Disclaimer

Visual Basic 2013 Made Easy 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 2013 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 © 2013 Liew Voon Kiong

All rights reserved. No Part of this 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 millions of readers who have visited my Visual Basic Tutorial website at for their support and encouragement.

About the Author

Dr. Liew Voon Kiong holds a bachelor's degree in Mathematics, a master's degree in Management and a doctorate in Business Administration. He has been involved in Visual Basic programming for more than 20 years. He created the popular online Visual Basic Tutorial at www.vbtutor.net, which has attracted millions of visitors since 1996. It has consistently been one of the highest ranked Visual Basic websites.

To provide more support for Visual Basic students, teachers, and hobbyists, Dr. Liew has written this book to complement the free Visual Basic 2013 tutorial with much more content. He is also the author of the Visual Basic Made Easy series, which includes Visual Basic 6 Made Easy, Visual Basic 2008 Made Easy, Visual Basic 2010 Made Easy, Visual Basic 2017 Made Easy and Excel VBA Made Easy. Dr. Liew's books have been used in high school and university computer science courses all over the world.
# Table of Contents

1.1 A Brief Description of Visual Basic 2013 9  
1.2 Visual Studio 2013 Integrated Development Environment 10  
1.3 Creating a New Project in Visual Studio 2013 11  
2.1 Customizing the Form 17  
2.2 Adding Controls to the Form 20  
3.1 The Concept of Event-Driven Programming 25  
3.2 Writing the Code 27  
4.1 Text Box 30  
4.2 Label 32  
4.3 List Box 33  
   4.3.1 Adding Items to a Listbox 33  
   4.3.2 Removing Items from a List Box 37  
4.4 Combo Box 37  
   4.4.1 Adding Items to a Combobox 38  
   4.4.2 Removing Items from a Combobox 40  
5.1 Loading an Image in a Picture Box 42  
   5.1.1 Loading an Image at Design Time 42  
   5.1.2 Loading an Image at Runtime 45  
5.2 Loading an Image in a Picture Box using Open File Dialog Control 46  
6.1 Visual Basic 2013 Data Types 49  
   6.1.1 Numeric Data Types 49  
      Table 6.1: Numeric Data Types 50  
   6.1.2 Non-numeric Data Types 50  
      Table 6.2: Non-numeric Data Types 51  
   6.1.3 Suffixes for Literals 51  
      Table 6.3: Suffixes 51  
6.2 Variables and Constants 52  
   6.2.1 Variable Names 52
6.2.2 Declaring Variables
6.2.3 Assigning Values to Variables
6.2.4 Scope of Declaration
6.2.5 Declaring Constants
7.1 Introduction to Arrays
7.2 Dimension of an Array
7.3 Declaring Arrays
8.1 Mathematical Operators
8.2 Writing Code that Performs Mathematical Operations
9.1 String Manipulation Using + and & signs
9.2 String Manipulation Using Built-in Functions
  9.2(a) Len Function
  9.2(b) Right Function
  9.2(c) Left Function
  9.2(d) Mid Function
  9.2(e) Trim Function
  9.2(f) Ltrim Function
  9.2(g) The Rtrim Function
  9.2(h) The InStr function
  9.2(i) Ucase and the Lcase Functions
  9.2(j) Chr and the Asc functions
10.1 Conditional Operators
10.2 Logical Operators
10.3 Using the If Control Structure with the Comparison Operators
  10.3(a) If....Then Statement
  10.3(b) If....Then...Else Statement
  10.3(c) If....Then...ElseIf Statement
11.1 The Select Case...End Select Structure
11.2 The usage of Select Case
Example 11.1: Examination Grades
20.2 Creating the Graphics Object 141
20.3 Creating the Pen Object 141
20.4 Drawing a Line 142
20.5 Drawing Lines that Connect Multiple Points 144
   Example 20.1 144
20.6 Drawing a curve that Connect Multiple Points 145
   Example 20.2 146
20.7 Drawing Quadratic Curve 147
   Example 20.3 147
20.8 Drawing Sine Curve 149
   Example 20.4 150
20.9 Creating a Rectangle 151
20.10 Customizing Line Style of the Pen Object 153
   Example 20.5 153
20.11 Drawing an Ellipse 155
   Example 20.6 Drawing an Ellipse 156
   Example 20.7 DrawEllipse method 157
20.12 Drawing a Circle 158
   Example 20.8 Drawing a Circle 158
   Example 20.9 158
20.13 Drawing Text 159
   Example 20.10 160
20.14 Drawing Polygons 162
20.15 Drawing a Pie 165
   Example 20.14 : Drawing a pie that sweeps clockwise through 60 degree. 165
20.16 Filling Shapes with Color 166
   20.16(a) Drawing and Filling a Rectangle with Color 167
   Example 20.15 167
   20.16(b) Drawing and Filling an Ellipse with Color 168
   Example 20.16 168
Chapter 1
Introduction to Visual Basic 2013

❖ A brief description of Visual Basic 2013
❖ Getting to know the Visual Basic 2013 Integrated Development Environment

1.1 A Brief Description of Visual Basic 2013

Visual Basic 2013 is the latest version of Visual Basic launched by Microsoft in 2013. Visual Basic 2013 is almost similar to Visual Basic 2012 but it has added some new features. Visual Basic has gone through many phases of development since the days of BASIC that was built for DOS. BASIC stands for Beginners' All-purpose Symbolic Instruction Code. 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 gradually phased out the DOS versions of BASIC and completely replaced them by Visual Basic.

Studio 2013. Microsoft offers free version of Visual Studio 2013 that comes in five Express editions, they are:

- Visual Studio Express 2013 for Web
- Visual Studio Express 2013 for Windows
- Visual Studio Express 2013 for Windows Desktop
- Visual Studio Express 2013 for Windows Phone
- Visual Studio Team Foundation Server Express 2013


1.2 Visual Studio 2013 Integrated Development Environment

To start learning programming in Visual Basic 2013, you need to download and install a copy of Visual Studio Express 2013 for Windows Desktop in your computer. When you launch Visual Studio 2013 Express, you will be presented with the Integrated Development Environment, as shown in Figure 1.1
Figure 1.1: Visual Studio Express 2013 IDE

The Visual Studio 2013 Express default IDE comprises three sections, the start page, the output section and the solution explorer. In the start page, you can either start a new project, open a project or open a recent project. You can also check for the latest news in Visual Studio 2013 Express for Windows Desktop. The IDE also consists of a menu bar and a tool bar where you can perform various tasks by clicking the menu items.

1.3 Creating a New Project in Visual Studio 2013

To start a new Visual Studio Express 2013 project, click on New Project under the Start section to launch the Visual Studio 2013 New Project page as shown in Figure 1.2. You can also choose to open a recent project:
Figure 1.2: Visual Studio 2013 Express New Project Page

The New Project Page comprises three templates, Visual Basic, Visual C# and Visual C++. Since we are going to learn Visual Basic 2013, we shall select Visual Basic. Visual Basic 2013 offers you four types of projects that you can create; they are Windows Forms Application, WPF Application, Console Application and Class Library. Since we are going to learn how to create windows desktop applications, we shall 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, MyFirstProgram. After you have renamed the project, click OK to continue. The Visual Basic Express 2013 IDE Windows will appear, as shown in Figure 1.3. Visual Basic Express 2013 IDE comprises a few windows, the Form window, the Solution Explorer window and the Properties window. It also consists of a toolbox which contains many useful controls that allows a programmer to develop his or her Visual Basic 2013 programs.
Figure 1.3: The Visual Basic 2013 Express IDE

The Toolbox is not shown until you click on the Toolbox tab. When you click on the Toolbox tab or use the shortcut keys Ctrl+w+x, the common controls Toolbox will appear, as shown in Figure 1.4.
Figure 1.4: Visual Basic 2013 Express Tool Box
Now, we shall proceed to show you how to create your first program. First, change the text of the form to My First Visual Basic 2013 Program in the properties window; it will appear as the title of the program. Next, insert a button and change its text to OK. The design interface is shown in Figure 1.5

![Design Interface](image)

**Figure 1.5: The Design Interface**

Now click on the OK button to bring up the code window and enter the following statement between Private Sub and End Sub procedure, as shown in Figure 1.6:

```vbnet
MsgBox("My First Visual Basic 2013 Program")
```

Now click on the Start button on the toolbar or press F5 to run the program then click on the OK button, a dialog box that displays the “My First Visual Basic 2013 Program” message will appear, as shown in Figure 1.7. The function MsgBox is a built-in function of Visual Basic 2013 which can display the text enclosed within the brackets.
Summary

- In section 1.1, you have learned about the evolution of Visual Basic.
- In section 1.2, you have learned how to launch Visual Basic Studio Express 2013 the start page
- In section 1.3, you have learned how to launch the new project dialog write your first VB2013 program.
Chapter 2
Designing the Interface

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

As Visual Basic 2013 is a GUI-based programming language, the first step in developing an application is to build a graphical user interface. To build a graphical user interface, you need to customize the default form by changing its properties at design phase and at runtime. After customizing the default form, then you can proceed to add controls from the toolbox to the form and then customize their properties.

2.1 Customizing the Form

When you start a new Visual Basic 2013 project, the IDE will display the default form along with the Solution Explorer window and the Properties window for the form as shown in Figure 2.1. The name of the default form is Form1. The properties window displays all properties related to Form1 and their corresponding attributes or values. You can change the name of the form, the title of the form, the background colour, the foreground colour, the size and more. Try changing the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>MyForm</td>
</tr>
<tr>
<td>Text</td>
<td>Mt First VB2013 Program</td>
</tr>
<tr>
<td>BackColor</td>
<td>Blue</td>
</tr>
<tr>
<td>MaximizeBox</td>
<td>False</td>
</tr>
</tbody>
</table>
The output interface is shown in Figure 2.2. Notice that the title has been changed from Form1 to My First Program, background changed to blue colour and the window cannot be maximized.

Figure 2.1
You can also change the properties of the form at run-time by writing the relevant codes. The default form is an object and an instant of the form can be denoted by the name `Me`. The property of the object can be defined by specifying the object’s name followed by a dot or period:

```
ObjectName.property
```

For example, we can set the background of the form to blue using the following code:

```
Me.BackColor=Color.Blue
```

To produce the same interface as shown in Figure 2.2, type in the following code by clicking the form to enter the code window:
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As EventArgs) Handles MyBase.Load

Me.Text = "My First VB2010 Program"
Me.BackColor = Color.Blue
Me.MaximizeBox = False

End Sub

Press F5 to run the program and you will get the exact interface as that shown in Figure 2.2

2.2 Adding Controls to the Form

In previous section, we have learned how to build an initial interface in Visual Basic 2013 by customizing the default form. Now we shall continue to build the interface by adding controls to the form. There are numerous controls that we can add to the form. Among the controls, the most common ones are button, label, text box, list box, combo box, picture box, check box, radio and more. The controls can be made visible or invisible at runtime. However, some controls will only run in the background and cannot be seen at runtime, one such control is the timer.

The Toolbox is usually hidden when you start Visual Basic 2013, you need to click View on the menu bar and then select Toolbox to reveal the tool box, as shown in Figure 2.3. You can also use shortcut keys Ctrl+w+x to bring out the tool box.
Figure 2.3: Toolbox

You can position the tool box by dragging it anywhere you like while its status is set to float. You can also dock the toolbox by right-clicking on the tool box and choose dock from the pop-up menu. The docked tool box that appears side by side with the default form is as shown in Figure 2.4.
Figure 2.4

You can also dock the tool box at the bottom, below the default form, as shown in Figure 2.5

Further, you may also pin the tool box to the side bar or the bottom bar by clicking on the pin icon on the menu bar of the tool box.

How and where you want to position your tool box is entirely up to you but I strongly suggest that you place the tool box alongside or at the bottom of the default form so that it is easy for you to add controls from the tool box into the form. You should never cover the form with the tool box because it will be difficult to add controls to the form.
Adding a control to the form is an easy task, what you need to do is double click it or drag it onto the form. You can drag the control around in the form and you can also resize it.

To demonstrate how to add controls and then change their properties, we shall design a picture viewer. First, change the title of the default form to Picture Viewer in its properties window. Next, insert a picture box on the form and change its background colour to white. To do this, right click the picture box and select properties in the popup menu, then look for the BackColor Property as shown in the properties window in Figure 3.4.

Finally, add two buttons to the form and change the text to View and Close in their respective properties windows. Now, we have designed a basic picture viewer. We shall add more features later.

The picture viewer is not functional until we write code to response to events triggered by the user. We will deal with the programming part in the coming lessons.
Figure 2.6

Summary
- In section 2.1, you have learned how to customize the form by changing the values of its properties.
- In section 2.2, you have learned how to add controls to the form and change their properties at design phase and at runtime.

Chapter 3
Writing the Code

❖ Learn the basics of code writing in Visual Basic 2013
In previous chapter, we have learned how to design the user interface by adding controls to the form and by changing their properties. However, the user interface alone will not work without adding code to them. In this chapter, we shall learn how to write code for all the controls so that they can interact with events triggered by the users. Before learning how to write Visual Basic 2013 code, let us dwell into the concept of event-driven programming

3.1 The Concept of Event-Driven Programming

Visual Basic 2013 is an event driven programming language because we need to write code to response to certain events triggered randomly by the user via the controls on the form. These events do not occur in a certain order. The events usually comprises but not limited to the user’s inputs. Some of the common events are load, click, double click, drag and drop, pressing the keys and more.

Every form and every control you place on the form has a set of events related to them. To view the events, double-click the control (object) on the form to enter the code window. The default event will appear at the top part on the right side of the code window. You need to click on the default event to view other events associated with the control. The code appears on the left side is the event procedure associated with the load event. Figure 3.1 illustrates the event procedure load associated with the default form and Figure 3.2 shows the events associated with button.
Figure 3.1: Events associated with Form
3.2 Writing the Code

To start writing code in Visual Basic 2013, click on any part of the form to go into the code window as shown in Figure 3.1. This is the structure of an event procedure. In this case, the event procedure is to load Form1 and it starts with Private Sub and end with End Sub. This procedure includes the Form1 class and the event Load, and they are bind together with an underscore, i.e. Form_Load. It does nothing other than loading an empty form. To make the load event does something, insert the statement

```
MsgBox “Welcome to Visual Basic 2013”
```
The Code

```
Public Class Form1
    Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        MsgBox ( "My First Visual Basic 2013 Program", ,"My Message")
    End Sub
End Class
```

MsgBox is a built-in function in Visual Basic 2013 that displays a message in a pop-up message box. The MsgBox function comprises a few arguments, the first being the message that is displayed and the third one is the title of the message box. When you run the program, a message box displaying the text “My First Visual Basic 2013 Program” will appear, as shown in Figure 4.3. Notice that its title is My Message.

![Message Box Example](image)

**Figure 3.3**

You will notice that above Private Sub structure there is a preceding keyword Public Class Form1. This is the concept of an object oriented programming language. When we start a windows application in Visual Basic 2013, we will see a default form with the name Form1 appears in the IDE, it is actually the Form1 Class
that inherits from the Form class System.Windows.Forms.Form. A class has events as it creates an instant of a class or an object.

You can also write code to perform arithmetic calculation. For example, you can use the MsgBox and the arithmetic operator plus to perform addition of two numbers, as shown below:

```csharp
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    MsgBox("2" & "+" & "5" & ";" & 2 + 5)
End Sub
```

*The symbol & (ampersand) is to perform string concatenation.

The output is as shown in Figure 3.4

![Figure 3.4](image)

**Summary**
- In section 3.1, you have learned the concepts of event driven programming
- In section 3.2, you have learned how to write code for the controls
Chapter 4
Working with Controls

❖ Learn to work with a text box
❖ Learn to work with a label control
❖ Learn to work with a list box
❖ Learn to work with a combo box

In the preceding chapter, we have learned how to write simple Visual Basic 2013 code. In this lesson, we will learn how to work with some common controls and write codes for them. Some of the commonly used controls are label, textbox, button, list box and combo box. However, in this chapter, we shall only deal with text box, label, list box and combo box. We shall deal with other controls later.

4.1 Text Box

Text box is the standard control for accepting input from the user as well as to display the output. It can handle string (text) and numeric data but not images or pictures. String in a text box can be converted to a numeric data by using the function Val(text). The following example illustrates a simple program that processes the input from the user.

Example 4.1

In this program, you add two text boxes and a button on the form. The two text boxes are used to accept inputs from the user. Besides, a button is also programmed to calculate the sum of the two numbers using the plus operator. The value enter into a text box is stored using the syntax Textbox1.Text, where Text is one of the properties of text box.

The following program will add the value in textbox 1 and value in textbox 2 and output the sum in a message box. The runtime interface is shown in Figure 4.1
Private Sub Button1_Click(sender As Object, e As EventArgs)
Handles Button1.Click
    MsgBox("The sum is" & Val(TextBox1.Text) + Val(TextBox2.Text))
End Sub

Figure 4.1

Clicking Add button produces the answer in a message box, as shown in Figure 4.2:

Figure 4.2
4.2 Label

Label is a very useful control for Visual Basic, as it is not only used to provide instructions and guides to the users, it can also be used to display outputs. It is different from text box because it can only display static text, which means the user cannot change the text. Using the syntax Label.Text, it can display text and numeric data. You can change its text in the properties window and also at runtime.

Example 4.2

Based on Example 4.1, you now add two labels, one is to display the text Sum= and the other label is to display the answer of the Sum. For the first label, change the text property of the label by typing Sum= over the default text Label1. Further, change its font to bold and font size to 10. For the second label, delete the default text Label2 and change its font to bold and font size to 10. Besides that, change its background colour to white.

In this program, instead of showing the sum in a message box, we wish to display the sum on the label.

The Code

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

Label2.Text = Val(TextBox1.Text) + Val(TextBox2.Text)

End Sub
```

*The function Val is to convert text to numeric value. Without using Val, you will see that two numbers are joined together without adding them.*
The function of the List Box is to present a list of items where the user can click and select the items from the list. Items can be added at design time and at runtime. The items can also be removed at design time and also at runtime.

### 4.3.1 Adding Items to a Listbox

To demonstrate how to add items at design time, start a new project and insert a list box on the form then right-click on the list box to access the properties window. Next, click on collection of the Item property, you will be presented with the String Collection Editor whereby you can enter the items one by one by typing the text and
press the Enter key, as shown in Figure 4.4. After clicking on the OK button, the items will be displayed in the text box, as shown in Figure 4.5.

![String Collection Editor](image)

**Figure 4.4**
Figure 4.5

Items can also be added at runtime using the Add( ) method. Before we proceed further, we need to know that Visual Basic 2013 is a full-fledged object oriented programming language. Therefore, visual basic 2013 comprises objects. All objects have methods and properties, and they can are differentiated and connected by hierarchy. For a listbox, Item is an object subordinated to the object ListBox. Item comprises a method call Add() that is used to add items to the list box. To add an item to a list box, you can use the following syntax:

```vbnet
ListBox.Item.Add("Text")
```

For example, you can key-in the following statement

```vbnet
Private Sub Button1_Click(sender As Object, e As EventArgs)
Handles Button1.Click
ListBox1.Items.Add("Nokia")
End Sub
```
The item “Nokia” will be added to the end of the list, as shown in Figure 4.6

![List Box Image]

**Figure 4.6**

You can also allow the user to add items using the InputBox function, as follows:

```vbnet
Private Sub Button1_Click(sender As Object, e As EventArgs)
Handles Button1.Click
Dim myitem
myitem = InputBox("Enter your Item")
ListBox1.Items.Add(myitem)
End Sub
```

* The keyword Dim is to declare the variable myitem. You will learn more about variables in coming lessons