 C# Font window. Make the changes as we did before and press OK to save the property.

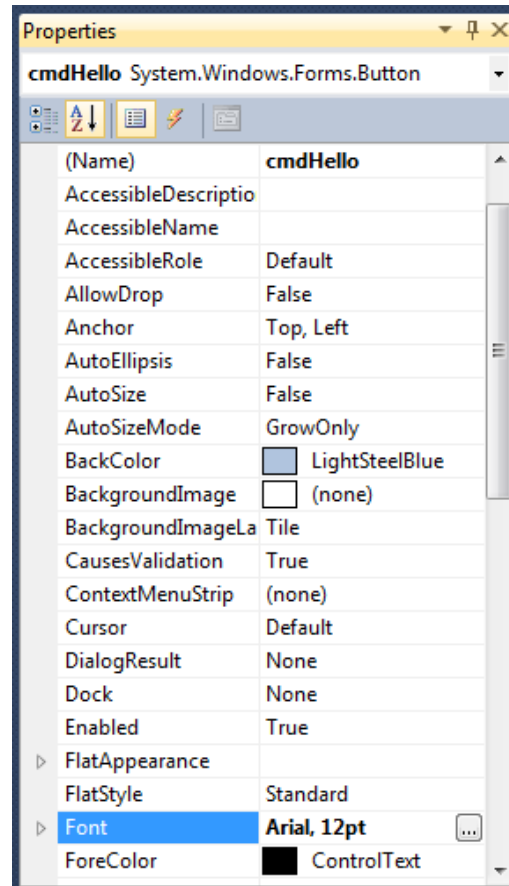


Figure 2.16 – Changing the (Name) to cmdHello

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.

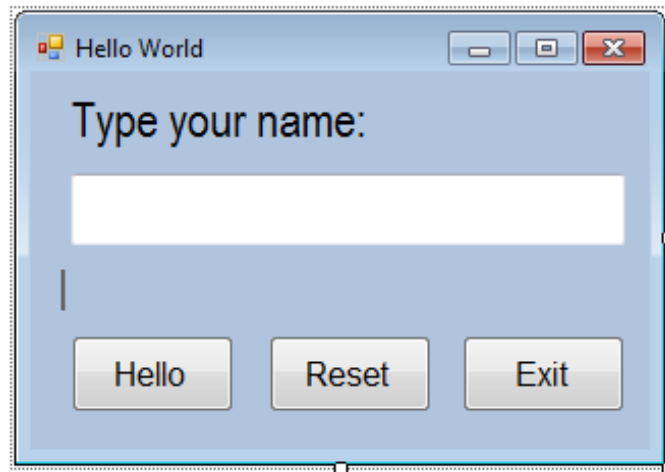


Figure 2.17 – Insert Two More Command Buttons

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 copyright (c) 2012 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.



Figure 2.18 – Adding a Copyright Statement

Adding Comments in Visual C# 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 of the ownership. This is for those user that may run the application without checking the label on the bottom of the form with the copyright information. It is a great tool to alert the client to the rules of the program and tell them what the application will do.

To begin the actual coding of the program, double click on the Hello command button. At the top of the program just after the open bracket under namespace Hello_world, place the following comments with two slashes (//). Remember, two slashes (//) will precede a comment and when the code is compiled, comments are ignored.

Type the following line of code:

```
//Hello World copyright (c) 2012 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
```

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Hello_World
{
    ///Hello World.vb copyright (c) 2012 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 partial class frmHelloWorld : Form
    {
        public frmHelloWorld()
        {
            InitializeComponent();
        }
    }
}

```

Figure 2.19 – Adding a Copyright Statement

Declaring Variables in a Program

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 C#, we will begin our program by adding two variables to the portion of code that will run when the Hello command button is clicked on. At each line of code, we will end it with a semicolon (;).

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

Type the following code:

```

//declare variables
String date;
String time;

```

```

namespace Hello_World
{
    ///Hello World.vb copyright (c) 2012 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 partial class frmHelloWorld : Form
    {
        public frmHelloWorld()
        {
            InitializeComponent();
        }

        private void cmdHello_Click(object sender, EventArgs e)
        {
            ///declare variables

            string date;
            string time;
        }
    }
}

```

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. We will use the DateTime function to capture the computer system date and time. We use the now attribute and we use the

`now.GetDateTimeFormats('d')[0];`
to assign the date and

`now.GetDateTimeFormats('t')[0];`
to assign the time.

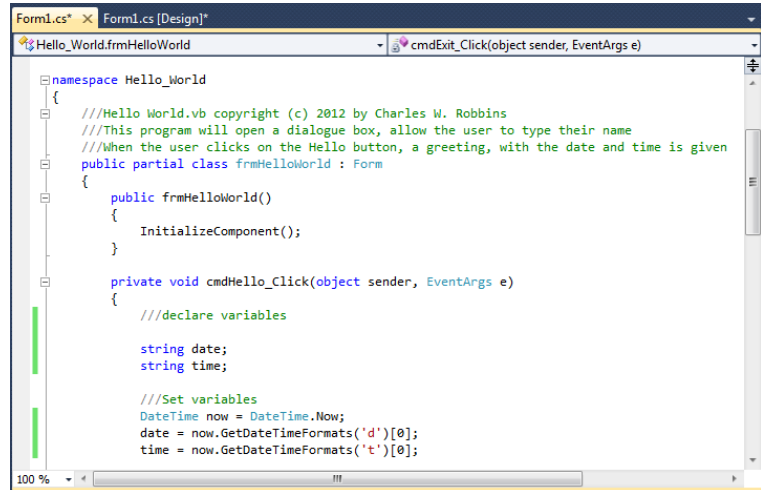
The image shows a screenshot of a Visual Studio code editor window titled 'Form1.cs [Design]*'. The code is written in VBA and is part of a namespace 'Hello_World'. It includes a class 'frmHelloWorld' which inherits from 'Form'. The class has a constructor 'frmHelloWorld()' that calls 'InitializeComponent()'. There is also a private event handler 'cmdHello_Click(object sender, EventArgs e)' which declares two string variables, 'date' and 'time', and then sets them using 'now.GetDateTimeFormats('d')[0]' and 'now.GetDateTimeFormats('t')[0]' respectively. The code is color-coded with green for comments, blue for keywords, and black for identifiers and literals. The editor interface includes a file explorer on the left, a search bar at the top, and a status bar at the bottom showing '100 %' zoom.

Figure 2.21 – Setting the Variables in the VBA Code

Type the following code for the variables:

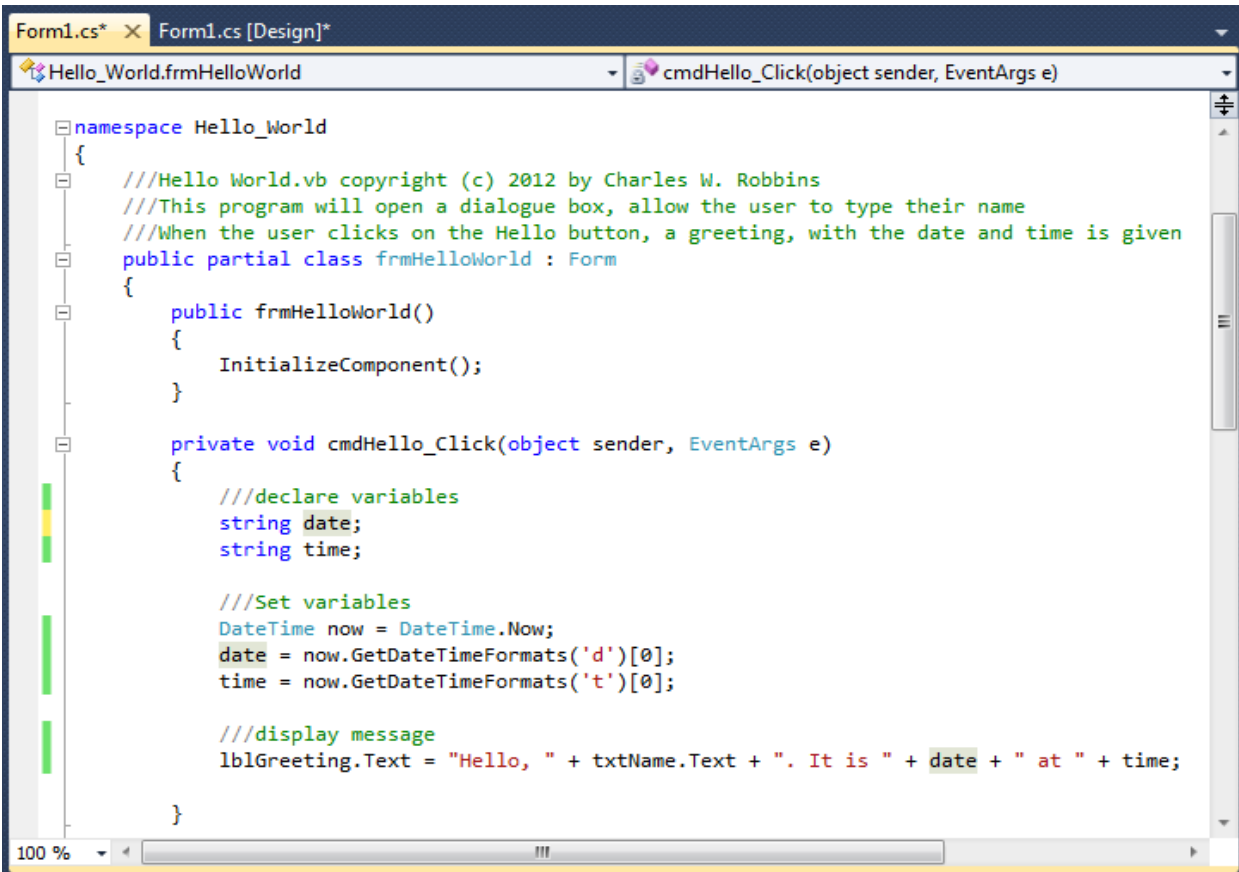
```
//Set variables  
DateTime now = DateTime.Now;  
date = now.GetDateTimeFormats('d')[0];  
time = now.GetDateTimeFormats('t')[0];
```

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.Text = "Hello, " + txtName.Text + ". It is " + date + " at " + time;
```



```
namespace Hello_World
{
    ///Hello World.vb copyright (c) 2012 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 partial class frmHelloWorld : Form
    {
        public frmHelloWorld()
        {
            InitializeComponent();
        }

        private void cmdHello_Click(object sender, EventArgs e)
        {
            ///declare variables
            string date;
            string time;

            ///Set variables
            DateTime now = DateTime.Now;
            date = now.GetDateTimeFormats('d')[0];
            time = now.GetDateTimeFormats('t')[0];

            ///display message
            lblGreeting.Text = "Hello, " + txtName.Text + ". It is " + date + " at " + time;
        }
    }
}
```

Figure 2.22 – Computing the Greeting with String Concatenation

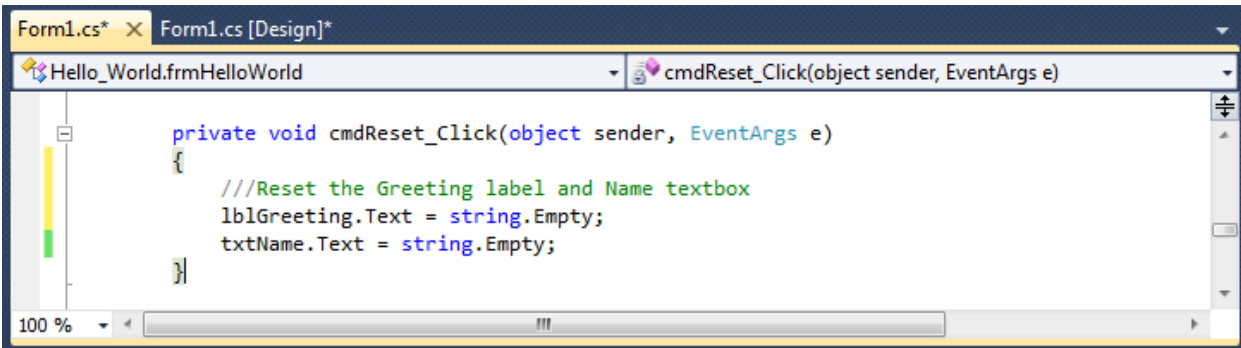
The “Hello, ” + txtName.Text is concatenated with the + sign. We can use multiple + signs to add text strings and variables containing text together. The string variable date and time are added into the message, so we can state the day and time of the greeting.

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.Empty. This makes the property blank. We will set the label for Greeting, lblGreeting.text property to a blank entry by using the equal sign “=” and the null string string.Empty, 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.

```
//Reset the Greeting label and Name textbox  
lblGreeting.Text = string.Empty;  
txtName.Text = string.Empty;
```

We can also use the lblGreeting.Text = “”; with two quotes and we would accomplish the same thing.

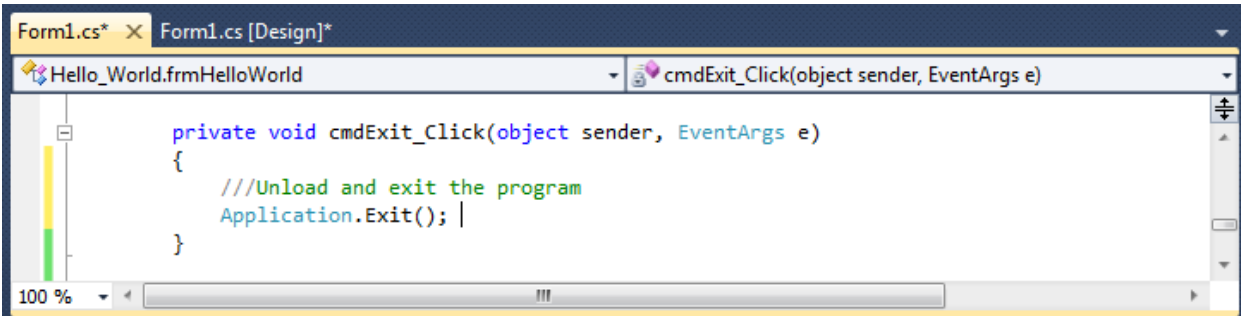


```
Form1.cs* X Form1.cs [Design]*
Hello_World.frmHelloWorld cmdReset_Click(object sender, EventArgs e)

private void cmdReset_Click(object sender, EventArgs e)
{
    ///Reset the Greeting label and Name textbox
    lblGreeting.Text = string.Empty;
    txtName.Text = string.Empty;
}
```

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

Exiting the Program



```
Form1.cs* X Form1.cs [Design]*
Hello_World.frmHelloWorld cmdExit_Click(object sender, EventArgs e)

private void cmdExit_Click(object sender, EventArgs e)
{
    ///Unload and exit the program
    Application.Exit();
}
```

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
Application.Exit()
```


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.

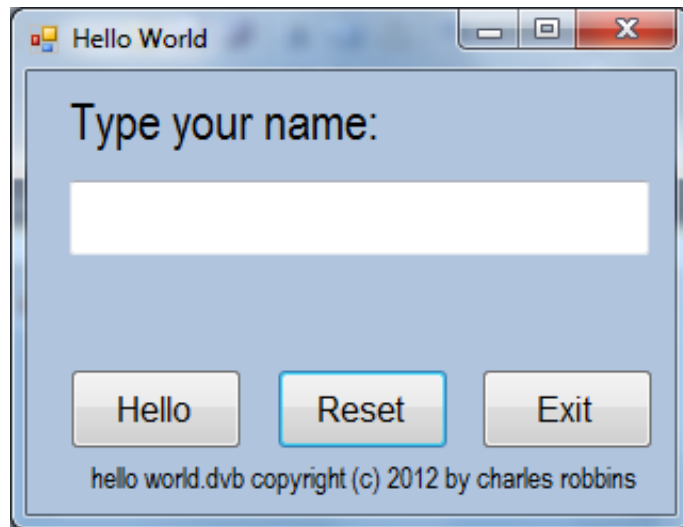


Figure 2.25 – Launching the Program

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.

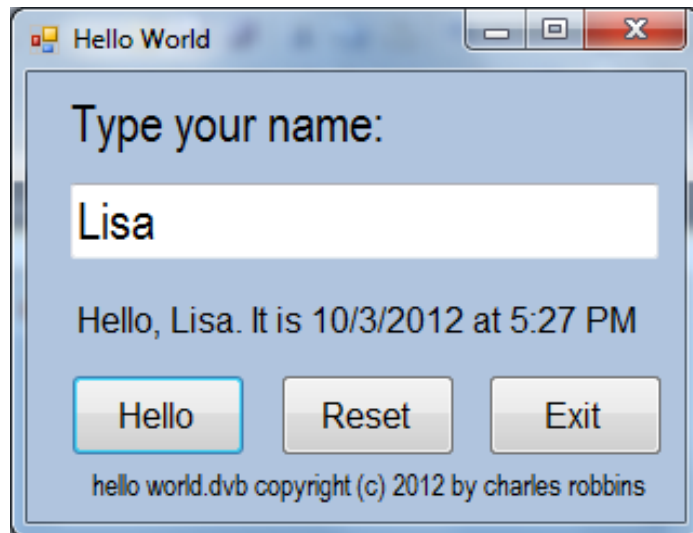


Figure 2.26 – Running the Program

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 C# 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 190-1 * - Write a Visual C# 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 C# Project in less than 1 hour to maintain your World Class ranking.