

جامعة بغداد / كلية التربية للعلوم الصرفة ابن الهيثم / قسم علوم الحاسبات
المرحلة الثالثة صباحي / مسائي (نظري)
للعام الدراسي ٢٠٢٤-٢٠٢٥
اساتذة المادة: م . د. شهلاء طالب

Visual Programming

Lecture 2 –Basic Controls and Events

Basic Controls

VB.Net provides a huge variety of controls that help you to create rich user interface. Functionalities of all these controls are defined in the respective control classes. The control classes are defined in the **System.Windows.Forms** namespace.

The following table lists some of the commonly used controls –

Sr.No.	Widget & Description
1	Forms  The container for all the controls that make up the user interface.
2	TextBox  It represents a Windows text box control.
3	Label  It represents a standard Windows label.
4	Button  It represents a Windows button control.
5	ListBox  It represents a Windows control to display a list of items.
6	ComboBox  It represents a Windows combo box control.

7	RadioButton  It enables the user to select a single option from a group of choices when paired with other RadioButton controls.
8	CheckBox  It represents a Windows CheckBox.
9	PictureBox  It represents a Windows picture box control for displaying an image.
10	ProgressBar  It represents a Windows progress bar control.
11	ScrollBar  It Implements the basic functionality of a scroll bar control.
12	DateTimePicker  It represents a Windows control that allows the user to select a date and a time and to display the date and time with a specified format.

Controls Properties

Forms and controls have **properties**, **events**, and **methods**. Together they make the forms and controls useful for programmers.

You can change the appearance of the controls (and form) by setting their properties in the properties window.

Here is a shortlist of the properties we use in the course:

Property Name	Objective	Code	Stage of changing
Text	String appear in title of control	TextBox1.Text = "any text" Label1.Text = "any text" Button1.Text = "any text"	Design and Run
MultiLine	To enter more than one line in TextBox1 only	True or False	Design
BackColor	Background color for control.	TextBox1.BackColor = Color.anycolor Label1.BackColor = Color.anycolor Button1.BackColor = Color.anycolor	Design and Run
ForeColor	Color of text written on control.	TextBox1.ForeColor = Color.anycolor Label1.ForeColor = Color.anycolor Button1.ForeColor = Color.anycolor	Design and Run

TextAlign	The horizontal and vertical alignment of the text inside the control.	Left/Right/Center From properties	Design and Run
Left Top	Horizontal/vertical position of the control, counted by the number of pixels relative to the left/top side of its parent.	TextBox1.Left = no, TextBox1.Top = no Label1.Left= no, label1.Top = no Button1.Left= no , Button1.Top = no	Run
Width Height	The width or height of the control, counted by the number of pixels.	TextBox1.Width = no, TextBox1.Hight = no Label1.Width= no, label1.Hight= no Button1.Width= no , Button1.Hight = no	Run
Hide	To hide the control	TextBox1.Hide() Label1.Hide() Button1.Hide()	Run
Visible	The control appear or disappear	TextBox1 .Visible = True or False Label1. Visible = True or False Button1.Visible = True or False	Design and Run
Enabled	The control enable or disable	TextBox1.Enabled = True or False Label1. Enabled= True or False Button1. Enabled= True or False	Design and Run

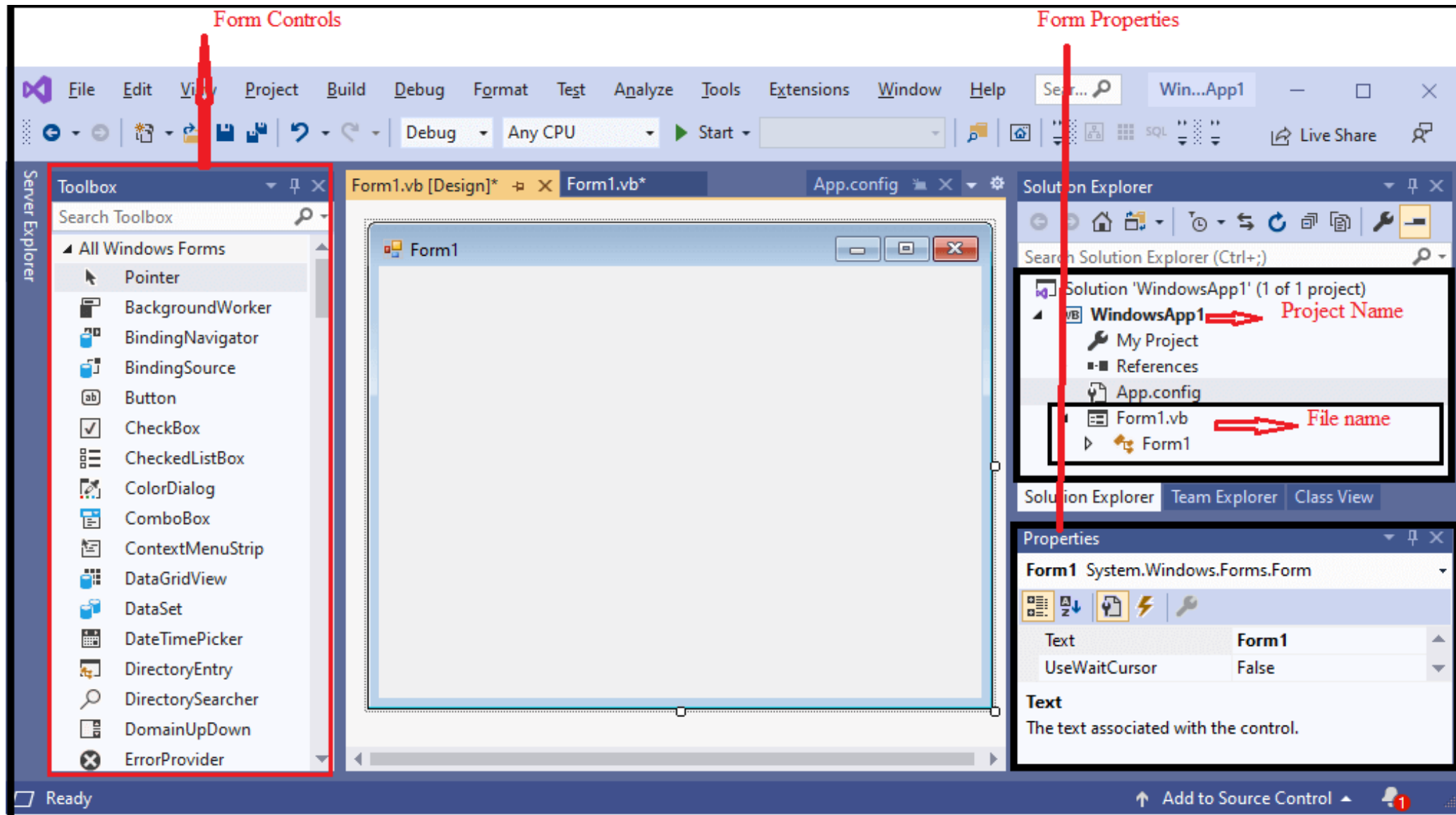
For creating a Windows Forms application in [VB.NET](#), we need to follow the following steps in Microsoft [Visual Studio](#).

- 1.GoTo File Menu.

- 2.Click on New Project.

- 3.Click on Windows Forms App or Application

And finally, click on the 'Create' button to create your project, and then, it displays the following window form with a name Form1.



Form1

Enter the Name

Enter Your lucky number

...

Form1

Enter the Name

Enter Your lucky number

...

Form1

Enter the Name

Enter Your lucky number

You have entered the Name JavaTpoint Number 7

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

```
    nameStr = TextBox1.Text
```

```
    num = TextBox2.Text
```

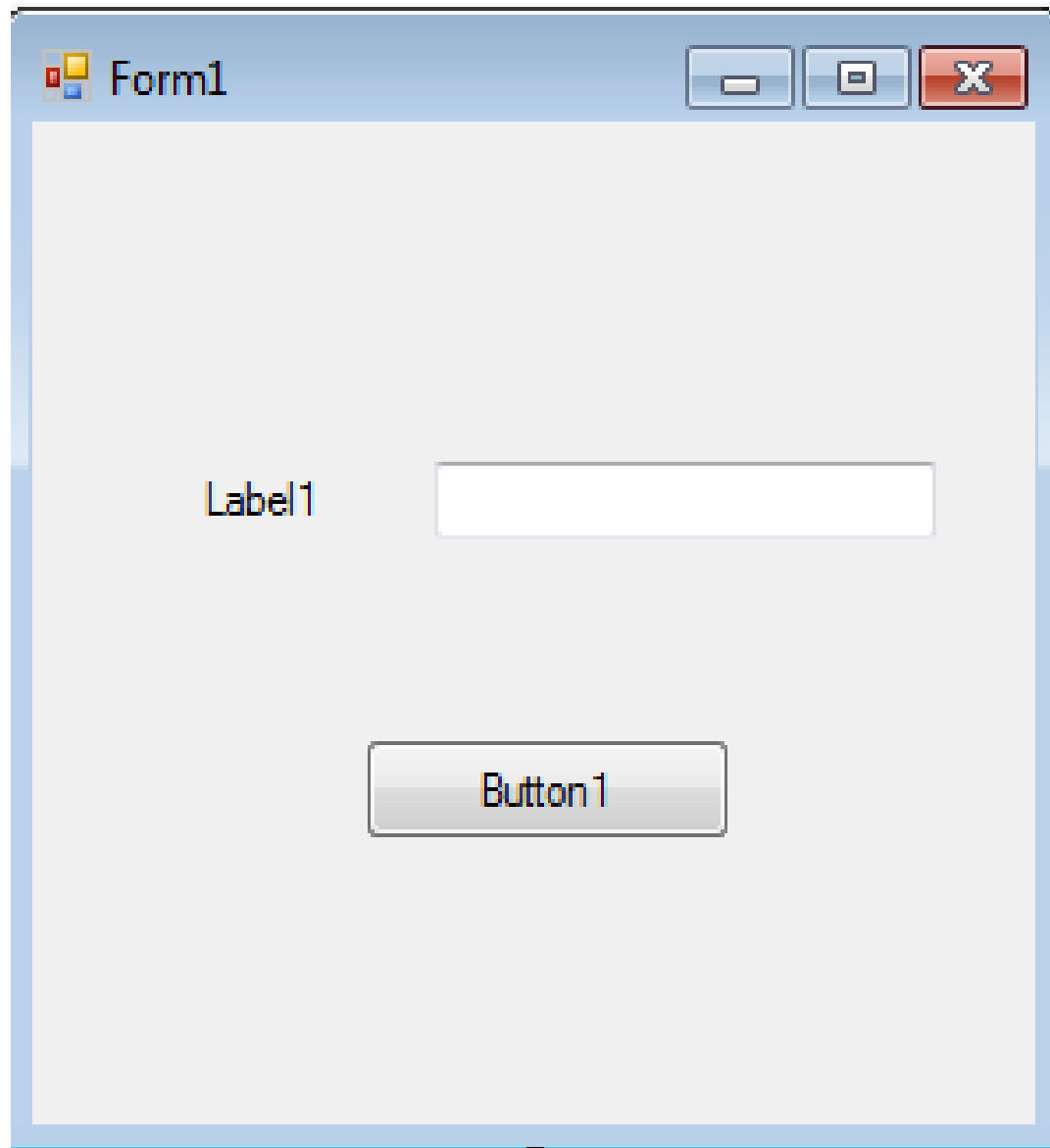
```
    Label3.Text = "You have entered the Name " & nameStr + " Number  
" & num
```

```
End Sub
```

Here is an example program that changes some properties of the form and the controls. To enter the code into Visual Basic IDE, you can double click Button1 in design view. Can you guess what will happen after Button1 is clicked?

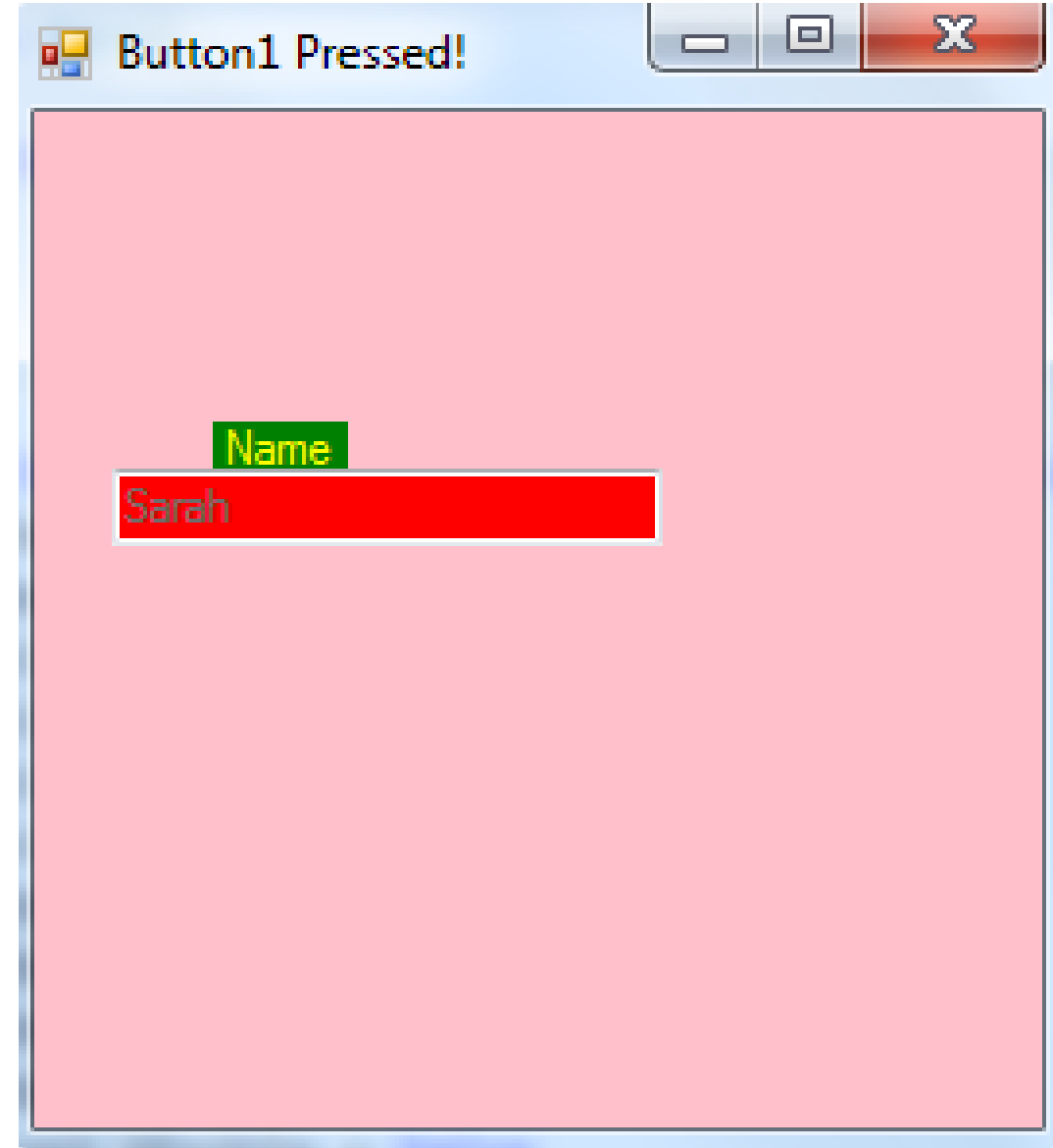
```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles  
Button1.Click  
    Me.Text = "Button1 Pressed!"  
    Me.BackColor = Color.Pink  
    Label1.Text = "Name"  
    Label1.BackColor = Color.Green  
    Label1.ForeColor = Color.Yellow  
    Label1.Top = 80  
    TextBox1.Text = "Sarah"  
    TextBox1.BackColor = Color.Red  
    TextBox1.Enabled = False  
    TextBox1.Left = 20  
    Button1.Visible = False  
  
End Sub
```

Before



A screenshot of a Windows-style window titled "Form1". The window has a light gray background and a blue title bar with standard minimize, maximize, and close buttons. Inside the window, there is a label "Label1" on the left, a white text input field in the center, and a button labeled "Button1" at the bottom.

After



A screenshot of the same window after the button has been pressed. The title bar now reads "Button1 Pressed!". The background of the window is a solid light pink. In the center, there is a green rectangular label with the text "Name" in yellow. Below it is a red rectangular text input field containing the text "Sarah".

Exercise 1:

Write a program with two controls: Button1 and TextBox1. When Button1 is clicked, the following things should happen:

- (a) TextBox1 is disabled,
- (b) The background color of TextBox1 becomes yellow,
- (c) Button1 becomes visible, and
- (d) The form's background color becomes white.

Exercise 2:

Identify the mistakes in the following source code. There is one mistake in each line.

(Note: There are no mistakes with the words Me, Label1, Button1 and TextBox1.)

```
Me.Title = "Title of the form"
```

```
Label1.BackColor = Colour.Green
```

```
Button1.Visible = Ture
```

```
TextBox1.Enable = False
```

```
TextBox1.Text = Very good
```

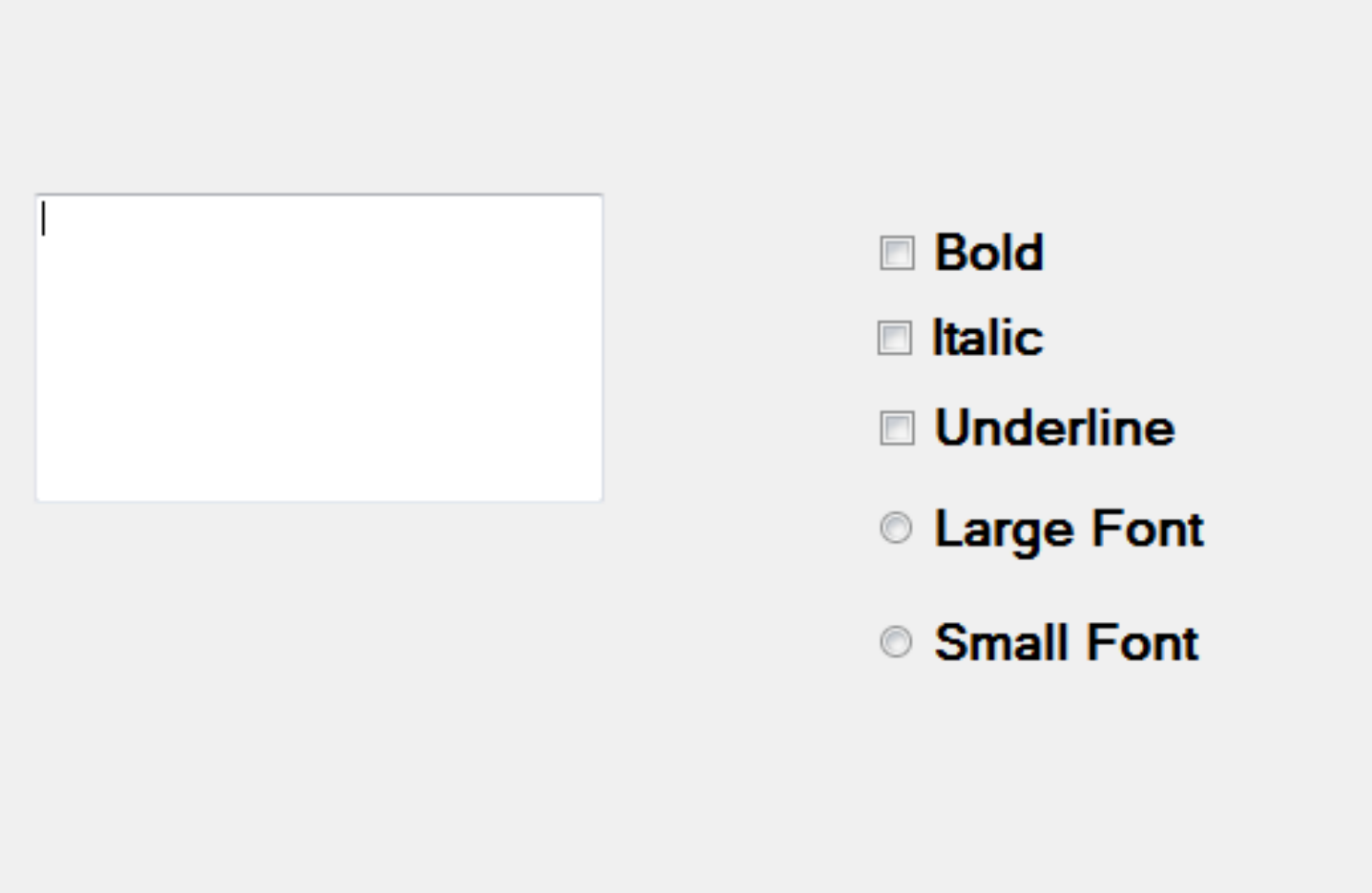
CheckBox and RadioButton

The checkbox is a control that allows the user to select multiple items.

The radio button is another control in Visual Basic 2012 that allows selection of choices. However, it operates differently from the CheckBox. While the CheckBoxes allow the user to select one or more items, radio buttons are mutually exclusive, which means the user can only choose one item only out of a number of choices.

Example

In this example, the user can enter text into a TextBox and format the font using the three CheckBoxes that represent bold, italic and underline. Also change the size of text into a TextBox by using two RadioButtons.



The screenshot shows a standard Windows application window titled "Form1". Inside the window, on the left side, is a white rectangular text box with a thin black border and a vertical cursor at the top left. To the right of the text box, there is a vertical list of five formatting options. Each option consists of a small icon followed by the text label. The first three options are "Bold", "Italic", and "Underline", each preceded by a square checkbox icon. The last two options are "Large Font" and "Small Font", each preceded by a circular radio button icon. All text labels are in a bold, black, sans-serif font.

Form1

☐ **Bold**

☐ **Italic**

☐ **Underline**

☐ **Large Font**

☐ **Small Font**

```
Public Class Form1
```

```
    Private Sub CheckBox1_CheckedChanged(sender As Object, e As EventArgs) Handles CheckBox1.CheckedChanged
```

```
        If CheckBox1.Checked Then
```

```
            TextBox1.Font = New Drawing.Font("Times New Roman", 20, FontStyle.Bold)
```

```
        End If
```

```
    End Sub
```

```
    Private Sub CheckBox2_CheckedChanged(sender As Object, e As EventArgs) Handles CheckBox2.CheckedChanged
```

```
        If CheckBox2.Checked Then
```

```
            TextBox1.Font = New Drawing.Font("Times New Roman", 20, FontStyle.Italic)
```

```
        End If
```

```
    End Sub
```

```
    Private Sub CheckBox3_CheckedChanged(sender As Object, e As EventArgs) Handles CheckBox3.CheckedChanged
```

```
        If CheckBox3.Checked Then
```

```
            TextBox1.Font = New Drawing.Font("Times New Roman", 20, FontStyle.Underline)
```

```
        End If
```

```
    End Sub
```

```
    Private Sub RadioButton1_CheckedChanged(sender As Object, e As EventArgs) Handles RadioButton1.CheckedChanged
```

```
        TextBox1.Font = New Drawing.Font("Times New Roman", 20)
```

```
    End Sub
```

```
    Private Sub RadioButton2_CheckedChanged(sender As Object, e As EventArgs) Handles RadioButton2.CheckedChanged
```

```
        TextBox1.Font = New Drawing.Font("Times New Roman", 10)
```

```
    End Sub
```

```
End Class
```

Visual Basic Calculator

The image shows a Windows-style application window titled "Simple Calculator". The window has a standard title bar with minimize, maximize, and close buttons. The main content area is divided into two primary sections. On the left, under the heading "Simple Calculator", is a group box labeled "Operators:". This group box contains seven buttons: "+" and "-" in the first row, "*" and "/" in the second row, "\" and "^" in the third row, and a "Mod" button spanning the width of the group in the fourth row. On the right, there is another group box labeled "Operation:". This group box contains three text input fields: "Operand 1:" at the top, an empty space with a small rectangular box in the middle, "Operand 2:" below that, and "Result" at the bottom. At the very bottom of the window, there are two buttons: "Clear" and "Exit".

Simple Calculator

Simple Calculator

Operators:

+

-

*

/

\

^

Mod

Operation

Operand 1:

Operand 2:

Result

Clear

Exit

Events

Events are basically a user action like key press, clicks, mouse movements, etc., or some occurrence like system generated notifications. Applications need to respond to events when they occur.

Clicking on a button, or entering some text in a text box, or clicking on a menu item, all are examples of events. An event is an action that calls a function or may cause another event. Event handlers are functions that tell how to respond to an event.

VB.Net is an event-driven language. There are mainly two types of events –

- Mouse events
- Keyboard events

Handling Mouse Events

Mouse events occur with mouse movements in forms and controls. Following are the various mouse events related with a Control class –

- **MouseDown** – it occurs when a mouse button is pressed
- **MouseEnter** – it occurs when the mouse pointer enters the control
- **MouseHover** – it occurs when the mouse pointer hovers over the control
- **MouseLeave** – it occurs when the mouse pointer leaves the control
- **MouseMove** – it occurs when the mouse pointer moves over the control
- **MouseUp** – it occurs when the mouse pointer is over the control and the mouse button is released
- **MouseWheel** – it occurs when the mouse wheel moves and the control has focus

Handling Keyboard Events

Following are the various keyboard events related with a Control class –

- **KeyDown** – occurs when a key is pressed down and the control has focus
- **KeyPress** – occurs when a key is pressed and the control has focus
- **KeyUp** – occurs when a key is released while the control has focus

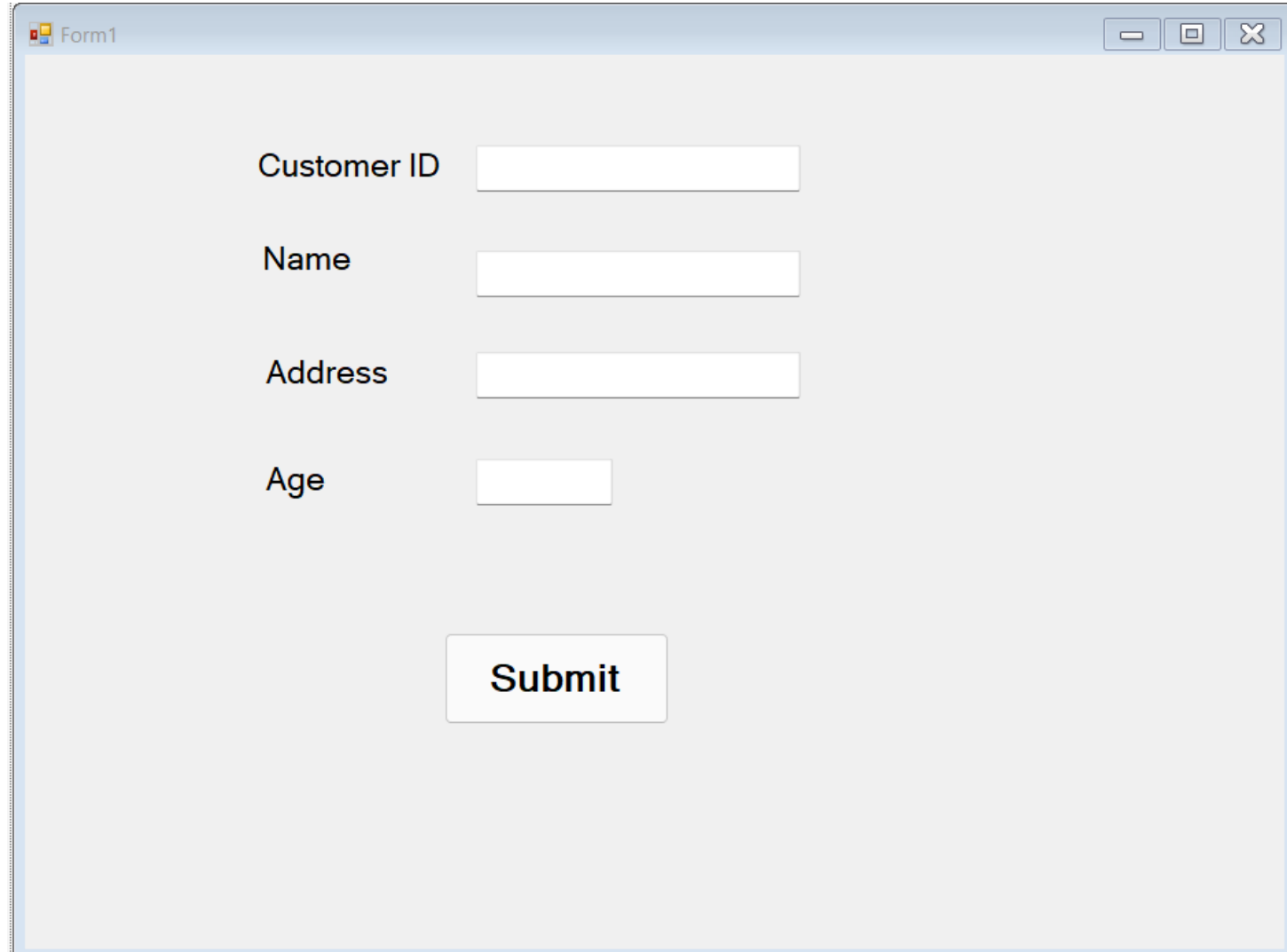
The event handlers of the KeyDown and KeyUp events get an argument of type **KeyEventArgs**. This object has the following properties –

- **Alt** – it indicates whether the ALT key is pressed
- **Control** – it indicates whether the CTRL key is pressed
- **Handled** – it indicates whether the event is handled
- **KeyCode** – stores the keyboard code for the event
- **KeyData** – stores the keyboard data for the event
- **KeyValue** – stores the keyboard value for the event
- **Modifiers** – it indicates which modifier keys (Ctrl, Shift, and/or Alt) are pressed
- **Shift** – it indicates if the Shift key is pressed

The event handlers of the KeyDown and KeyUp events get an argument of type **KeyEventArgs**. This object has the following properties –

- **Handled** – indicates if the KeyPress event is handled
- **KeyChar** – stores the character corresponding to the key pressed

Example: The code will verify that the user enters some numbers for his customer ID, Name and age.



The image shows a Windows application window titled "Form1". Inside the window, there is a form with four text input fields and one submit button. The labels for the input fields are "Customer ID", "Name", "Address", and "Age". The "Customer ID", "Name", and "Address" fields are of a standard size, while the "Age" field is smaller. Below these fields is a button labeled "Submit".

Field Label	Field Type
Customer ID	Text Input
Name	Text Input
Address	Text Input
Age	Text Input
Submit	Button

```
Public Class Form1
    Private Sub TextBox1_KeyUp(sender As Object, e As KeyEventArgs) Handles TextBox1.KeyUp
        If (Not Char.IsNumber(ChrW(e.KeyCode))) Then
            MessageBox.Show("Enter numbers for your Customer ID")
            TextBox1.Text = " "
        End If
    End Sub
    Private Sub TextBox4_KeyUp(sender As Object, e As KeyEventArgs) Handles TextBox4.KeyUp
        If (Not Char.IsNumber(ChrW(e.KeyCode))) Then
            MessageBox.Show("Enter numbers for age")
            TextBox4.Text = " "
        End If
    End Sub
    Private Sub TextBox2_KeyUp(sender As Object, e As KeyEventArgs) Handles TextBox2.KeyUp
        If (Char.IsNumber(ChrW(e.KeyCode))) Then
            MessageBox.Show("Enter char for name")
            TextBox2.Text = " "
        End If
    End Sub
End Class
```