

# **Visual Basic 2017**

## **Made Easy**

**Dr.Liew**

## **Disclaimer**

Visual Basic 2017 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 2017 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 © 2017 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 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](http://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 2017 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** 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

---

<b>1.1 A Brief Description of Visual Basic 2017</b>	<b>11</b>
<b>1.2 The Visual Studio 2017 Integrated Development Environment</b>	<b>13</b>
<b>1.3 Creating a New Project in Visual Studio 2017</b>	<b>14</b>
<b>2.1 Customizing the Form</b>	<b>20</b>
<b>2.2 Adding Controls to the Form</b>	<b>25</b>
<b>3.1 The Concept of Event-Driven Programming</b>	<b>30</b>
<b>3.2 Writing the Code</b>	<b>32</b>
<b>4.1 TextBox</b>	<b>35</b>
Example 4.1	35
<b>4.2 Label</b>	<b>37</b>
Example 4.2	37
<b>4.3 ListBox</b>	<b>38</b>
4.3.1 Adding Items to a Listbox	38
Adding items using the String Collection Editor	39
b) Adding Items using the Add() Method	40
Example 4.3	40
Example 4.4	41
Example 4.5 Geometric Progression	42
Figure 4.9 The runtime interface	44
4.3.2 Removing Items from a List Box	44
Example 4.5	44
Example 4.6	45
Example 4.6	46
Example 4.7	46
Example 4.8	47
<b>4.4 ComboBox</b>	<b>47</b>
4.4.1 Adding Items to a ComboBox	48

4.4.2 Removing Items from a Combobox	51
<b>5.1 Loading an Image in a Picture Box</b>	<b>53</b>
5.1.1 Loading an Image at Design Time	53
5.1.2 Loading an Image at Runtime	56
<b>5.2 Loading an Image in a Picture Box using Open File Dialog Control</b>	<b>57</b>
<b>6.1 Visual Basic 2017 Data Types</b>	<b>60</b>
6.1.1 Numeric Data Types	60
Table 6.1: Numeric Data Types	61
6.1.2 Non-numeric Data Types	61
Table 6.2: Non-numeric Data Types	62
6.1.3 Suffixes for Literals	62
Table 6.3: Non-numeric Data Types	62
<b>6.2 Variables and Constants</b>	<b>63</b>
6.2.1 Variable Names	63
6.2.2 Declaring Variables	64
Example 6.3	65
6.2.3 Assigning Values to Variables	66
Example 6.4	66
6.2.4 Scope of Declaration	67
6.2.5 Declaring Constants	68
<b>7.1 Introduction to Arrays</b>	<b>70</b>
<b>7.2 Dimension of an Array</b>	<b>70</b>
<b>7.3 Declaring Arrays</b>	<b>71</b>
Example 7.1	72
Example 7.2	72
Example 7.3	73
Example 7.4	73
<b>8.1 Mathematical Operators</b>	<b>76</b>
<b>8.2 Writing Code that Performs Mathematical Operations</b>	<b>77</b>
Example 8.1 Standard Arithmetic Calculations	77

Example 8.2 Pythagorean Theorem	78
Example 8.3: BMI Calculator	78
<b>9.1 String Manipulation Using + and &amp; signs</b>	<b>80</b>
Example 9.1	80
<b>9.2 String Manipulation Using Built-in Functions</b>	<b>83</b>
9.2 (a) Len Function	83
Example 9.3	83
9.2(b) Right Function	84
Example 9.4	84
9.2(c) Left Function	85
9.2 (d) Mid Function	85
Example 9.5	85
Example 9.6	86
9.2(e) Trim Function	87
Example 9.7	87
9.2(f) Ltrim Function	88
9.2(g)The Rtrim Function	88
9.2(h) The InStr function	88
9.2(i) Ucase and the Lcase Functions	88
9.2(j) Chr and the Asc functions	89
<b>10.1 Conditional Operators</b>	<b>90</b>
<b>10.2 Logical Operators</b>	<b>91</b>
<b>10.3 Using If ...Then...Else</b>	<b>92</b>
10.3(a) If....Then Statement	92
Example 10.1	92
10.3(b) If...Then...Else Statement	92
Example 10.2	93
Example 10.3	94
10.3(c) If....Then...Elseif Statement	96
Example 10.4 Grade Generator	97

<b>11.1 The Select Case...End Select Structure</b>	<b>100</b>
<b>11.2 The usage of Select Case is shown in the following examples</b>	<b>101</b>
Example 11.1: Examination Grades	101
Example 11.2	102
The output is shown in Figure 11.2	103
Example 11.3	103
Example 11.4	104
<b>12.1 For....Next Loop</b>	<b>106</b>
Example 12.1 a	107
Example 12.1b	107
Example 12.1c	107
Example 12.1d	108
<b>12.2 Do Loop</b>	<b>108</b>
Example 12.2(a)	109
<b>12.3 While....End While Loop</b>	<b>111</b>
Example 12.3	111
<b>13.1 What is a Sub Procedure</b>	<b>112</b>
<b>13.2 Examples of Sub Procedure</b>	<b>112</b>
Example 13.1	112
Example 13.2: Password Cracker	113
<b>14.1 Creating User-Defined Functions</b>	<b>117</b>
Example 14.1: BMI Calculator	118
Example 14.2: Future Value Calculator	119
<b>14.2 Passing Arguments by Value and by Reference</b>	<b>121</b>
<b>15.1 The Abs Function</b>	<b>125</b>
Example 15.1	125
<b>15.2 The Exp function</b>	<b>126</b>
Example 15.2	126
<b>15.3 The Fix Function</b>	<b>127</b>
Example 15.3	128

<b>15.4 The Int Function</b>	<b>128</b>
<b>15.5 The Log Function</b>	<b>129</b>
Example 15.4	129
<b>15.6 The Rnd( ) Function</b>	<b>130</b>
Example 15.5	130
<b>15.7 The Round Function</b>	<b>131</b>
Example 15.6	131
<b>16.1 Format Function for Numbers</b>	<b>133</b>
16.1(a) Built-in Format function for Numbers	133
16.1(b) User-Defined Format	135
<b>16.2 Formatting Date and Time</b>	<b>136</b>
16.2(a) Formatting Date and time using predefined formats	136
16.2(b) Formatting Date and time using user-defined formats	138
<b>17.1 Check Box</b>	<b>141</b>
Example 17.1: Shopping Cart	141
Example 17.2	143
Example 17.3	143
<b>17.2 Radio Button</b>	<b>146</b>
Example 17.4	146
Example 17.2	148
<b>18.1 Introduction</b>	<b>151</b>
<b>18.2 Using On Error GoTo Syntax</b>	<b>152</b>
Example 18.1: Division Errors	152
<b>18.3 Errors Handling using Try.....Catch.....End Try Structure</b>	<b>153</b>
<b>19.1 Concepts of Object Oriented Programming</b>	<b>156</b>
<b>19.2 Creating Class</b>	<b>157</b>
<b>20.1 Introduction</b>	<b>162</b>
<b>20.2 Creating the Graphics Object</b>	<b>163</b>
<b>20.3 Creating the Pen Object</b>	<b>163</b>



<b>20.4 Drawing a Line</b>	<b>164</b>
<b>20.5 Drawing Lines that Connect Multiple Points</b>	<b>165</b>
Example 20.1	166
<b>20.6 Drawing a curve that Connect Multiple Points</b>	<b>167</b>
Example 20.2	167
<b>20.7 Drawing Quadratic Curve</b>	<b>169</b>
Example 20.3	169
<b>20.8 Drawing Sine Curve</b>	<b>171</b>
Example 20.4	171
<b>20.8 Drawing a Rectangle</b>	<b>173</b>
<b>20.6 Customizing Line Style of the Pen Object</b>	<b>175</b>
Example 20.5	175
<b>20.7 Drawing an Ellipse</b>	<b>176</b>
Example 20.6	177
Example 20.7	179
<b>20.8 Drawing a Circle</b>	<b>179</b>
Example 20.8	179
Example 20.9	180
<b>20.9 Drawing Text</b>	<b>180</b>
Example 20.10	181
Example 20.11	183
<b>20.10 Drawing Polygons</b>	<b>184</b>
Example 20.12 Drawing a Triangle	185
Example 20.13 Drawing a Quadrilateral	
<b>20.11 Drawing a Pie</b>	<b>187</b>
Example 20.14 Drawing a pie that sweeps clockwise through 60 degree.	188
<b>20.12 Filling Shapes with Color</b>	<b>188</b>
20.12(a) Drawing and Filling a Rectangle with Color	189
Example 20.15	189

<b>20.12(b) Drawing and Filling an Ellipse with Color</b>	<b>191</b>
Example 20.16	191
<b>20.12(c) Drawing and Filling a Polygon with Color</b>	<b>192</b>
20.12(d) Drawing and Filling a Pie	193
Example 20.18	193
<b>21.1 Creating a Digital Clock</b>	<b>195</b>
<b>21.2 Creating a Stopwatch</b>	<b>196</b>
<b>21.3 Creating a Digital Dice</b>	<b>198</b>
<b>22.1 Creating Motion</b>	<b>201</b>
<b>22.2 Creating a Graphical Dice</b>	<b>203</b>
<b>22.3 Creating a Slot Machine</b>	<b>205</b>
<b>23.1 Introduction to Database</b>	<b>208</b>
<b>23.2 Creating a Database Application</b>	<b>209</b>
<b>23.3 Creating Connection to a Database using ADO.NET</b>	<b>211</b>
<b>23.4 Populating Data in ADO.NET</b>	<b>218</b>
Example 23.1	220
<b>23.5 Browsing Records</b>	<b>221</b>
<b>23.6 Editing, Saving, Adding and Deleting Records</b>	<b>223</b>
Example 23.2	223
<b>23.7 Accessing Database using DataGridView</b>	<b>227</b>
Example 23.3	228
<b>23.8 Performing Arithmetic Calculations in a Database</b>	<b>229</b>
Example 23.4	229
Example 23.5	231
Example 23.6	232
<b>24.1 Introduction</b>	<b>235</b>
<b>24.2 Reading a Text File</b>	<b>235</b>
<b>24.3 Writing to a Text File</b>	<b>240</b>
Example 25.1: Displaying a Message	245
Example 25.2	246

<b>25.3 Creating a Text File Reader in Console</b>	<b>247</b>
Example 25.3	247
<b>25.4 Creating a Console App using If...Then....Else</b>	<b>248</b>
Example 25.4	248
<b>26.1 Creating Menu Items on the Menu Bar</b>	<b>251</b>
<b>26.2 Creating The Toolbar</b>	<b>260</b>
<b>27. Deploying Your VB2017 Applications</b>	<b>266</b>

# Chapter 1

## Introduction to Visual Basic 2017

---

- ❖ A brief description of Visual Basic 2017

## ❖ Getting to know the Visual Basic 2017 Integrated Development Environment

### **1.1 A Brief Description of Visual Basic 2017**

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 and Visual Basic 2017. Although the .NET portion was discarded in 2005, all versions of the Visual Basic programming language released since 2002 are regarded as VB.NET programming language

Visual Basic 2017 was released in 2017. It comes as a .NET desktop development component of the Visual Studio Community 2017 Release Candidate integrated development environment(IDE). It is used to build windows desktop applications using the .NET framework. Besides that, Visual Studio Community 2017 RC also comes with other Windows development tools that include Universal Windows Platform Development that creates applications for the Universal Windows Platform with C#, VB , JavaScript and C++. On top of that, it also include the Desktop Development with C++.

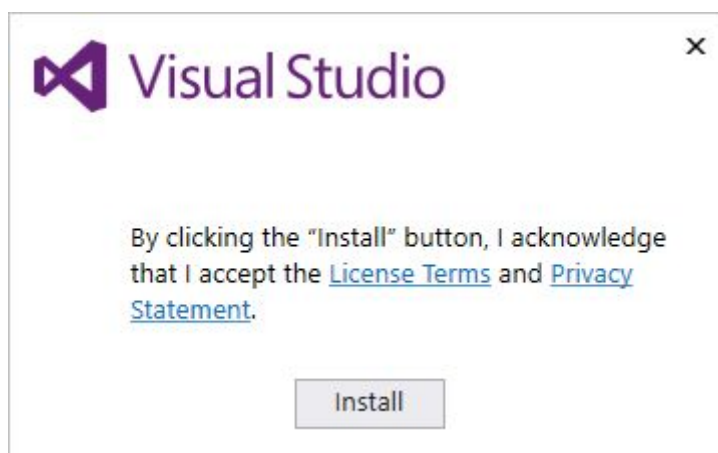
In addition, to cater for the increasing needs of web and cloud-based applications, VS2017RC also provides the Web and Cloud development tools that include ASP.NET, Python, Azure SDK, Node.js, data storage and processing, data science and analytical applications and Office/Sharepoint development. Furthermore, VS2017 also cater for the development of mobile applications by including the mobile and gaming tools such as mobile development with .NET, game development with Unity, mobile development with JavaScript, mobile development with C++ and game development with C++. With the mobile development and gaming tools, you can build IOS and Android mobile apps and mobile games.

You can download the free version of Visual Studio 2017 RC from the following link:  
<https://www.visualstudio.com/downloads/>

After downloading the file, run the VS2017RC community installer file

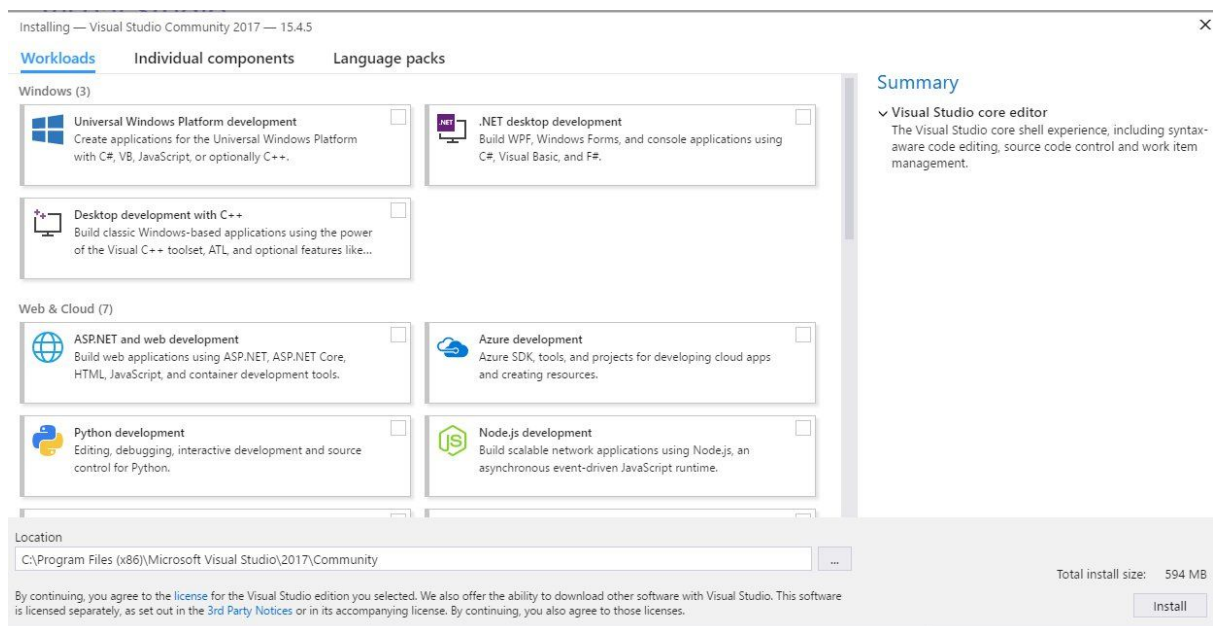
[vs\\_community\\_\\_695901156.1467100807 \(7\).exe](#).

If you receive a User Account Control notice, click Yes. Next, it will ask you to acknowledge the Microsoft License Terms and the Microsoft Privacy Statement, as shown in Figure 1.1. Click Install to continue.



**Figure 1.1**

You'll see several status screens that show the progress of the installation. After the installer has finished installing, it's time to pick the feature set that you want, as shown in Figure 1.2. Since we are keen on developing Visual Basic 2017 desktop app, we will select the .NET desktop development component. Besides that, you might want to install a particular language by clicking the Language packs. After making your selections, click install.

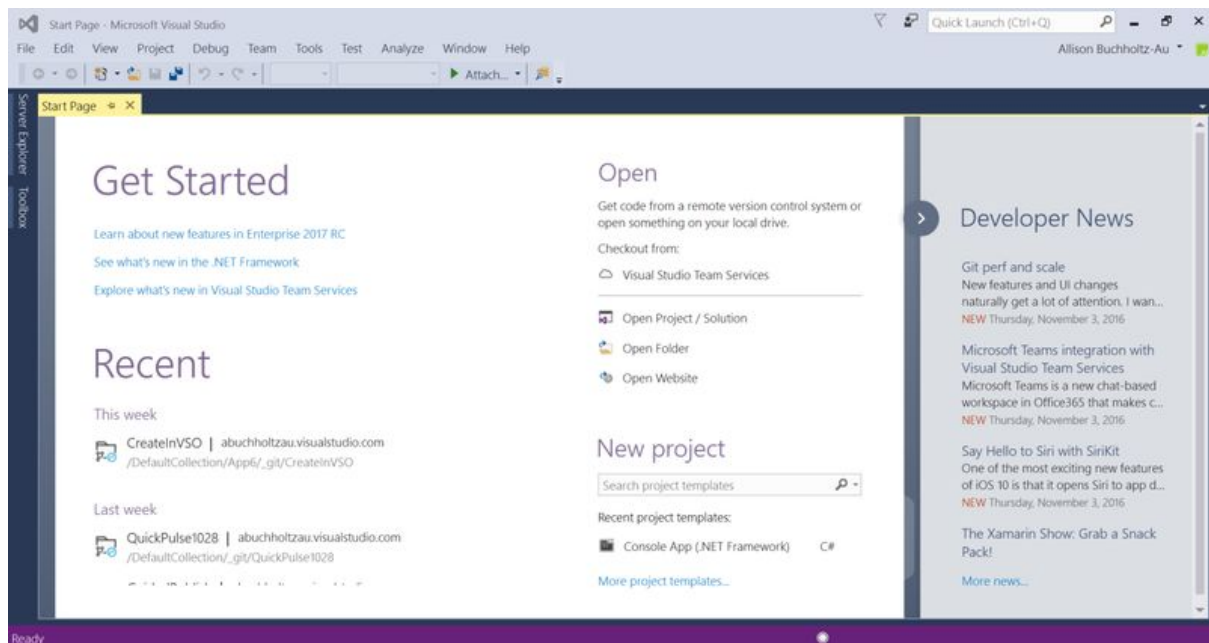


**Figure 1.2**

Upon completion of the installation, you are now ready to launch Visual Studio 2017 RC and start programming in Visual Basic 2017

## 1.2 The Visual Studio 2017 Integrated Development Environment

When you launch Microsoft Visual Studio 2017 Express, you will be presented with the Start Page of Microsoft VS 2017, as shown in Figure 1.3

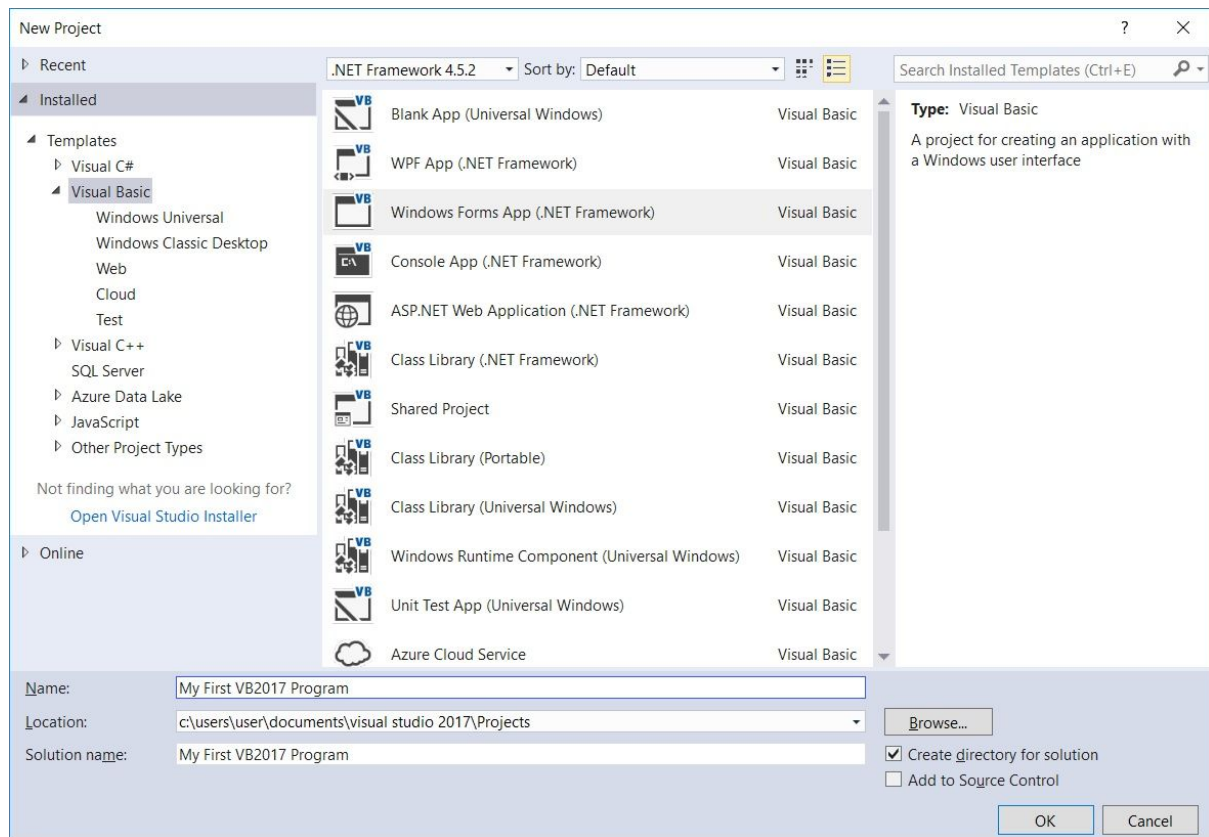


**Figure 1.3 Visual Studio Express 2017 IDE**

The Visual Studio 2017 start page comprises a few sections, the Get Started section, the Recent section, the Open section, the New project section and the Developers News section. 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 2017 Express for Windows Desktop. The Start Page also consists of a menu bar and a toolbar where you can perform various tasks by clicking the menu items.

### **1.3 Creating a New Project in Visual Studio 2017**

To start a new Visual Studio Express 2017 project, click on New Project under the Start section to launch the Visual Studio 2017 New Project page as shown in Figure 1.4. You can also choose to open a recent project:



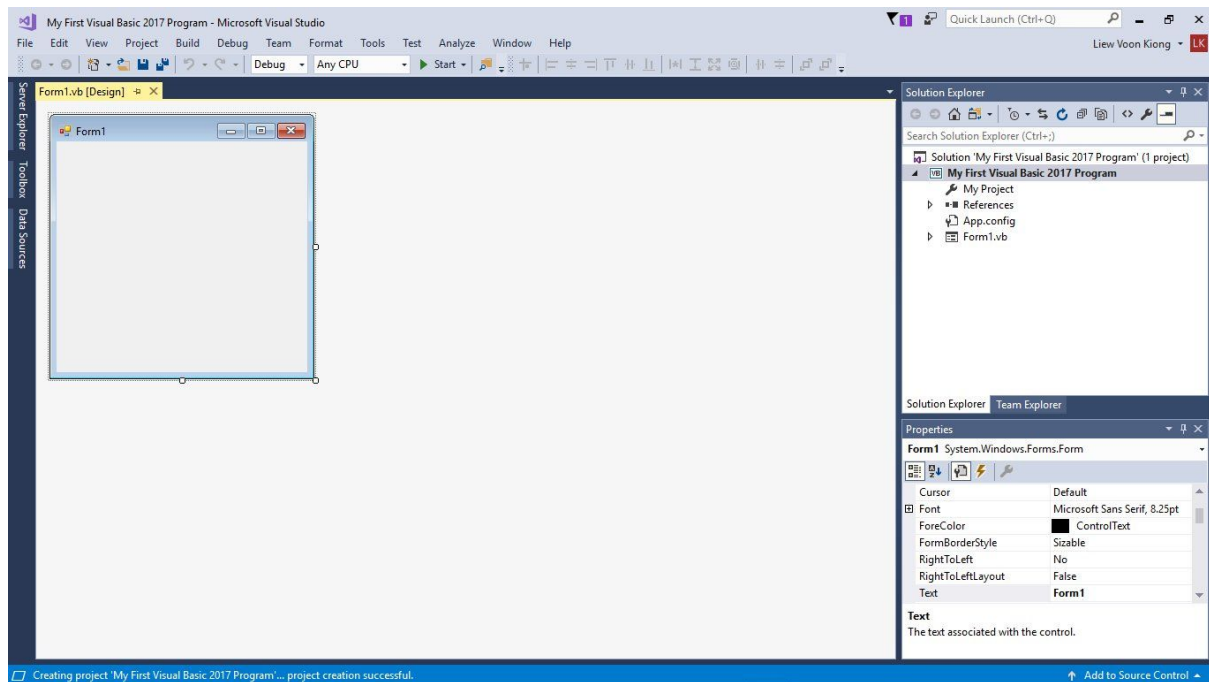
**Figure 1.4 Visual Studio 2017 Express New Project Page**

The New Project Page comprises a few templates, among them are Visual Basic, Visual C# and Visual C++. Since we are only learning Visual Basic 2017, we shall select Visual Basic. Visual Basic 2017 offers you several types of projects that you can create; they are *Blank Apps*, *Windows Forms APP(.NET Framework)*, *WPF App(.NET Framework)*, *Console App(.NET Framework)*, *Class Library(.NET Framework)*, *Shared Project* and more. Since we are only learning how to create windows desktop applications, we shall select Windows Forms App.

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 2017 Program. After you have renamed the project, click OK to continue. The Visual Basic Express 2017 IDE Windows will appear, as shown in Figure 1.5. Visual Basic Express 2017 IDE comprises a few windows, the Form window, the Solution Explorer window and the Properties window. It also consists of a toolbox which

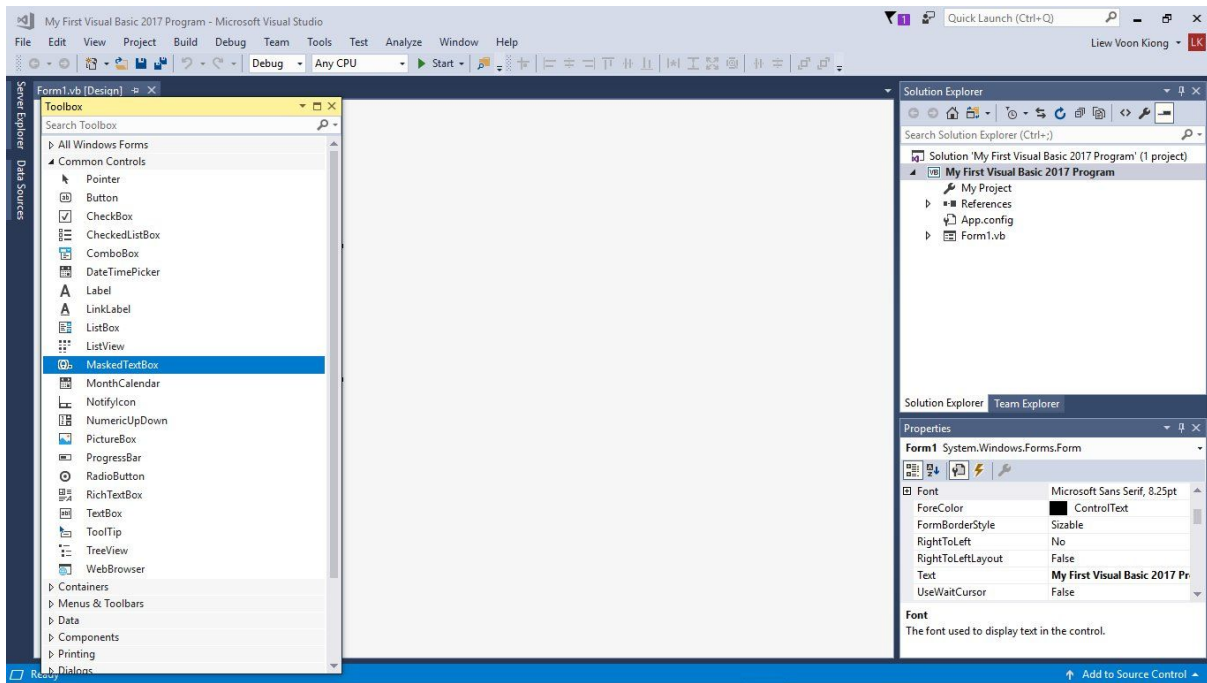


contains many useful controls that allows a programmer to develop his or her Visual Basic 2017 programs.



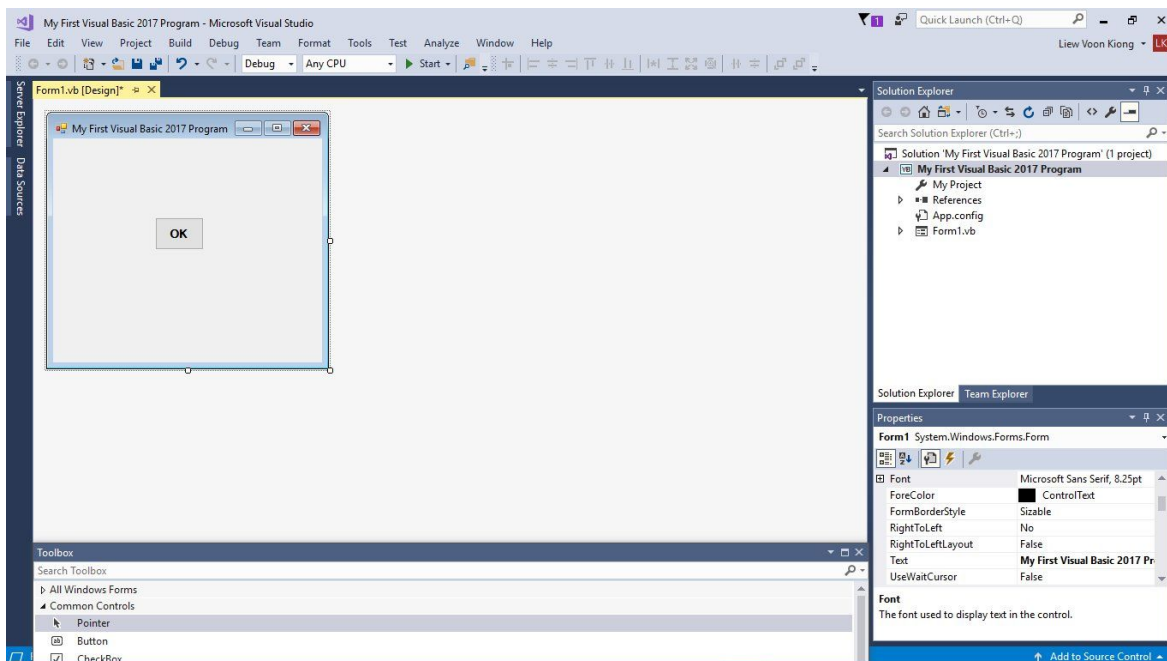
**Figure 1.5 The Visual Basic 2017 Express IDE**

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 1.6. You can drag and move your toolbox around and dock it to the right, left, top or bottom of the IDE.



**Figure 1.6 Visual Basic 2017 Express Tool Box**

Next, we shall proceed to show you how to create your first program. First, change the text of the form to 'My First Visual Basic 2017 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 1.7



**Figure 1.7 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:

```
MsgBox("My First Visual Basic 2017 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 2017 Program” message will appear, as shown in Figure 1.8. The function **MsgBox** is a built-in function of Visual Basic 2017 which can display the text enclosed within the brackets

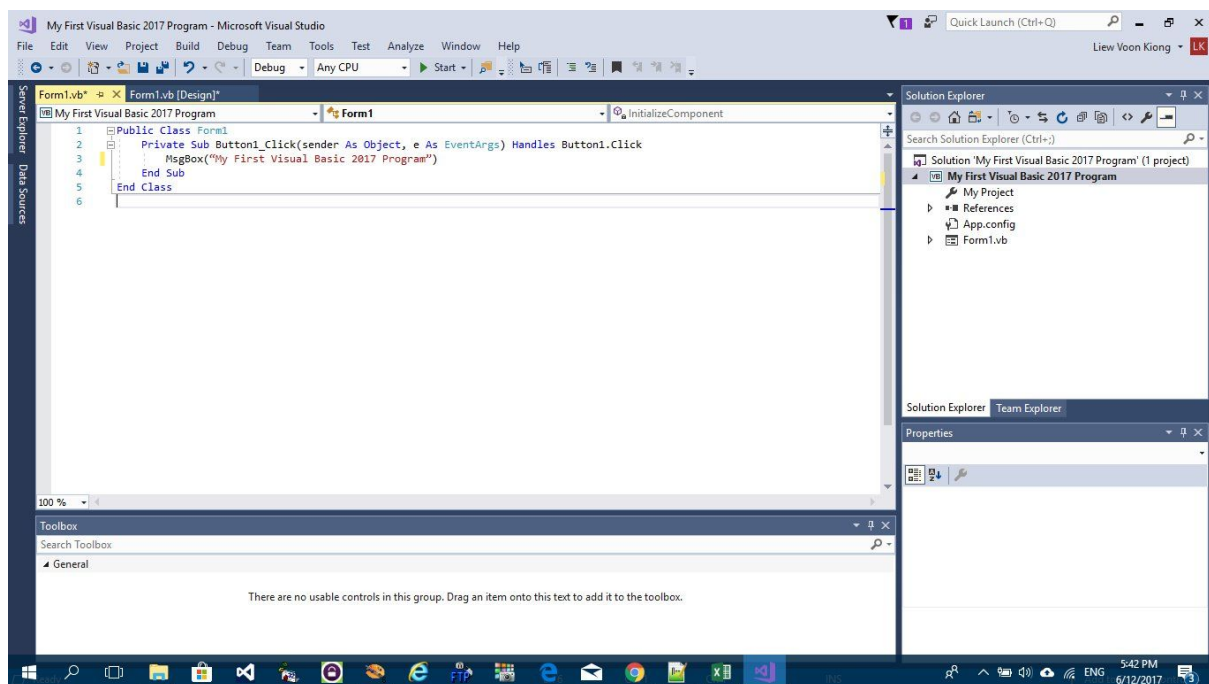


Figure 1.8 Visual Basic 2017 Code Window



**Figure 1.9 The Runtime Interface**

### **Summary**

- In section 1.1, you have learned about the history of Visual Basic 2017
- In section 1.2, you have learned how to install and launch Visual Basic Studio Express 2017
- In section 1.3, you have learned how to launch the new project dialog and the Visual Basic Express 2017 IDE. You have also learned how to write your first program.

# Chapter 2

## Designing the Interface

---

- ❖ Customizing
- ❖ Adding controls
- ❖ Setting Control Properties

As Visual Basic 2017 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, 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 2017 project, the VS2017 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 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, the size and more. Try changing the following properties:

Property	Value
Name	MyForm
Text	My First Visual Basic 2017 Program
BackColor	Aqua
ForeColor	DarkBlue
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 the 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**

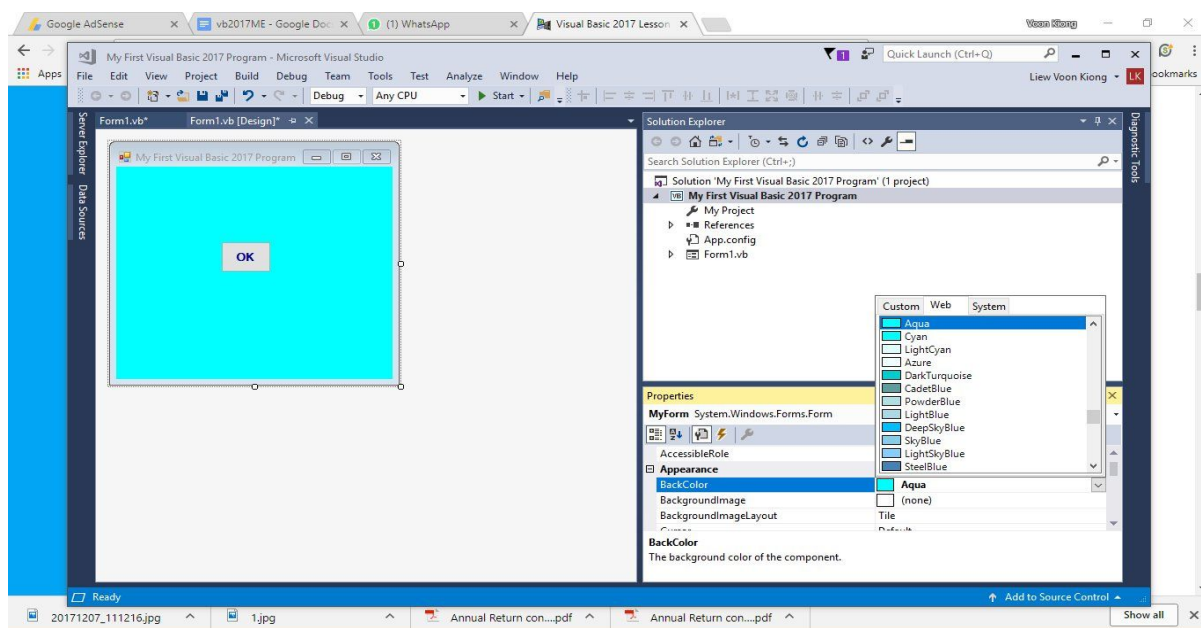
Another method of setting the colors is to manually type in the RGB color code or the hex color code. The values of R, G and B ranges from 0 to 255, therefore, by varying the values of the RGB we can obtain different colors. For example, a RGB value of 128, 255, 255 yield the cyan color.

On the other hand, the hex color code system use 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 in the range 00 to FF (in hexadecimal notation), or 0 to 255 in decimal notation. For example, **#0000ff** represents the cyan color. However, when you type in the hex color code in the properties window of VS2017, 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.

color	code	color	code	color	code	color	code	color	code
.	eeeeee	.	ffffcc	.	ffccff	.	ff99ff	.	ff66ff
.	dddddd	.	ffff99	.	ffcccc	.	ff99cc	.	ff66cc
.	cccccc	.	ffff66	.	ffcc99	.	ff9999	.	ff6666
.	bbbbbb	.	ffff33	.	ffcc66	.	ff9966	.	ff6633
.	aaaaaa	.	ffff00	.	ffcc33	.	ff9933	.	ff6633
.	999999	.	ccffff	.	ffcc00	.	ff9900	.	ff6600
.	888888	.	ccffcc	.	ccccff	.	cc99ff	.	cc66ff
.	777777	.	ccff99	.	cccccc	.	cc99cc	.	cc66cc
.	666666	.	ccff66	.	cccc99	.	cc9999	.	cc6699
.	555555	.	ccff33	.	cccc66	.	cc9966	.	cc6666
.	444444	.	ccff00	.	cccc33	.	cc9933	.	cc6633
.	333333	.	99ffff	.	cccc00	.	cc9900	.	cc6600
.	222222	.	99ffcc	.	99ccff	.	9999ff	.	9966ff
.	111111	.	99ff99	.	99cccc	.	9999cc	.	9966cc
.	ff0000	.	99ff66	.	99cc99	.	999966	.	996699
.	ee0000	.	99ff33	.	99cc33	.	999933	.	996633
.	cc0000	.	66ffff	.	66ccff	.	6699ff	.	6666ff
.	33ffff	.	00ffff	.	00ccff	.	3399ff	.	3366ff
.	ff00ff	.	cc00ff	.	00ee00	.	0000ff	.	6600ff

**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 2017 Program, background changed to aqua color , the text OK color is dark blue and the window cannot be maximized.





**Figure 2.3**



**Figure 2.4**

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
```

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)
```

To achieve 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(sender As Object, e As EventArgs) Handles  
MyBase.Load  
Me.Text = "My First Visual Basic 2017 Program"  
Me.BackColor = Color.Cyan  
Me.MaximizeBox = False  
Me.MinimizeBox = True  
End Sub
```

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

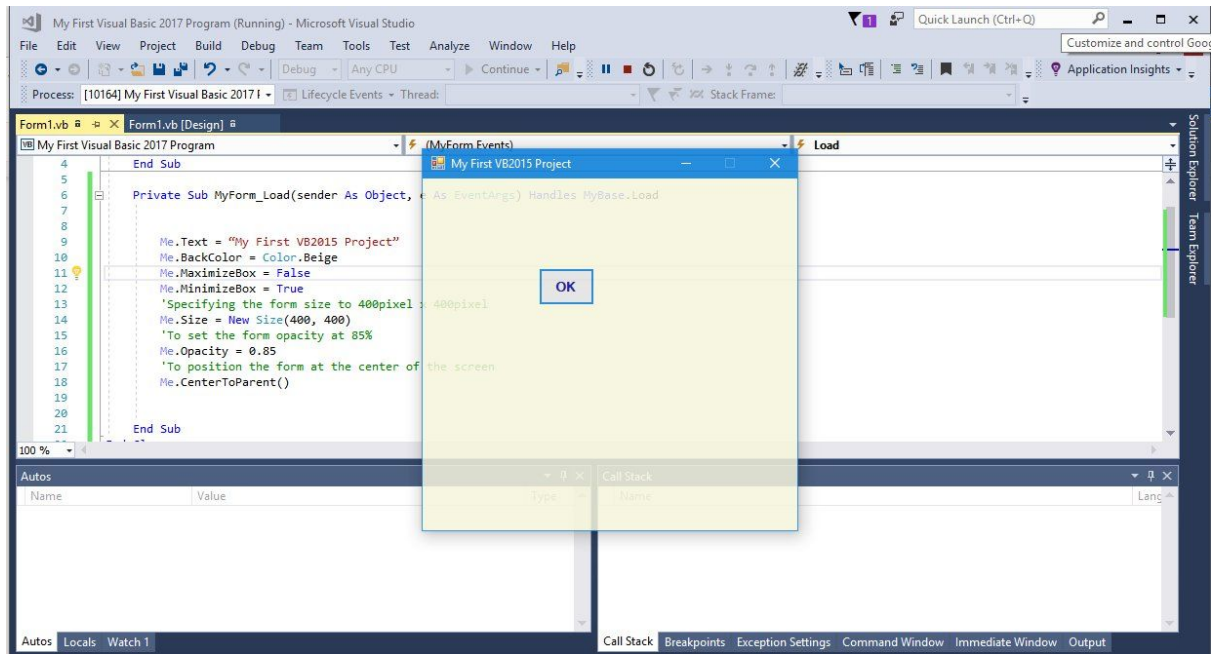
```
Me.BackColor = Color.FromArgb(0,255,255)
```

Press F5 to run the program and you will get the exact interface as that shown in Figure 2.4.

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 VB2015 Project"  
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 runtime interface is as shown in Figure 2.5

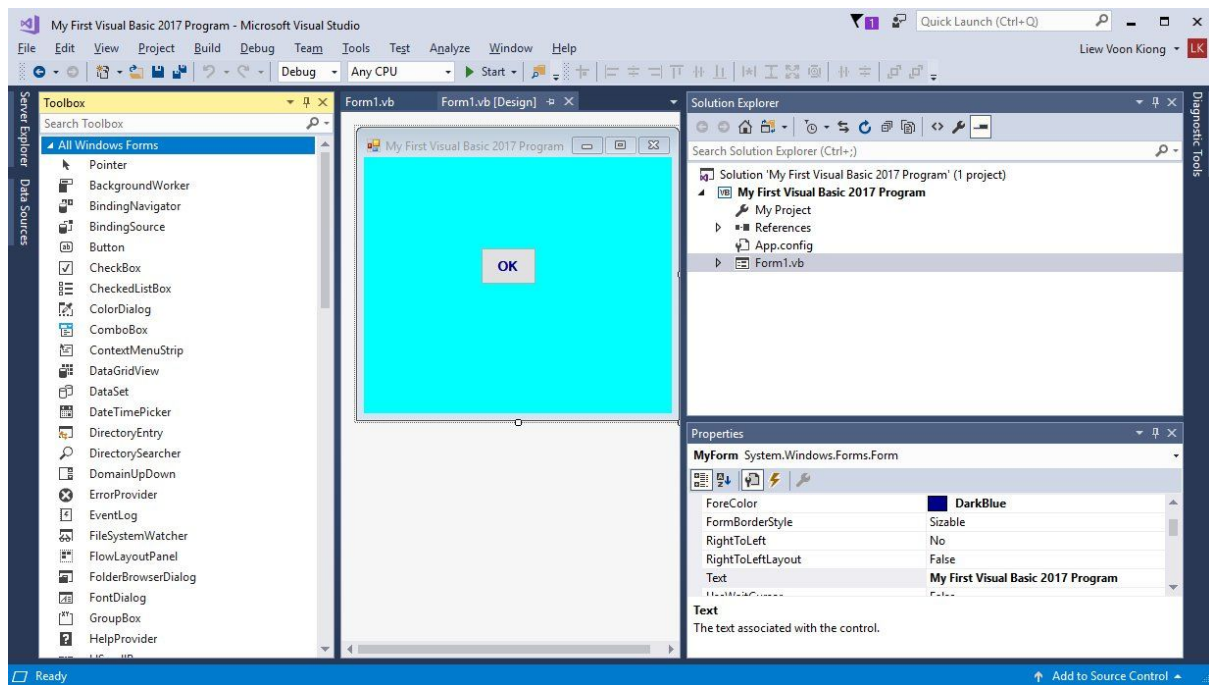


**Figure 2.5**

## 2.2 Adding Controls to the Form

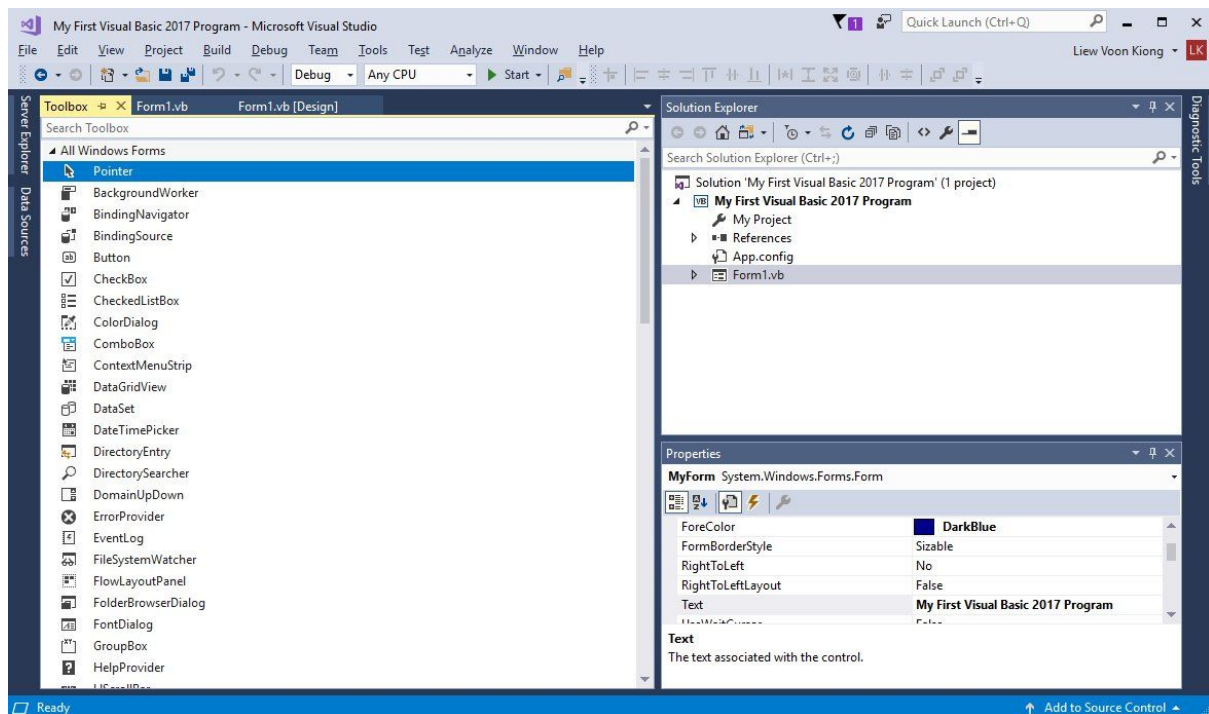
In section 2.1, we have learned how to build an initial interface in Visual Basic 2017 by customizing the default form. Next, we shall continue to build the interface by adding some controls to the form. The controls are objects that consist of three elements, namely properties, methods, and events. 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. 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 2017 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**

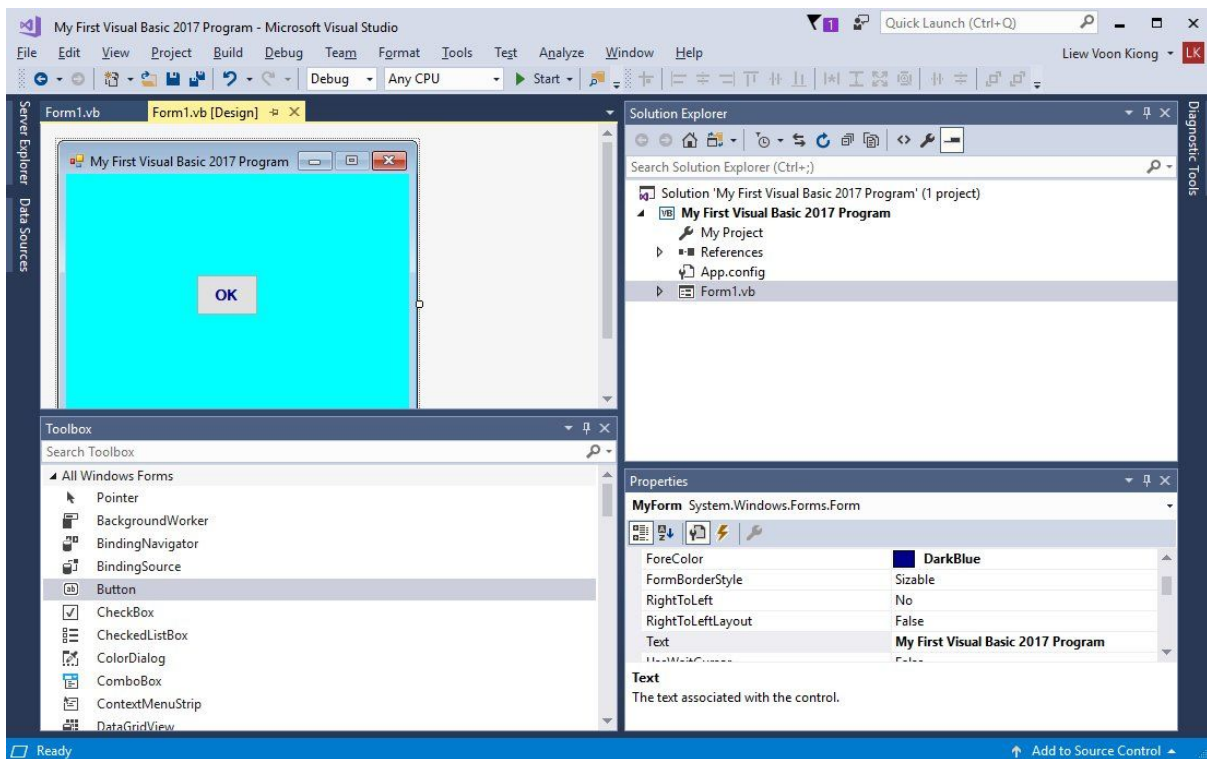
You can position the toolbox by dragging it anywhere you like while its status is set to float. You can also dock the toolbox by right-clicking on the tool box and choose dock from the pop-up menu. The docked Toolbox that appears side by side with the Solution Explorer, and as one of the tabbed windows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



**Figure 2.7**

You can also dock the tool box at the bottom, below the default form, as shown in Figure 2.8. 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 toolbox.

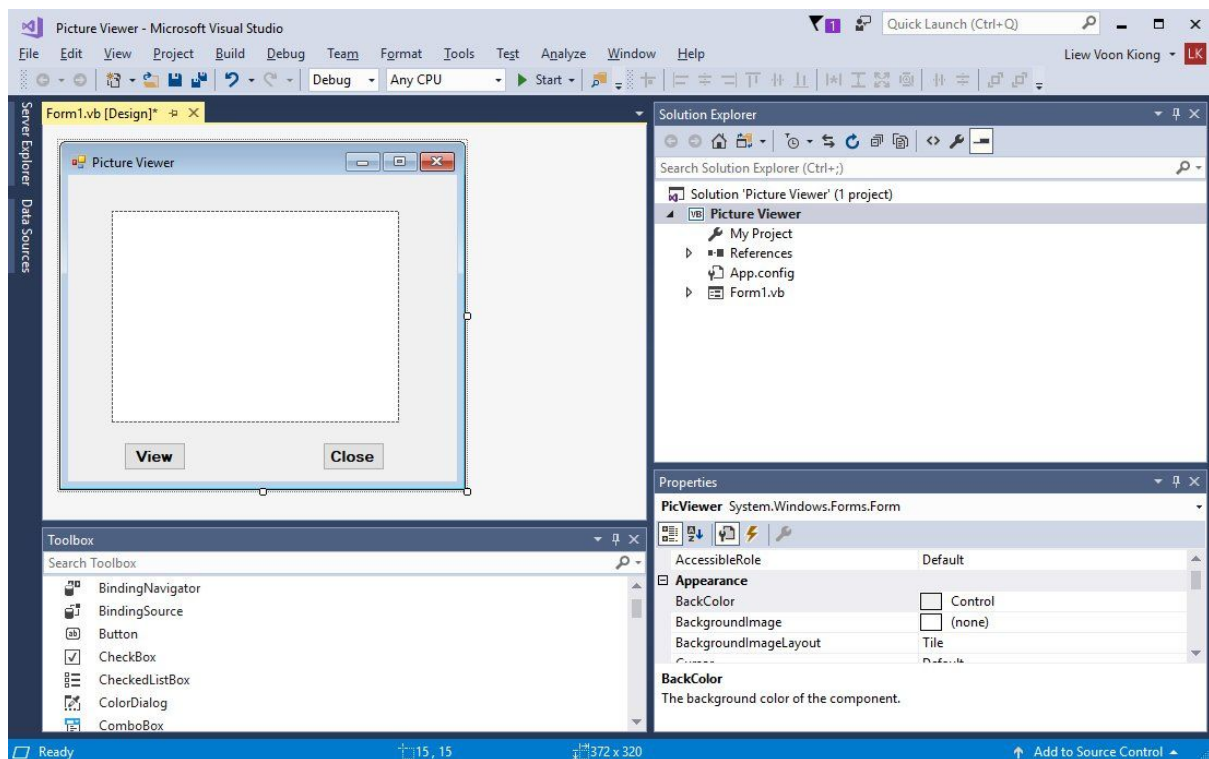
How and where you want to position your tool box is entirely up to you but we strongly suggest that you place the tool box alongside or at the bottom of the default form so that it is easy for you to add controls from the tool box into the form. You should never cover the form with the toolbox because 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 in 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

---

### ❖ Learn the basics of writing code in Visual Basic 2017

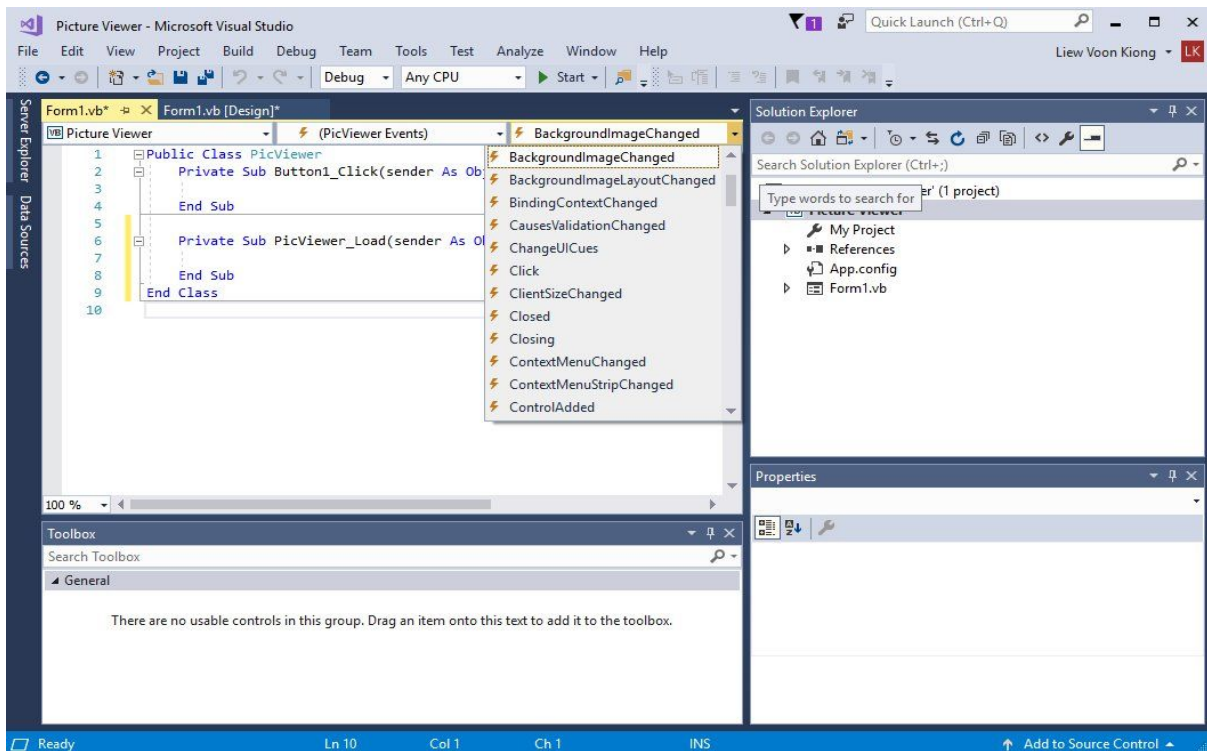
In the previous chapter, we have learned how to design the user interface by adding controls to the form and by changing their properties. However, the user interface alone will not work without adding code to them. In this chapter, we shall learn how to write code for all the controls so that they can interact with the events triggered by the users. Before learning how to write Visual Basic 2017 code, let us dwell into the concept of event-driven programming

### 3.1 The Concept of Event-Driven Programming

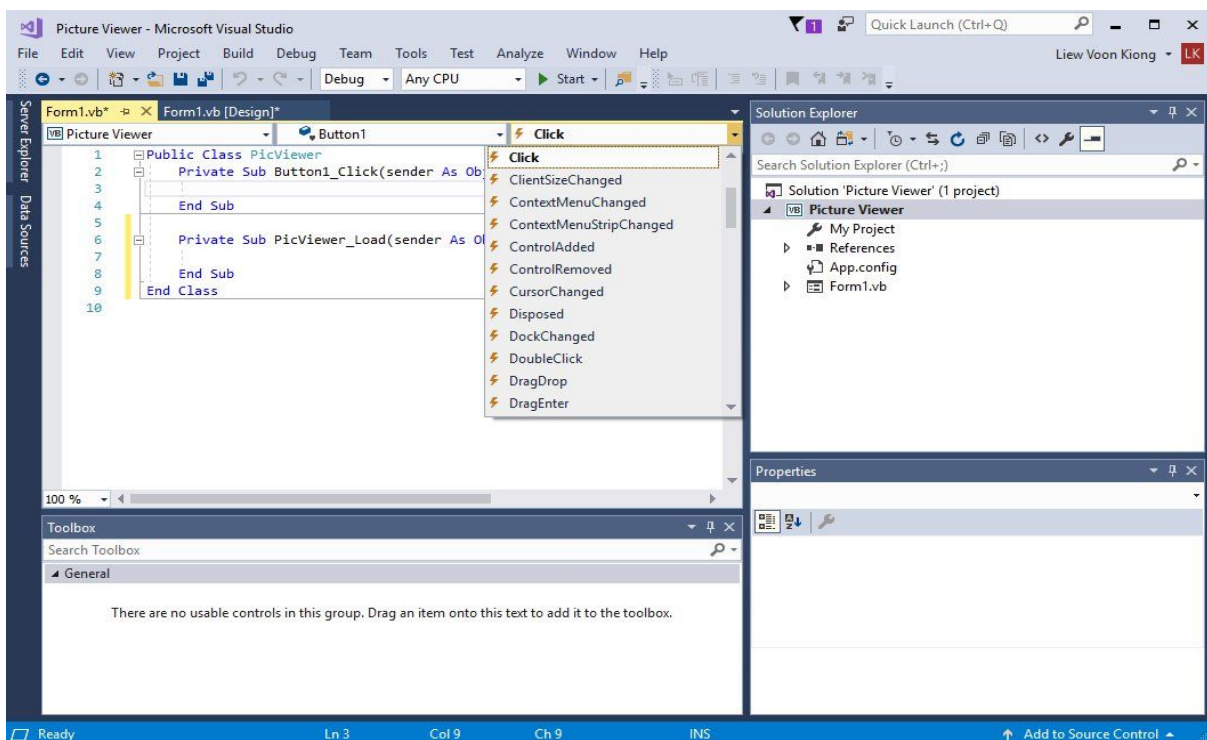
Visual Basic 2017 is an event-driven programming language which means that the code is executed in response to events triggered by the user actions like clicking the mouse, pressing a key on the keyboard, selecting an item from a drop-down list, typing some words into textbox and more. It may also be an event that runs in response to some other program. Some of the common events in Visual Basic 2017 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 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 2017, 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 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 2017" )
```

### The Code

```
Public Class Form1
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
MsgBox ( "My First Visual Basic 2017 Program", , "My Message")
End Sub
End Class
```

MsgBox is a built-in function in Visual Basic 2017 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 2017 Program” will appear, as shown in Figure 3.3.



### 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 2017, 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:

```
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