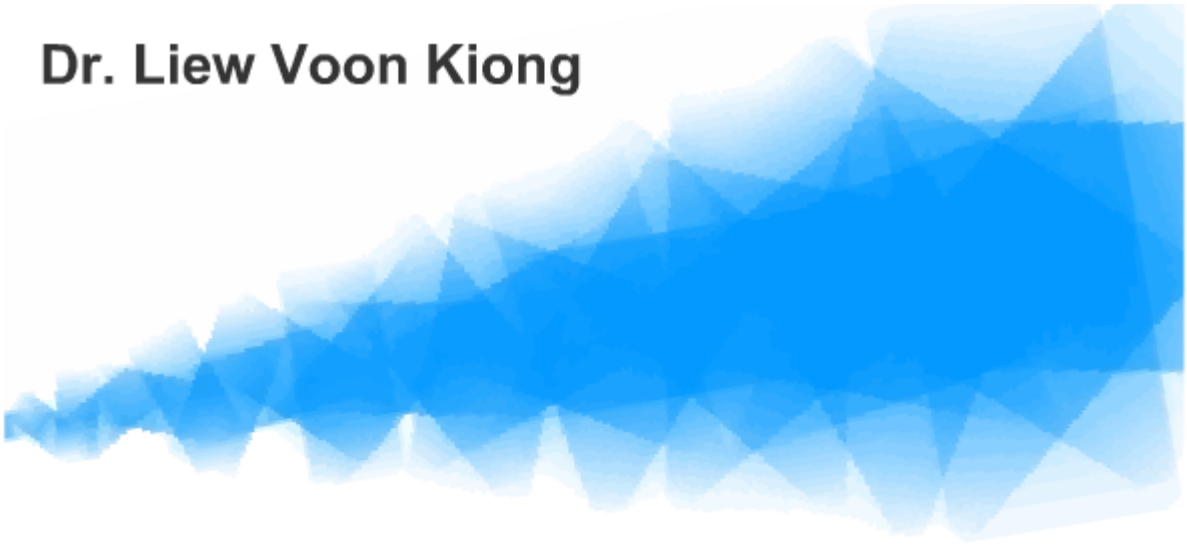
Dr. Liew Voon Kiong



Visual Basic 2010

Made Easy

Disclaimer

Visual Basic 2010 ® Made Easy- A complete tutorial for beginners is an independent publication and is not affiliated with, nor has it been authorized, sponsored, or otherwise approved by Microsoft Corporation.

Trademarks

Microsoft, Visual Basic, Excel and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.
All other trademarks belong to their respective owners.

Liability

The purpose of this book is to provide basic guides for people interested in Visual Basic 2010 programming. Although every effort and care has been taken to make the information as accurate as possible, the author shall not be liable for any error, harm or damage arising from using the instructions given in this book.

Copyright ® 2011 Liew Voon Kiong
All rights reserved. No Part of this e-book may be reproduced, in any form or by any means, without permission in writing from the author.

Acknowledgement

I would like to express my sincere gratitude to many people who have made their contributions in one way or another to the successful publication of this book.
My special thanks go to my children Xiang, Yi and Xun who have contributed their ideas and edited this book. I would also like to appreciate the support provided by my beloved wife Kim Huang and my youngest daughter Yuan. I would also like to thank the million of guests who have visited my **Visual Basic Tutorial** website at www.vbtutor.net for their support and encouragement.

About the Author

Dr. Liew Voon Kiong holds a bachelor degree in Mathematics, a master degree in Management and a doctoral degree in Business Administration. He has been involved in programming for more than 15 years. He created the popular online Visual Basic Tutorial at www.vbtutor.net in 1996 and since then the web site has attracted millions of visitors .It is one of the top searched **Visual Basic** websites in many search engines including Google. In order to provide more support for the Visual Basic hobbyists, he has written this book based on the Visual Basic 2010 tutorial. He is also the author of **Visual Basic 6 Made Easy, Visual Basic 2008 Made Easy** and **Excel VBA Made Easy.**

TABLE OF CONTENTS

Chapter 1: Introduction to Visual Basic 2010 **1**
1.1 A Brief Description of Visual Basic 2010 1
1.2 Navigating the Visual Basic 2010 Integrated Development Environment 2
1.3 Object Oriented Programming 5

Chapter 2: Designing the Interface **9**
2.1 Adding Controls to the Form 9
2.2 Setting Control Properties in Properties Window 10
2.3 Setting Control Properties Using Code 13

Chapter 3: Writing the Code **17**
3.1 Understanding Event Driven Programming 17
3.2 Understanding the Code Structure of an Event Procedure 19
3.3 Writing a Simple Multiplication Program 22
3.4 Writing a Program that Add Items to List 22

Chapter 4: Managing VB2010 Data **25**
4.1 Visual Basic 2010 Data Types 25
4.2 Managing Variables 27
4.3 Constants 30

Chapter 5: Performing Mathematical Operations **32**

Chapter 6 :String Manipulation **37**
6.1 String Manipulation Using + and & signs. 37

6.2 String Manipulation Using VB2010 Built-in Functions 40

Chapter 7: Controlling Program Flow 44

7.1 Conditional Operators 44

7.2 Logical Operators 45

7.3 Using the If Control Structure with the Comparison Operators 45

Chapter 8: Select Case Control Structure 50

Chapter 9: Looping 54

9.1 For....Next Loop 54

9.2 Do Loop 56

9.3 While ...End While Loop 58

Chapter 10: An Introduction to Functions 59

10.1 MsgBox ( ) Function 59

10.2 The InputBox( ) Function 63

Chapter 11: String Functions **65**
11.1 The Mid Function 65
11.2 The Right Function 67
11.3 The Left Function 68
11.4 The Trim Function 69
11.5 The Ltrim Function 69
11.6 The Rtrim Function 69
11.7 The InStr function 70
11.8 The Ucase and the Lcase Functions 70
11.9 The Chr and the Asc functions 71

Chapter 12: Mathematical Functions **73**
12.1 The Abs function 73
12.2 The Exp function 73
12.3 The Fix Function 74
12.4 The Int Function 75
12.5 The Log Function 75
12.6 The Rnd( ) Function 75
12.7 The Round Function 76
12.8 The Sqrt Function 77
12.9 Trigonometric Functions 78

Chapter 13: Formatting Functions **84**
13.1 Format Functions 84
13.2 Formatting Using ToString Method 87

Chapter 14: Formatting Date and Time **90**
14.1 Formatting Date and Time Using Predefined 90
Formats 91
14.2 Formatting Date and Time Using User-Defined 93
Formats
14.3 Formatting Time Using ToString Method

Chapter 15: Creating User-Define Functions **97**

Chapter 16: Using Advanced controls **106**
16.1 The Check Box 106
16.2 The Radio Button 112
16.3 The List Box 116
16.4 The Combo Box 121

Chapter 17: Creating a Simple Web Browser **128**

Chapter 18: Errors Handling **131**
18.1 Introduction 131
18.2 Using On Error GoTo Syntax 132
18.3 Errors Handling Using Try.....Catch....End Try Structure 134

Chapter 19: Reading and write Files **136**
19.1 Introduction 136
19.2 Reading a Text File 136
19.3 Writing to a Text File 140

Chapter 20: Creating and Managing Graphics **145**
20.1 Introduction 145
20.2 Creating the Graphics Object 145
20.3 Creating the Pen Object 146
20.4 Drawing a Line 147

20.5 Creating a Rectangle 148
20.6 Customizing Line Style of the Pen Object 150
20.7 Drawing an Ellipse 151
20.8 Drawing a Circle 153
20.9 Drawing Text 155
20.10 Drawing a Polygon 157
20.11: Drawing a Pie 160
20.12 Drawing and Filling an Ellipse 162
20.13 Drawing and Filling a Polygon 163
20.14 Drawing and Filling a Pie 165

Chapter 21: Creating Arrays 167
21.1 Introduction to Arrays 167
21.2 Dimension of an Array 167
21.3 Declaring an Array 168

Chapter 22: Using Timer 176
22.1 Creating a Clock 176
22.2 Creating a Stopwatch 178
22.3 Creating a Digital Dice 180

Chapter 23: Creating Animation 183
23.1 Moving an Object 183
23.2 Creating Animation using Timer 185
23.3 Creating a Simple Lucky Seven Slot Machine 188
23.4 Creating a Graphical Slot Machine 190

Chapter 24: Adding Menus and Toolbar 195
24.1 Adding Menus 195
24.2 Adding ToolBar Icons 200

Chapter 25: Packaging Applications for Distribution **208**
208
25.1 Creating the Setup Program Using Publish Wizard 213
25.2 Testing your Install Program

Appendix I : The ASCII Table 215
Appendix II : List of Culture Codes 216
Index 220

Chapter 1

Introduction to Visual Basic 2010

- A brief description of Visual Basic 2010
- Getting to know the Visual Basic 2010 Integrated Development Environment

1.1 A brief Description of Visual Basic 2010

Visual Basic 2010 is the latest version of Visual Basic launched by Microsoft in 2010. It is almost similar to Visual Basic 2008 but it has added many new features. Visual Basic has gone through many phases of development since the days of BASIC that was built for DOS. BASIC stands for **B**eginners' **A**ll-purpose **S**ymbolic **I**nstruction **C**ode. The program code in Visual Basic resembles the English language. Different software companies had produced many different versions of BASIC for DOS, such as Microsoft QBASIC, QUICKBASIC, GWBASIC, and IBM BASICA and more. Then, Microsoft launched the first graphical BASIC, Visual Basic Version 1 in 1991. It is GUI based and especially developed for MS window. Since then Microsoft slowly phased out the DOS versions of BASIC and completely replaced them by Visual Basic.

Visual Basic was initially a functional or procedural programming language until the popular Visual Basic 6. Then, Microsoft transformed Visual Basic into a more powerful object oriented programming language by launching Visual Basic.Net, Visual Basic 2005, Visual Basic 2008 and the latest Visual Basic 2010. Visual Basic 2010 is a full-fledged Object-Oriented Programming (OOP) Language; it has caught up with other OOP languages such as C++, Java, C# and others. However, you do not have to know OOP to learn VB2010. In fact, if you are familiar with Visual Basic 6, you can learn VB2010 effortlessly because the

syntax and interface are almost similar. Visual Basic 2010 Express Edition is available for free download from the Microsoft site as shown below: http://www.microsoft.com/visualstudio/en-us/products/2010-editions/express

1.2 Navigating the Visual Basic 2010 Integrated Development Environment

1.2.1 The Start Page

When you launch Visual Basic 2010 Express, you can see the start page of the Integrated Development Environment, as shown in Figure 1.1.

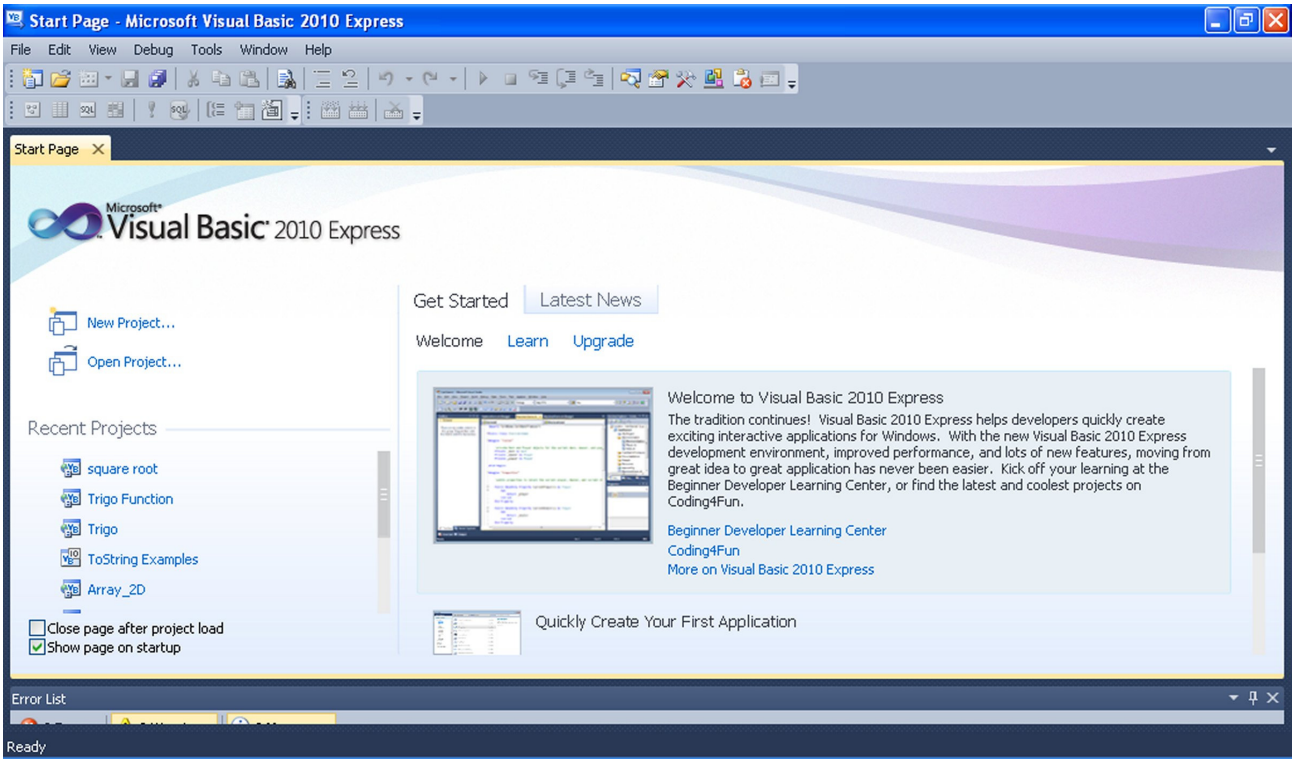


Figure 1.1: The VB2010 IDE Start Page

The IDE consists of a few panes, namely:

- The Recent Projects Pane- it shows the list of projects that you have created recently.
- The Get Started Pane- It provides some helpful tips so that you can quickly develop your new application.
- The Latest News pane- It provides latest online news about Visual Basic 2010 Express. It will announce new releases and updates.

Besides that, it also shows two icons, New Project and Open Project.

1.2.2 The New Project Dialog

When you click on the **New Project** icon, the Visual Basic 2010 New Project dialog will appear, as shown in Figure 1.2

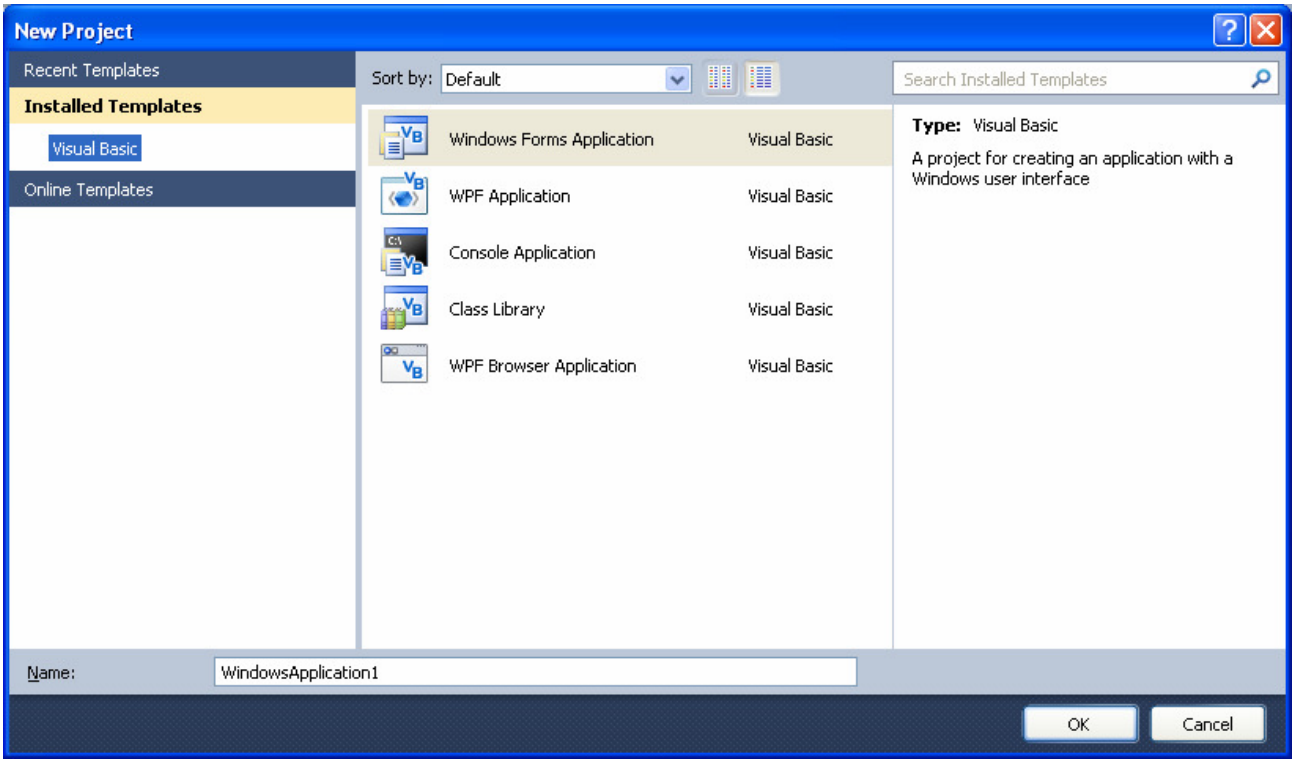


Figure 1.2: VB2010 New Project Dialog

The dialog box offers you five types of projects that you can create. They are Windows Form Application, WPF Application, Console Application, Class Library

and WPF Browser Application. As we are going to create a standard Windows application, we will select Windows Forms Application. At the bottom of this dialog box, you can change the default project name **WindowsApplication1** to some other name you like, for example, **MyFirstApplication**. After you have renamed the project, click OK to go into the Designer interface.

1.2.3 The Designer Interface

The VB2010 IDE Designer interface is shown in Figure 1.3. The Designer consists of the **Menu bar**, the **Toolbars**, an empty **Form**, the **Solution Explorer** and the **Properties Window**.

The VB2010 Designer environment that appears on your PC or laptop might not be the same here, depending how you customize it. You can customize your interface by dragging the windows and dock them or let them float. You can also hide them. To dock a window, you drag its title bar and drag it to the side, top or bottom of the workspace or another window. In Figure 1.3, we have dragged the Solution Explorer and the Properties Window to the side and docked them. You can also resize the docked window by dragging the side of the window. To free up and float the docked window, you just drag its title bar and move it away from the edge of the workspace.
If you do not see a particular window such as the properties window, you can click on the View menu and click the name of the window, that particular window will appear.

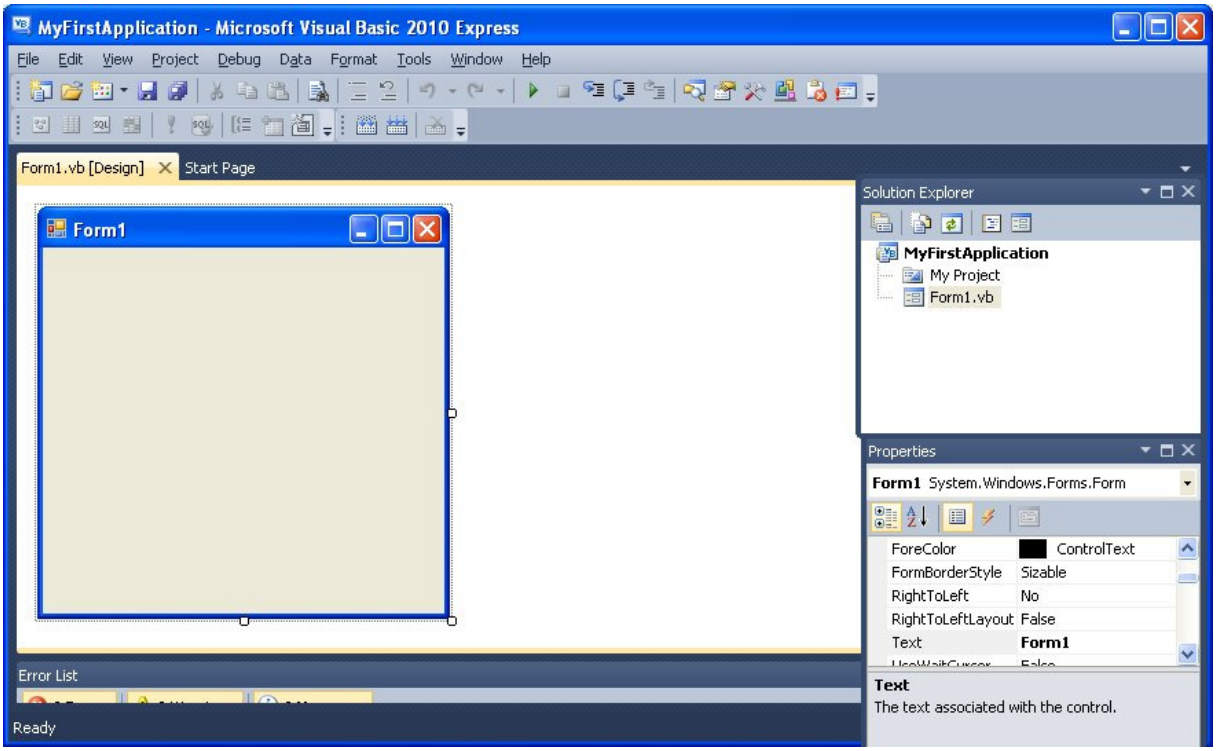


Figure 1.3: VB2010 IDE with A New Form

- **Form**-The Form is the first place to build your application. It is the place to design the user interface.
- **Solution Explorer** -The solution explorer displays a list of projects, files and other components that you can easily browse and access. For example, it displays My Project and Form1.vb in Figure 1.3
- **Properties Window-** This is the place to set the properties of the objects in your application. The objects include the default form and the controls you place in the form. We will learn more about setting properties later.

1.3 Understanding the Concept of Object Oriented Programming

The main different between VB2010 and Visual Basic 6 is that is it is a full Object Oriented Programming Language while VB6 may have OOP capabilities, it is not fully object oriented. In order to qualify as a fully object oriented programming language, it must have three core technologies namely **encapsulation,**

inheritance and **polymorphism**. Read more about the three terms in the box below:

Encapsulation refers to the creation of self-contained modules that bind processing functions to the data. These user-defined data types are called classes. Each class contains data as well as a set of methods, which manipulate the data. The data components of a class are called instance variables and one instance of a class is an object. For example, in a library system, a class could be member, and John and Sharon could be two instances (two objects) of the library class.

Inheritance

Classes are created according to hierarchies, and inheritance allows the structure and methods in one class to be passed down the hierarchy. That means less programming is required when adding functions to complex systems. If a step is added at the bottom of a hierarchy, then only the processing and data associated with that unique step needs to be added. Everything else about that step is inherited. The ability to reuse existing objects is a major advantage of object technology.

Polymorphism

Object-oriented programming allows procedures about objects to be created whose exact type is not known until runtime. For example, a screen cursor may change its shape from an arrow to a line depending on the program mode. The routine to move the cursor on screen in response to mouse movement would be written for "cursor," and polymorphism allows that cursor to take on whatever shape is required at run time. It also allows new shapes to be integrated easily.

VB2010 is a fully Object Oriented Programming Language, just like other OOP such as C++ and Java. It is different from the earlier versions of VB because it focuses more on the data itself while the previous versions focus more on the actions. Previous versions of VB are **procedural** or **functional** programming language. Some other procedural programming languages are C, Pascal and Fortran.

VB2010 allows users to write programs that break down into modules. These modules represent the real-world objects; we also call them classes or types. An object can be created out of a class , it is an instance of the class. A class can also comprise subclass. For example, apple tree is a subclass of the plant class and the apple in your backyard is an instance of the apple tree class. Another example is student class is a subclass of the population class while a student with the name John is an instance of the student class. A class consists of data members as well as methods. In VB2010, the program structure to define a population class can be written as follows:

```
Public Class Population
'Data Members
Private Name As String
Private Birthdate As String
Private Gender As String
Private Age As Integer
'Methods
Overridable Sub ShowInfo( )
MessageBox.Show(Name)
MessageBox.Show(Birthdate)
MessageBox.Show(Gender)
MessageBox.Show(Age)
End Sub
```

```
End Class
```

After you have created the population class, you can create a subclass that inherits the attributes or data from the population class. For example, you can create a student class that is a subclass of the population class. Under the student class, you do not have to define any data fields that were already defined under the population class; you only have to define the data fields that are different from an instance of the population class. For example, you may want to include StudentID and Address in the student class. The program code for the StudentClass is as follows:

```
Public Class Student
Inherits Population
Public StudentID as String
Public Address As String
Overrides  Sub ShowInfo( )
MessageBox.Show(Name)
MessageBox.Show(StudentID)
MessageBox.Show(Birthdate)
MessageBox.Show(Gender)
MessageBox.Show(Age)
MessageBox.Show(Address)
End Sub
```

Summary

- In section 1.1, you learned about the evolution of Visual Basic.
- In section 1.2, you have learned how to launch the start page, the new

project dialog and the designer interface. You have also learned that the designer interface consists of the Form, the Solution Explorer and the Properties window.

- You have also learned some basic concepts of object oriented programming, which comprises encapsulation, polymorphism and inheritance.

Chapter 2

Designing the Interface

- ❖ Learning to Design the Interface
- ❖ Adding controls
- ❖ Setting Control Properties

2.1 Adding Controls to the Form

The first step in creating a new VB2010 project is to design the interface of the application. You design an interface by adding controls to the form and then set their properties. You can add controls from the Toolbox. To see the Toolbox window, you can use the short-cut keys Ctrl+Alt+X or click on the Toolbox icon on the toolbar on top of the designer environment. The Toolbox consists of many useful controls such as Button, TextBox, Label, ComboBox, CheckBox and more, as shown in Figure 2.1

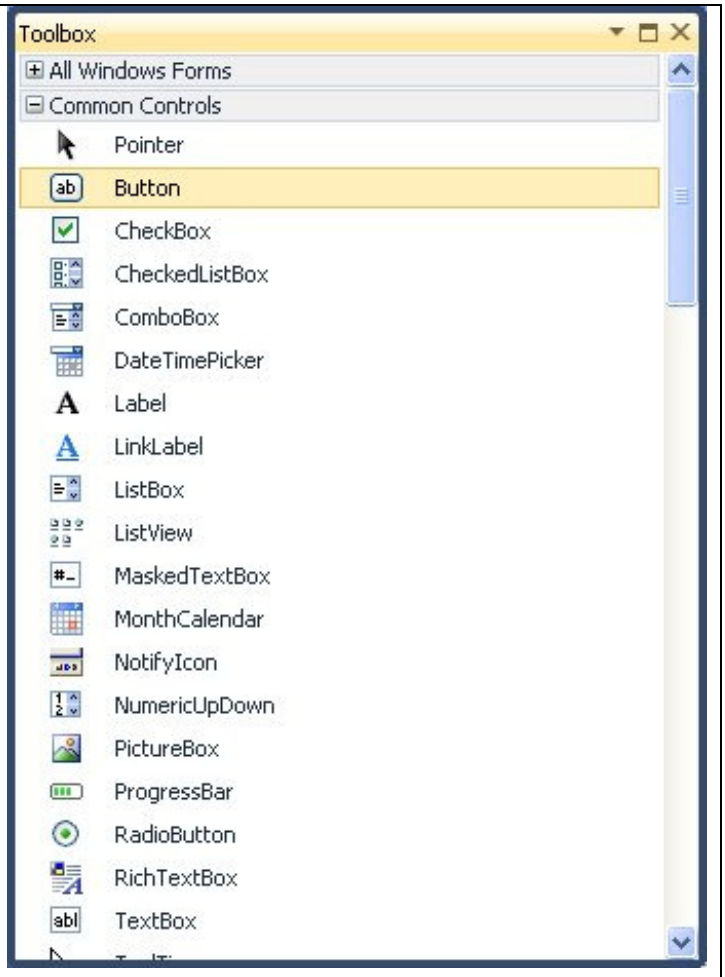


Figure 2.1

The Visual Basic 2010 Control Toolbox consists of all the controls essential for developing a VISUAL BASIC 2010 application. Controls in VB2010 are useful tools that can perform various tasks. We categorized into Common Controls, Containers, Menus, Toolbars, Data, Components, Printings and Dialogs. Now, we will focus on the common controls. Some of the most used common controls are Button, Label, ComboBox, ListBox, PictureBox, TextBox and more. To add a control to the form, just drag the particular control and drop it into the form. After putting it into the form, you can change its size and position easily. You can add as many controls as you want, but avoid crowding the form.

2.2 Setting the Control Properties Using Properties Window

To customize the interface to the users, you need to set the properties of the controls, from the form itself to the controls you add to the form. You can set the properties of the controls in the properties window at design time or by using the code. We shall learn how to set the control properties using the properties window first.

To set the properties of an object, right click on the object and choose properties in the dialog that appears to view the properties window. In the properties window, you can change the values of the properties that appear in a dropdown list, as shown in Figure 2.2. It is a typical Properties window for a form. The default text of the Text property is Form1, its default name is also Form1. You can change the title of the text to whatever title you like by editing the text.

The properties of the object appear in a list in the left column while the items listed in the right column represent the states or values of the properties. You can set the properties by highlighting the items in the right column then change them by typing or by selecting options. For example, to change the form's title to any name that you like, simple click in the box on the right of the Text property and

type in the new name. In the properties window, the item appears at the top part is the currently selected object.

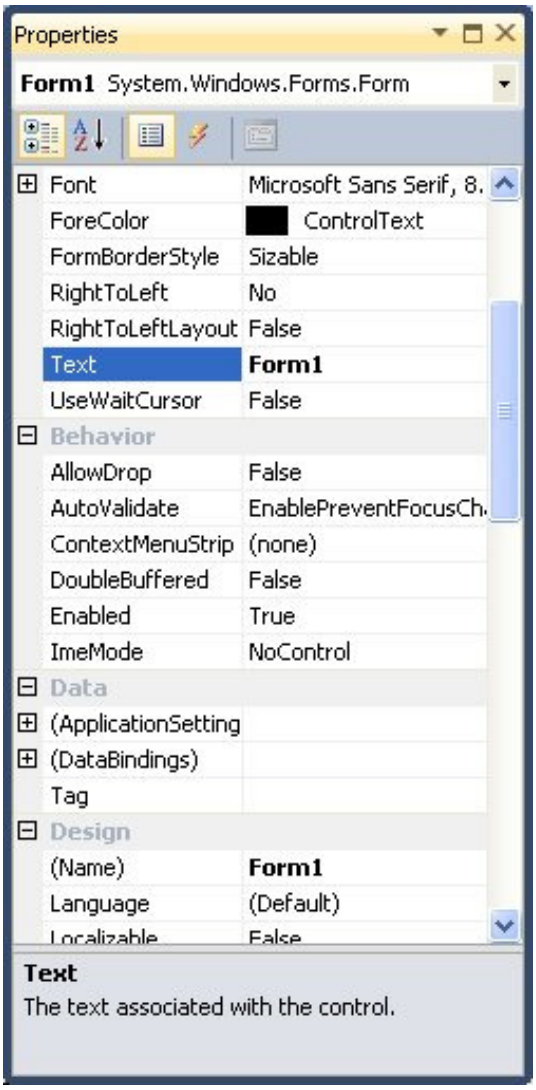


Figure 2.2 : The Properties Window

Example 2.1: Creating a Simple Program that display a welcoming message

In this example, we will create a simple program that will display a welcome message when you load the form. First, change the properties of the form as follows:

Property	Value
Name	WelcomeMsgFrm
BackColor	Pink(background of the form)
Font	Microsoft Sans Serif Size 10 and Bold
ForeColor	White (The color of text on title bar)
Text	Visual Basic 2010 (Text on title bar)

Table 2.1 : Properties of the Form

Next, insert a label into the form and set its properties as follows:

Property	Value
Autosize	False
Name	MsgLbl
BackColor	Purple
BorderStyle	FixedSingle
Font	Microsoft Sans Serif Size 10 and Bold
ForeColor	White
Text	Blank it

Table 2.2 : Properties of the Label

Next, click on the Form and enter the following code:

```
Private Sub WelcomeMsgFrm_Load(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles MyBase.Load

    MsgLbl.Text = "Welcome to VB2010 Programming"

End Sub
```

Run the program and the message will be displayed on the label, as shown in Figure 2.3

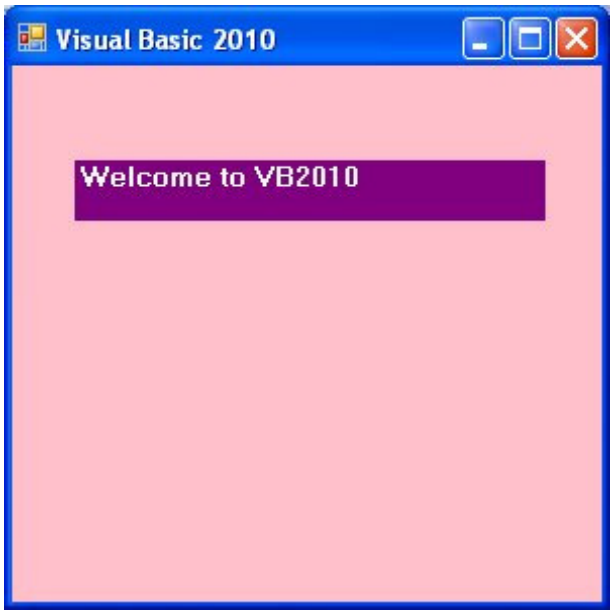


Figure 2.3

2.3 Setting Control Properties using Code

You can also change the properties of the object using code. The syntax to manipulate the properties of an object is

```
Object.property=property_Value
```

For example,

```
TextBox1.Text="Welcome to VB2010"
TextBox2.Text=100
```

The above code sets the text property of TextBox1 to display the text "Welcome to VB2010" and set the value of TextBox2 to 100.

Other properties you can change to give special effects at runtime are color, shape, animation effect and so on. For example, the following code will change the form color to yellow every time the form is loaded. VB2010 uses RGB (Red, Green, Blue) to determine the colors. The RGB code for yellow is 255, 255, 0. Me in the code refers to the current form and Backcolor is the property of the form's background color. The formula to assign the RGB color to the form is

Color.FormArbg(RGB code). Now, click on the form to go into the code window. Next, enter the following code between the opening statement Private Sub and the closing statement End Sub, as shown below. You don't have to worry about the code and the code structure yet; we will explain that in chapter 3.

```
Public Class Form1
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    Me.BackColor = Color.FromArgb(255, 255, 0)
End Sub
End Class
```

Now Press F5 and you will see a form appear with a yellow background, as shown in Figure 2.4
You may also use the following procedure to produce the same effect.

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
  Me.BackColor = Color.Yellow
End Sub
```

Figure 2.4: The form with yellow background

Here are some of the common colors and the corresponding RGB codes. You can always experiment with other combinations, but remember the maximum number for each color is 255 and the minimum number is 0. The table below shows some of the common colors with their corresponding codes.

Color	RGB code	Color	RGB code	Color	RGB Code
	255,0,0		255, 255, 0		255, 165, 0
	0,255,0		0, 255, 255		0, 0, 0
	0, 0, 255		255, 0, 255		255, 255, 255

Table 2.5: Common colors and their corresponding RGB codes

The following is a program that allows the user to enter the RGB code into three different Textboxes and when he or she clicks the Display Color button, the background color of the form changes according to the RGB code.

The code

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Dim rgb1, rgb2, rgb3 As Integer
        rgb1 = TextBox1.Text
```

```
        rgb2 = TextBox2.Text
        rgb3 = TextBox3.Text
        Me.BackColor = Color.FromArgb(rgb1, rgb2, rgb3)
End Sub
```

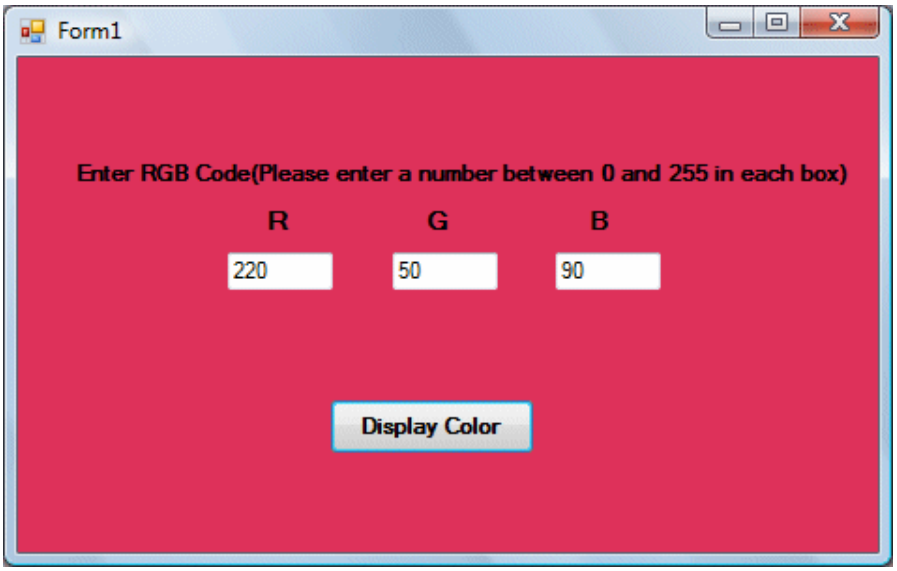


Figure 2.6: The RGB Program

Summary

- In section 2.1, you have learned how to add controls to the form from the Toolbox.
- In section 2.2, you learned how to set the properties of the controls using the properties window.
- In section 2.3, you learned how to set the properties of the controls using code. For example, you learned how to set foreground and background colors using RGB code.

Chapter 3

Writing the Code

- Learning how to write Visual Basic 2010 Code

In the previous chapter, you have learned to design an interface, adding controls and setting control properties. You have also learned how to write some simple code without understanding the concepts behind. In this chapter, you will learn some basic concepts about VB2010 programming and the techniques in writing code .I will keep the theories short so that it would not be too taxing for beginners.

3.1 Understanding Event Driven Programming

VB2010 is an object oriented and event driven programming language. In fact, all windows applications are event driven. Event driven means the user decides what to do with the program, whether he or she wants to click the command button, enter text in a text box, or close the application and more. An event is related to an object, it is an incident that happens to the object due to the action of the user, such as a click or pressing a key on the keyboard. A class contains events as it creates instant of a class or an object. When we start a windows application in VB2010 in previous chapters, we will see a default form with the Form1 appears in the IDE. Form1 is the Form1 Class that inherits from the Form class System.Windows.Forms.Form, as shown in Figure 3.1

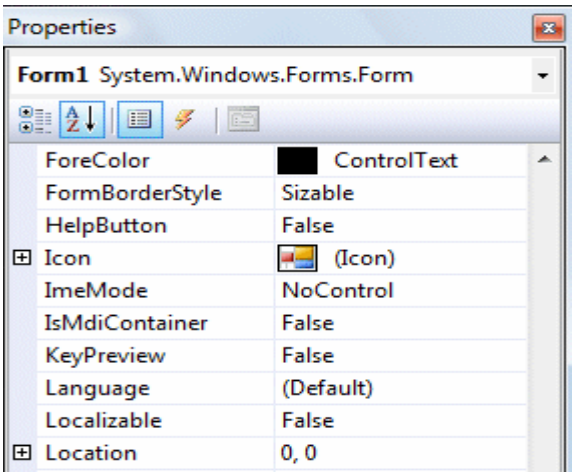


Figure 3.1: The Form1 Class

The other events associated with the Form1 class are click, DoubleClick, DragDrop, Enter and more, as shown in Figure 3.2 below (It appears when you click on the upper right pane of the code window)

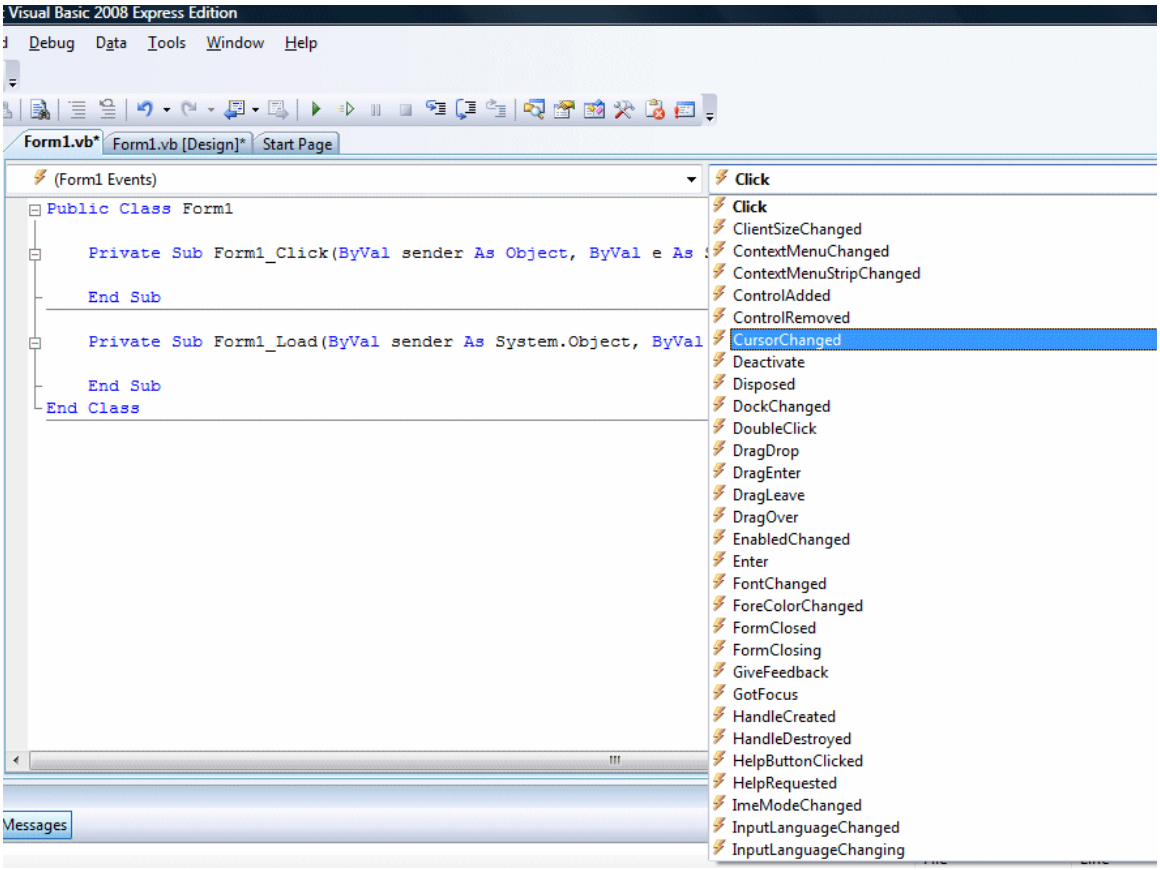


Figure 3.2

3.2 Understanding the Code Structure of an Event Procedure

Now you are ready to write the code for the event procedure so that it will do something more than loading a blank form. The structure of the code takes the following form:

```
Private Sub...

  Statements

End Sub
```

You have to enter the code between Private Sub and End Sub.

```
Private Sub
   Enter your code here
End Sub.
```

There are variations of the structure such as

i)
```
Public Sub
   Enter your code here
End Sub.
```

ii)
```
Sub
   Enter your code here
End Sub.
```

iii)
```
Function
   Enter your code here
```

```
    End Function
```

Let us enter the following code:

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

    Me.Text="My First VB2010 Program"
    Me.ForeColor = Color.Yellow
    Me.BackColor = Color.Blue

End Sub
```

When you press F5 to run the program, the output is shown in Figure 3.3 below:

Figure 3.3: The Output Window

The first line of the code will change the title of the form to “My First VB2010 Program” , the second line will change the foreground object to yellow(in this case, it is a label that you insert into the form and change its name to Foreground) and the last line changes the background to blue color. The equal sign in the code is to assign something to the object, like assigning yellow color to the foreground of the Form1 object (or an instance of Form1). **Me** is the name given

to the Form1 class. We can also call those lines as Statements. Therefore, the actions of the program will depend on the statements entered by the programmer. Here is another example.

```
Private Sub Button1_Click_1(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

    Dim name1, name2, name3 As String
    name1 = "John"
    name2 = "Chan"
    name3 = "Ali"
    MsgBox(" The names are " & name1 & " , " & name2 & " and " & name3)

End Sub
```

In this example, you insert one command button into the form and rename its caption as Show Hidden Names. The keyword Dim is to declare variables name1, name2 and name3 as string, which means they can only handle text. The function MsgBox is to display the names in a message box that are joined together by the "&" signs. The output is shown in Figure 3.4 below:

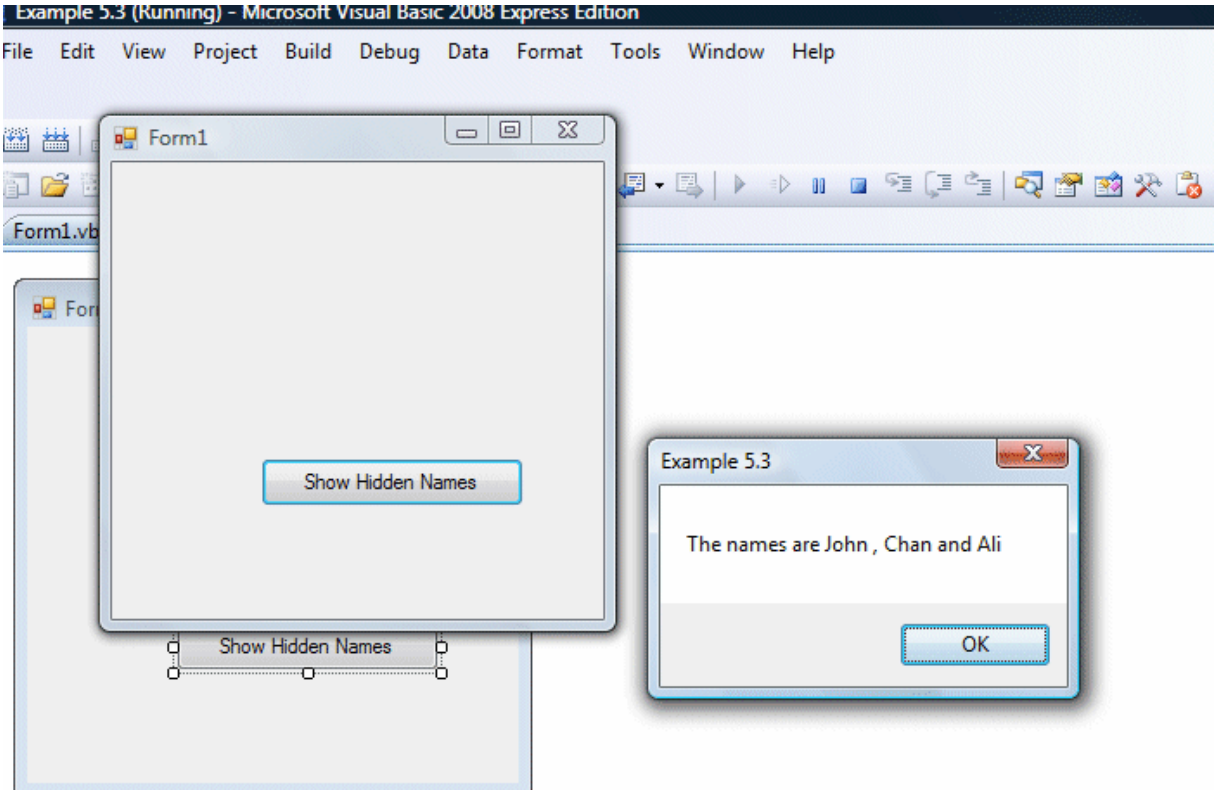


Figure 3.4: The Output Window for Displaying Names

3.3 Writing a Simple Multiplication Program

In this program, you insert two text boxes, three labels and one button. The text boxes are for the user to enter numbers, the label is to display the multiplication operator and the other label is to display the equal sign. The last label is to display the answer. The run time interface is shown in Figure 3.5

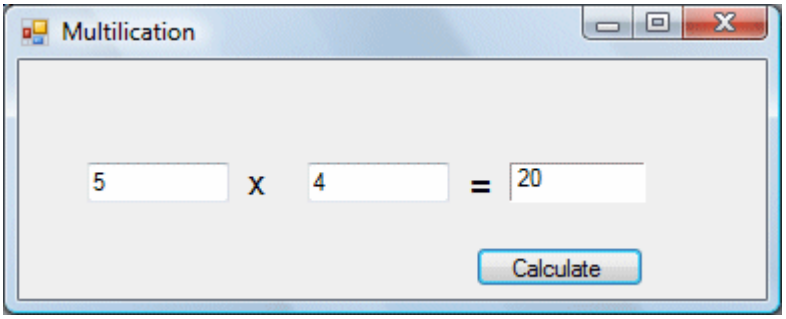


Figure 3.5: The Multiplication Program

The Code

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

    Dim num1, num2, product As Single
    num1 = TextBox1.Text
    num2 = TextBox2.Text
    product = num1 * num2
    Label3.Text = product

End Sub
```

3.4 Writing a Program that Add Items to a List Box

This program will add one item at a time to a list box as the user enters an item into the text box and click the Add button. In this program, you insert a TextBox and a ListBox into the Form. The function of the TextBox is to let the user enter an item one at a time and add it to the Listbox. The method to add an item to the ListBox is Add. The output interface is shown in Figure 3.6.

The Code

```
Class Frm1
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

    Dim item As String
    item = TextBox1.Text

    'To add items to a listbox
    ListBox1.Items.Add(item)

End Sub
End Class
```

Figure 3.6: The Add Items Program

Summary

- In section 3.1, you learned the concept of event driven programming.
- In section 3.2, you learned how to write a simple code for an event procedure, including the usage of MsgBox().
- In section 3.3, you learned how to create a multiplication program.
- In section 3.4, you learned how to write a program to add some items to a list box.