Visual Basic 2019 Handbook

A Concise Guide to VB2019 Programming

By Dr.Liew

Disclaimer

Visual Basic 2019 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 2019 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.

Acknowledgement

I would like to express my sincere gratitude to the 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 help in editing 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 **vbtutor.net** 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 2019 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 2013 Made Easy, Visual Basic 2015 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.

Contents

Chapter 1 Introduction to Visual Basic 2019

1.1 A Brief History of Visual Basic

Visual Basic is a third-generation event-driven programming language first released by Microsoft in 1991. The final version of the classic Visual Basic was Visual Basic 6. Visual Basic 6 is a user-friendly programming language designed for beginners. Therefore, it enables anyone to develop GUI Windows applications easily. Many developers still favor VB6 over its successor VB.NET.

In 2002, Microsoft released Visual Basic.NET (VB.NET) to replace Visual Basic 6. Thereafter, Microsoft declared VB6 a legacy programming language in 2008. However, Microsoft still provides some form of support for VB6. VB.NET is a fully object-oriented programming language implemented in the .NET Framework. It was created to cater for the development of the web as well as mobile applications. Subsequently, Microsoft has released many versions of VB.NET. They are Visual Basic 2005, Visual Basic 2008, Visual Basic 2010, Visual Basic 2012, Visual Basic 2013, Visual Basic 2015, Visual Basic 2017 and Visual Basic 2019. Although the .NET portion was discarded in 2005, all versions of the Visual Basic programming language released since 2002 are regarded as the VB.NET programming language

Microsoft has released Visual Studio 2019 in early 2019. VS 2019 allows you to code in different programming languages and different platforms, Visual Basic 2019 is one of them. The other Programming languages are C#, C++, F#, JavaScript, Java and Python. Visual Basic 2019 is the latest version VB.NET programming language released by Microsoft.

Learn more about Visual Studio 2019 from the Youtube link below:

https://youtu.be/n5sJ4EewKGk

1.2 Installation of Visual Studio 2019

You can download the free version of Visual Studio 2019 from the following link:

https://visualstudio.microsoft.com/vs/

Clicking the link brings up the Visual Studio 2019 download page, as shown below:

Figure 1.1

You can choose the free Visual Studio Community 2019 or the Full-featured Professional 2019 and End-to-End solution Enterprise 2019 to download. The free version that provides full-featured IDE for students, open source community and individuals. As this book was written based on the free version, proceed to download the free Visual Studio 2019 Community, select community, and download the installer file. The downloaded installer file will appear on your Windows 10 taskbar. Click it to install Visual Studio 2019. Clicking the Visual Studio 2019 Installer will start downloading, unpacking, and installing the files necessary for the installation of Visual Studio 2019, as shown in Figure 1.2

Figure 1.2

You will see several status screens that show the progress of the installation. After the installer has finished installing, it is time to pick the feature set that you wish to install, as shown in Figure 1.3. Since we are focusing on developing Visual Basic 2019 desktop app, we will select the .NET desktop development component. After making your selections, click install.

Figure 1.3

Upon completion of the installation, you are now ready to launch Visual Studio 2019 and start programming in Visual Basic 2019

1.3 Creating a Visual Basic 2019 Project

Launching Microsoft Visual Studio 2019 will bring you to the Visual Studio 2019 Start Page, as shown in Figure 1.4

Figure 1.4 Visual Studio 2019 Start Page

The Visual Studio 2019 start page comprises two sections, the Open Recent section, and the Get Started section. In the start page, you can select a recent project file or choose any

option in the Get Started section. You can choose to clone a project from GitHub or Azure DevOps, open a project or solution, open a local folder, create a new project, or continue without code.

Let us create a new project by clicking on the Create a new project option. You will now see the Create a new project template page, as shown in Figure 1.5. In the Create a new project page, select the Visual Basic language.

Figure 1.5 Create a new project template

Next, select the Windows Forms App (.Net Framework) template as we want to develop a Windows desktop project, as shown in Figure 1.6

Figure 1.6 Create a new project template

Upon clicking the selected project template, the project configuration page appears, as shown in Figure 1.7. You can configure your project by typing the project name and select a few other options.

Figure 1.7 Configuring Project

At the bottom of this dialog box, you can change the default project name WindowsApplication1 to some other name you like, for example, My First Visual Basic 2019 App. After renaming the project, click OK to continue. The Visual Basic 2019 IDE Windows will appear, as shown in Figure 1.8. Visual Basic 2019 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 the programmer to develop his or her Visual Basic 2019 programs.

Figure 1.8 The Visual Basic 2019 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+Alt+x, the common controls Toolbox will appear, as shown in Figure 1.9. You can drag and move your toolbox around and dock it to the right, left, top or bottom of the IDE.

Figure 1.9 Visual Basic 2019 Toolbox

Next, we shall proceed to show you how to create your first VB2019 application. First, change the text of the form to ‘My First VB 2019 App’ in the properties window; it will appear as the title of the application. Next, insert a button and change its text to OK. The design interface is shown in Figure 1.10

Figure 1.10 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 1.11.

```
MsgBox("My First Visual Basic 2019 App")
```

Clicking the Start button on the toolbar or press F5 to run the application will launch the runtime interface, as shown in Figure 1.11. Executing the application by clicking on the OK button will bring up a dialog box that displays the “My First Visual Basic 2019 App″ message, as shown in Figure 1.12. The function **MsgBox** is a built-in function of Visual Basic 2019 which can display the text enclosed within the brackets.

Figure 1.11 Visual Basic 2019 Code Window

Figure 1.12 The Runtime Interface

Figure 1.13 The Message Box

Summary

- In section 1.1, you have learned about the history of Visual Basic 2019
- In section 1.2, you have learned how to install and launch Visual Basic Studio 2019
- In section 1.3, you have learned how to launch the new project dialog and the Visual Basic 2019 IDE. You have also learned how to write your first program.

Chapter 2 Designing the User Interface

As Visual Basic 2019 is a GUI-based programming language, the first step in developing an application is to design the user interface (UI). To build a graphical user interface, first of all you need to customize the default form by changing its properties at design phase and at runtime, including its name, title, background color and so forth. After customizing the default form, you may 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 2019 project, the VB2019 IDE will display the default form along with the Solution Explorer window and the Properties window, as shown in Figure 2.1. The name of the default form is Form1. The properties window displays all the properties related to Form1 and their corresponding attributes or values. You can change the name of the form, the title of the form using the text property, the background color, the foreground color, size and more. Try changing the following properties:

Property	Value
Name	MyForm
Text	My First VB2019 App
BackColor	LavenderBlush
ForeColor	Crimson
MaximizeBox	False

In fact, you do not have to type in the color manually, you can indeed select a color from the color drop-down list that comprises three tabs, Custom, Web, and System, as shown in Figure 2.1. Clicking on the drop-down arrow will bring out a color palette or a list of color rectangles where you can select a color.

Figure 2.1

Another method of setting the colors is to manually type in the RGB color code or the hex color code. The values of R, G and B ranges from 0 to 255, therefore, by varying the values of the RGB we can obtain different colors. For example, an RGB value of 128, 255, 255 yield the cyan color.

On the other hand, the hex color code system uses a six-digit, three-byte hexadecimal number to represent colors. The bytes represent the red, green and blue components of the color. One byte represents a number ranging from 00 to FF (in hexadecimal notation), or 0 to 255 in decimal notation. For example, **#0000ff** represents the cyan color. However, when you type the Hex color code in the properties window of VS2019, it automatically converts the color to RGB color or the color name. Figure 2.2 shows a list of Hex color codes and the corresponding colors.

Figure 2.2 Hex Color Codes

The design interface is shown in Figure 2.2 and the runtime interface is shown in Figure 2.4. In the runtime interface, notice that the title has been changed from Form1 to My First Visual Basic 2019 App, background color changed to LavenderBlush , the text OK color is Crimson and the window cannot be maximized.

Figure 2.3 Design UI

Figure 2.4 Runtime UI

You can also change the properties of the form at runtime 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
```

In addition, you can also use the **FromArgb** method to specify the color using the RGB codes, as follows:

```
Me.BackColor = Color.FromArgb(0, 255, 0)
```

Now, type in the following code by clicking the form to enter the code window:

```
Private Sub Form1_Load(sender As Object, e As EventArgs)_
 Handles_ MyBase.Load
        Me.Text = "My First Visual Basic 2019 Application"
        Me.BackColor = Color.Turquoise
         Me.ForeColor = Color.Ivory
        MyBtn.BackColor = Color.DodgerBlue
        Me.MaximizeBox = False
        Me.MinimizeBox = True
End Sub
```

To runtime UI is shown in Figure 2.5. Notice that is is now different from that shown in Figure 2.4,

Figure 2.5

In place of Turquoise, you can use RGB code as follows:

```
Me.BackColor = Color.FromArgb(64,224,208)
```

In addition, you can also specify the size, the opacity and the position of the default form using the code, as follows:

```
Private Sub Form1_Load(sender As Object, e As EventArgs_
Handles MyBase.Load
Me.Text = "My First VB2019 App"
Me.BackColor =Color.Beige
Me.MaximizeBox = False
Me.MinimizeBox = True
Me.Size = New Size(400, 400)
Me.Opacity = 0.85
Me.CenterToParent()
End Sub
```

The property Opacity sets the degree of transparency. The runtime interface is as shown in Figure 2.6

Figure 2.6

2.2 Adding Controls to the Form

In section 2.1, we have learned how to build an initial UI in Visual Basic 2019 by customizing the default form. Next, we shall continue to build the UI by adding some controls to the form. The controls are objects that consist of three elements, namely properties, methods, and events. They can be added to the form from the Toolbox. Among the controls, the most common ones are the button, label, textbox, listbox, combobox, picture box, checkbox, radio button and more. These controls can be made visible or invisible at runtime. However, some controls will only run in the background and never be seen at runtime, one such control is the timer.

The Toolbox is usually hidden when you start Visual Basic 2019 IDE, you need to click View on the menu bar and then select Toolbox to reveal the tool box, as shown in Figure 2.6. You can also use shortcut keys Ctrl+w+x to bring out the toolbox.

Figure 2.6: Toolbox

You can position the Toolbox by dragging it anywhere you like while its status is set to float. You can also dock the toolbox by right-clicking on the Toolbox and choose dock from the pop-up menu. The docked Toolbox that appears side by side with the Solution Explorer, and as one of the tabbed windows together with the Form Design window and the code window, as shown in Figure 2.7.

Figure 2.7 Toolbox

You can also dock the Toolbox at the bottom, below the default form, as shown in Figure 2.8. Further, you may also pin the Toolbox to the side bar or the bottom bar by clicking on the pin icon on the menu bar of the toolbox.

How and where you want to position your tool box is entirely up to you but we 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 Toolbox because it will be difficult to add controls to the form.

Figure 2.8

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 the form, and you can also resize it.

To demonstrate how to add the 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 color 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 2.9. Finally, add two buttons to the form and change the text to View and Close in their respective properties windows. The picture viewer is not functional yet until we write code for responding to events triggered by the user. We will deal with the programming part in the coming chapters.

Figure 2.9

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

In the 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 the events triggered by the users. Before learning how to write Visual Basic 2019 code, let us delve into the concept of event-driven programming

3.1 The Concept of Event-Driven Programming

Visual Basic 2019 is an event-driven programming language meaning that the code is executed in response to events triggered by the user like clicking the mouse, pressing a key on the keyboard, selecting an item from a drop-down list, typing some words into text box and more. It may also be an event that runs in response to some other program. Some of the common events in Visual Basic 2019 are load, click, double-click, drag-drop, keypress 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 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 Form (its name has been changed to PicViewer therefore you can see the words PicViewer events) and Figure 3.2 shows the events associated with button.

Figure 3.1: Events associated with Form

Figure 3.2: Events associated with the button

3.2 Writing the Code

To start writing code in Visual Basic 2019, click on any part of the form to go into the code window as shown in Figure 3.1. The event procedure is to load Form1 and it starts with the keywords **Private Sub** and ends with **End Sub**. This procedure includes the Form1 class and the event Load, and they are bound 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 2019″)
```

The Code

```
Public Class Form1

Private Sub Form1_Load(sender As Object, e As EventArgs)_

Handles_ MyBase.Load

MsgBox (“My First Visual Basic 2019 APP", ,"My Message")

End Sub

End Class
```

MsgBox is a built-in function in Visual Basic 2019 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 2019 APP” will appear, as shown in Figure 3.3.

Figure 3.3

You will notice that above the Private Sub structure there is a preceding keyword **Public Class Form1**. This is the concept of the object-oriented programming language. When we start a windows application in Visual Basic 2019, 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 calculations. For example, you can use the **MsgBox** and the arithmetic operator plus to perform addition of two numbers, as shown below:

```
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

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

In the preceding chapter, we have learned how to write simple Visual Basic 2019 code. In this lesson, we shall learn how to work with some common controls and write codes for them. Some of the commonly used controls are Label, TextBox, Button, ListBox and ComboBox. However, in this chapter, we shall only deal with TextBox , Label, ListBox and ComboBox. We shall deal with the other controls later.

4.1 TextBox

TextBox is the standard control for accepting inputs 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 TextBox 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 for accepting inputs from the user. Besides that, we can also program a button to calculate the sum of the two numbers using the plus operator. The value entered into a TextBox is stored using the syntax **TextBox1.Text** , where Text is one of the properties of TextBox.

The following code will add the value in TextBox1 and the value in TextBox2 and displays the sum in a message box. The runtime interface is illustrated 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

After clicking the Add button, you will obtain the answer in a message box, as shown in Figure 4.2.

Figure 4.2

4.2 Label

Label is an especially useful control for Visual Basic 2019 because we can use it for multiple purposes like providing instructions and guides to the users, displaying outputs and more. It is different from the TextBox because it is read only, which means the user cannot change or edit its content at runtime. Using the syntax **Label.Text**, it can display string as well as numeric data . You can change its text property in the properties window or at runtime by writing an appropriate code.

Example 4.2

Based on Example 4.1, we add two Labels, one is for displaying 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 its font size to 10. For the second label, delete the default text Label2 and change its font to bold and the font size to 10. Besides that, change its background color 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

```
Private Sub Button1_Click(sender As Object, e As EventArgs)_
 Handles Button1.Click
   LblSum.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 without adding them.

The output is as shown in Figure 4.3

Figure 4.3

4.3 ListBox

The function of the ListBox is to display a list of items where the user can click and select the items from the list. Items can be added by the programmer at design time or at runtime using a code. We can also write code to allow the user to add items to the ListBox or remove the items from it.

4.3.1 Adding Items to a Listbox

a) Adding items using the String Collection Editor

To demonstrate how to add items at design time, start a new project and insert a ListBox on the form then right-click on the ListBox 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

Figure 4.4

Figure 4.5

b) Adding Items using the Add() Method

Items can also be added at runtime using the **Add()** method. Before we proceed further, we should know that Visual Basic 2019 is an object oriented programming language. Therefore, visual basic 2019 comprises objects. All objects have methods and properties, and they can be differentiated and connected by a hierarchy. For a ListBox, an Item is an object subordinated to the object ListBox. The Item object comprises a method call **Add()** that is used to add items to the ListBox. To add an item to a ListBox, you can use the following syntax:

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

Example 4.3

In this example, running the program will add the item “Vivo” to the end of the list, as shown in Figure 4.6

```
Private Sub BtnAdd_Click(sender As Object, e As EventArgs)_
 Handles BtnAdd.Click
        MyListBox.Items.Add("Vivo")
End Sub
```

Figure 4.6

Example 4.4

In this example, you can allow the user to add items via a popup input box. First, we create a variable `myitem` and then assign a value to `myitem` via the InputBox function that store the input from the user. We then use the `Add()` method to add the user’s item into the Listbox. The code is as follows:

```
Private Sub Button1_Click(sender As Object, e As EventArgs)_
 Handles Button1.Click
Dim myitem                    'declare the variable myitem
 myitem = InputBox("Enter your Item")
MyListBox.Items.Add(myitem)
```

```
End Sub
```

The runtime interface is as shown in Figure 4.7

Figure 4.7

After typing the item ‘Vivo” in the input box, the item will be added to the Listbox, as shown in Figure 4.8.

.

Figure 4.8

Example 4.5 Geometric Progression

This is a Visual Basic program that generates a geometric progression and displays the results in a Listbox. Geometric progression is a sequence of numbers where each subsequent number is found by multiplying the previous number with a fixed number. This fixed number is called the common ratio. The common ratio can be a negative number, an integer, a fraction, and any number but it must not be a zero.

The formula to find the nth term of the geometric progression is $\mathbf{ar^{n-1}}$, where a is the first number and r is the common ratio.

In visual basic 207, we employ the **Do...Loop Until** statement to generate the numbers in a geometric progression. In this program, we need to insert three text boxes for the user to enter the first number, the common ratio and the number of terms. We also need to insert a Listbox to list the generated numbers. Besides that, a command button is needed for the user to generate the numbers in the geometric progression. In addition, we also add another button for clearing the list of generated numbers.

To add the numbers to the list box, we use the **Add()** method. The syntax is **ListBox1.Items.Add(x)**, where x can be any variable.

The code

```
Private Sub BtnComp_Click(sender As Object, e As EventArgs)_
 Handles BtnComp.Click
        Dim x, n, num As Double
    Dim r As Double
    x = TxtFirst.Text
       r = TxtCR.Text
       num = TxtN.Text
       MyList.Items.Add("n" & vbTab & "x")
          MyList.Items.Add("____________")

        n = 1
    Do
        x = x * r
       MyList.Items.Add(n & vbTab & x)
       n = n + 1
       Loop Until n = num + 1
End Sub

Private Sub BtnClr_Click(sender As Object, e As EventArgs)_
Handles BtnClr.Click
       MyList.Items.Clear()
End Sub
```

The output is as shown in Figure 4.9

Figure 4.9 The runtime interface

4.3.2 Removing Items from a List Box

To remove items at design time, simply open the String Collection Editor and delete the items line by line or all at once using the Delete key. To remove the items at runtime, you can use the Remove method, as illustrated in the following Example 4.5.

Example 4.5

In this example, add a button and label it “Remove Items”. Click on this button and enter the following code

```
Private Sub Button2_Click(sender As Object, e As EventArgs)_
 Handles Button2.Click
MyListBox.Items.Remove("Iphone")
End Sub
```

The item “Iphone” will be removed after running the program, as shown in Figure 4.10

Figure 4.10

Example 4.6

You can also allow the user to choose an item to delete via an InputBox. To add this capability, insert a button at design time and change its text to Delete Item. Click on the button and enter the following statements in the code window:

```
Private Sub BtnDelete_Click(sender As Object, e As EventArgs)_
 Handles BtnDelete.Click
        Dim myitem
         myitem = InputBox("Enter your Item for Deletion")
          MyListBox.Items.Remove(myitem)
End Sub
```

The runtime interface is as shown in Figure 4.10. After entering the item Ipone in the input box and press OK, the item Iphone will be deleted from the listbox.

Figure 4.11

To remove a selected item from the Listbox, using the following syntax:

```
Listbox1.Items.Remove(ListBox1.SelectedItem)
```

Example 4.7

```
Private Sub BtnDelSel_Click(sender As Object, e As EventArgs)_
 Handles BtnDelSel.Click
        MyListBox.Items.Remove(MyListBox.SelectedItem)
End Sub
```

When the user run the program and select an item to delete, the item will be deleted, as shown in Figure 4.12 and Figure 4.13

Figure 4.12

Figure 4.13

To remove multiple selected items from the Listbox, you need to use the If...End If structure together with the For...Next loop. Besides that, you also need to ensure that the Listbox allows multiple selection. To enable multiple selection, set the selection mode to MultipleSimple in the Listbox properties windows. The code is as shown in Example 4.7.

Example 4.8

In this example, add an extra button to the previous example and label it as Clear Selected Items. Key in the following code:

```
Private Sub BtnDelSelected_Click(sender As Object, e As EventArgs)_
 Handles BtnDelSelected.Click
        If MyListBox.SelectedItems.Count > 0 Then
        For n As Integer = MyListBox.SelectedItems.Count_
- 1 To 0 Step -1
                'remove the current selected item from items
                MyListBox.Items.Remove(MyListBox.SelectedItems(n))
        Next n
         End If
        End Sub
```

To clear all the items at once, use the clear method, as illustrated in Example 4.8.

Example 4.9

In this example, add a button and label it “Clear the List”

```
Private Sub BtnClr_Click(sender As Object, e As EventArgs)_
 Handles BtnClr.Click
        MyListBox.Items.Clear()
End Sub
```

When you run the program and click the “Clear the List” button, all the items will be cleared. The complete design interface for remove the items from the Listbox is as shown in Figure 4.14

Figure 4.14

4.4 ComboBox

In Visual Basic 2019, the function of the ComboBox is also to present a list of items where the user can click and select the items from the list. However, the ComboBox only display one item at runtime and the user needs to click on the handle (small arrowhead) on the right of the combobox to see all the items that are presented in a drop-down list.

4.4.1 Adding Items to a ComboBox

In order to add items to the combobox at design time, you can also use the String Collection Editor as shown in Figure 4.15. Besides that, if you want to display an item as the default text in the combobox when you run the program, enter the name of the item by replacing the text property of the combobox.

Figure 4.15

After clicking the handle of the right side of the combo box, the user will be able to view all the items, as shown in Figure 4.16.

Figure 4.16

Besides that, you may add items using the **Add ()** method. For example, if you wish to add an item to the ComboBox, you can key in the following statement. The output is as shown in Figure 4.17

```
Private Sub Button1_Click(sender As Object, e As EventArgs)_
 Handles Button1.Click
MyComboBox.Items.Add("Vivo")
End Sub
```

Figure 4.17

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

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

The runtime interface is as shown in Figure 4.18

Figure 4.18

After you type the item 'Xiaomi' and click Ok, you can see that the item has been added to the combobox, as shown in Figure 4.19.

Figure 4.19

4.4.2 Removing Items from a Combobox

To remove items from the combobox at design stage, simply open the String Collection Editor and delete the items line by line or all at once using the Delete key.

To remove the items at runtime, you can use the Remove method, as illustrated in the following example. In this example, add a second button and label it "Remove Items". Click on this button and enter the following code:

```
Private Sub Button2_Click(sender As Object, e As EventArgs)_
 Handles Button2.Click
MyComboBox.Items.Remove("Iphone")
End Sub
```

The item “Ipad” will be removed after running the program. You can also let the user select a certain item to delete, the code is as follows:

```
Private Sub Button1_Click(sender As Object, e As EventArgs)_
Handles Button1.Click
MyComboBox.Items.Remove(MyComboBox.SelectedItem)
End Sub
```

To clear all the items at once, use the clear method, as illustrated in the following example. In this example, add a button and label it “Clear Items”

```
Private Sub Button3_Click(sender As Object, e As EventArgs)_
Handles Button2.Click
MyComboBox.Items.Clear()
End Sub
```

Summary

- In section 4.1, you have learned how to work with a text box
- In section 4.2, you have learned how to work with a label
- In section 4.3.1, you have learned how to add items to a listbox
- In section 4.3.2, you have learned how to remove items from a list box

- In section 4.4.1, you have learned how to add items to a combobox

In section 4.4.2, you have learned how to remove items from a combobox