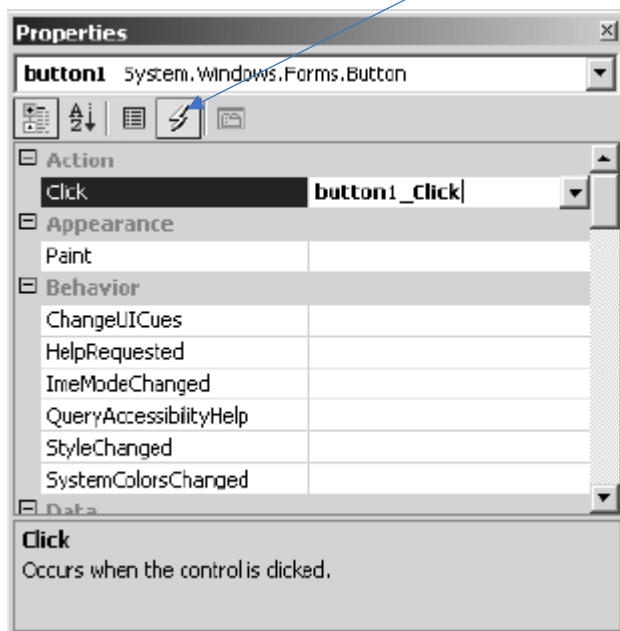


## EVENT HANDLING IN VB.NET

Event handling is performed differently in VB.NET than it is in Visual C++. In Visual C++, each control is derived from CWnd and a message map handles any events. The message map concept doesn't exist in VB.NET. The Control class, or any derived class, has virtual functions, which can be overridden to raise an event. Therefore, a Form class can use any event handler in its hierarchy. For example, the Control class has many event methods including GotFocus, ControlRemoved, LostFocus, and MouseWheel. Table 9.7 lists some common event-handling methods.

### Adding Event Handlers at Design-Time

To attach events to a control at design-time, use the Properties window. Right-click the control and click the Properties menu item. Select the Events tab by pressing the lightening button at the top of the Properties window. Now, pick an event and type the corresponding function name. Alternatively, you could double-click in the field next to the event and VS.NET will choose a name for you. In Figure 9.17, we add a button click event handler as button1\_Click.



The event handler method takes the following form:

```
Private Sub button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)  
End Sub
```

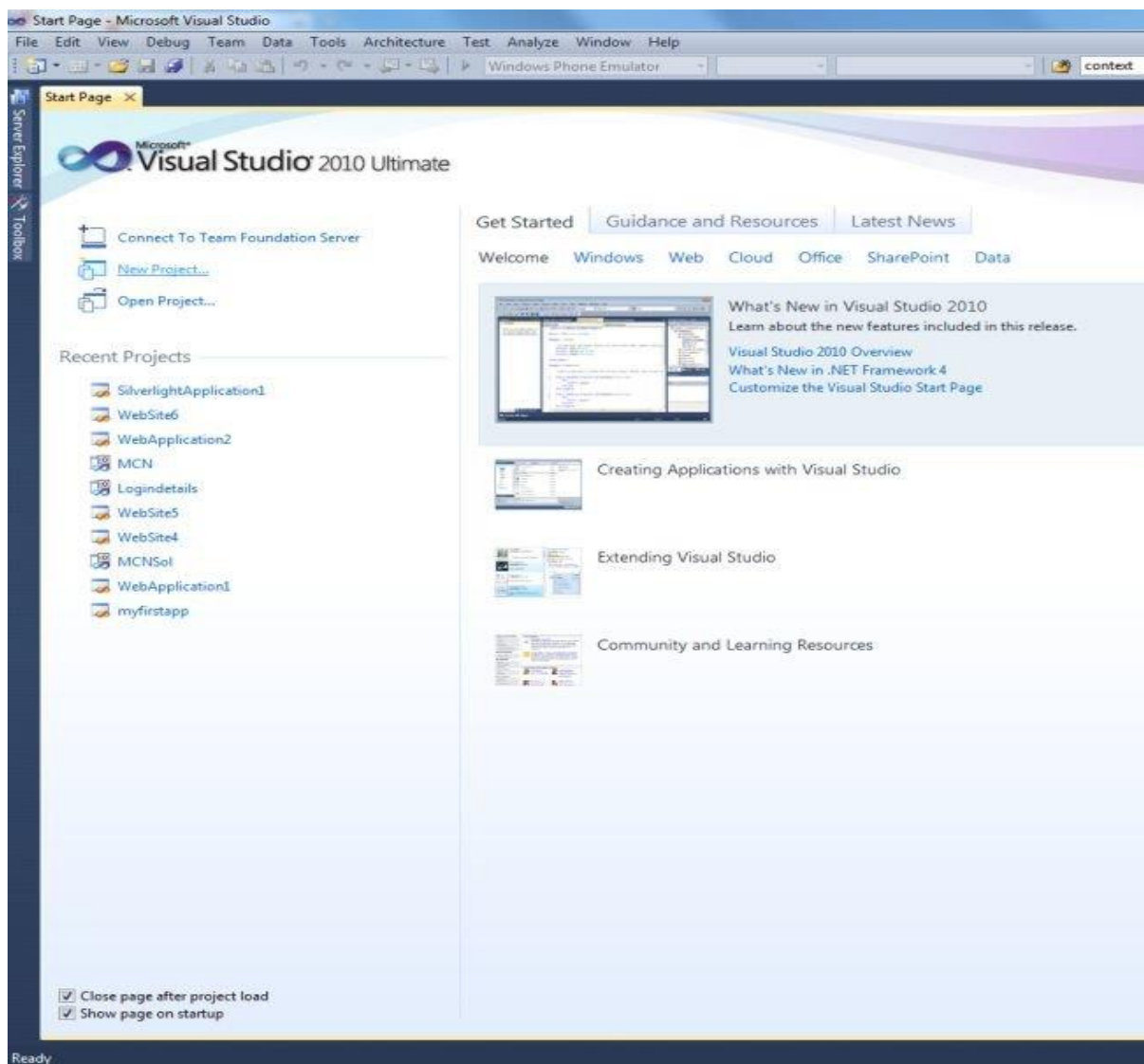
Now write whatever code, within the method, you wish to execute after the event takes place. You can use the "e" (event argument) parameter that is passed to retrieve any pertinent information about the event, such as which key was pressed. The sender parameter is the control that initiated the event, in this case, the button1 member.

## Event Handling On Click Event In VB.NET

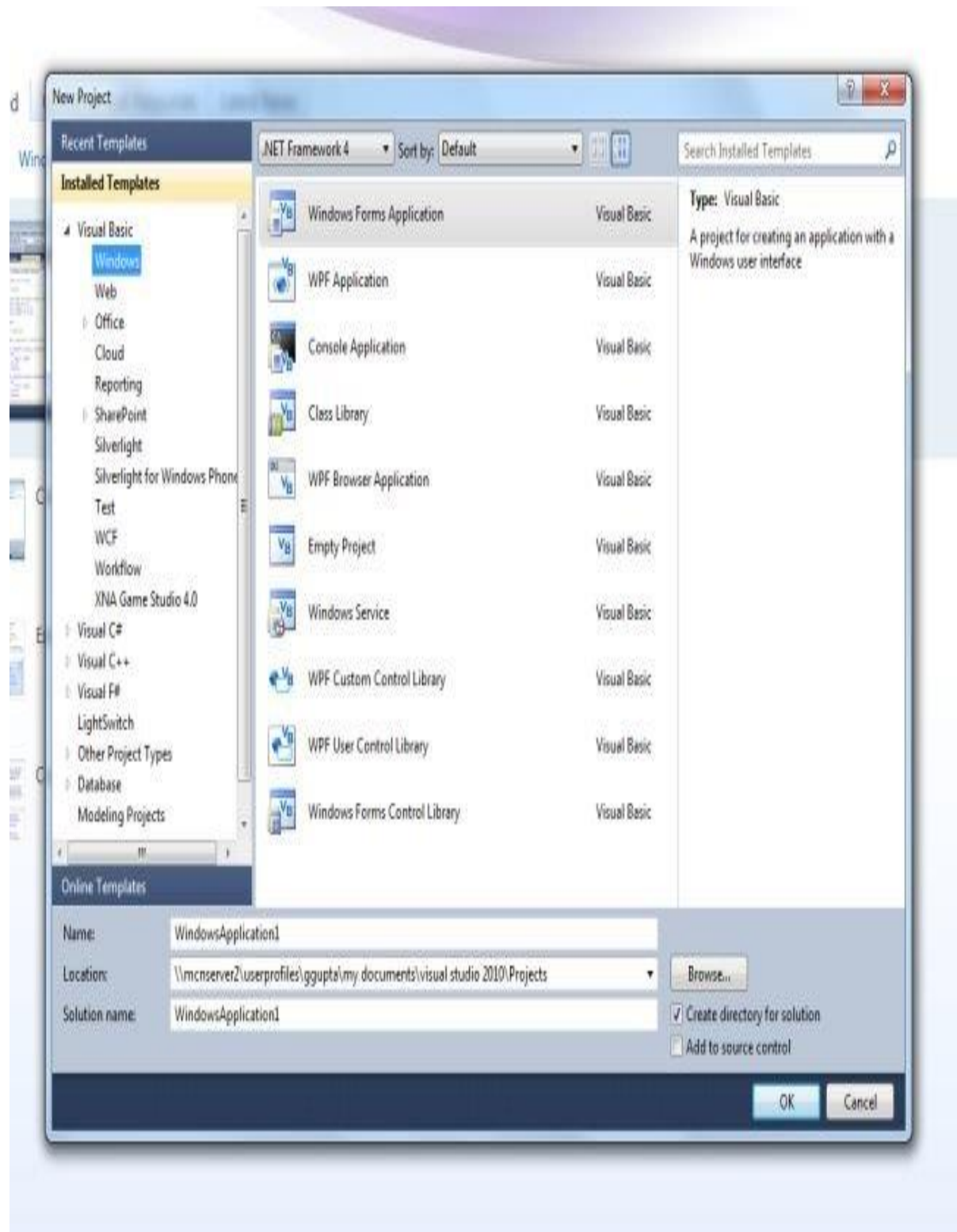
### Introduction

Here i will discuss the Event Handler for the Click event of the Calculate and Exit button. In this Calculate button calculates the discount percent, discount amount, and total based on the subtotal entered by the user whereas Exit button contains one statement that executes the Close method of the form. In this i have apply a condition, if a user enter the value more than 400, it will show value less than 400 will be accept and if user enter the value less than 400 it will calculate it. To perform this example, follow these steps:

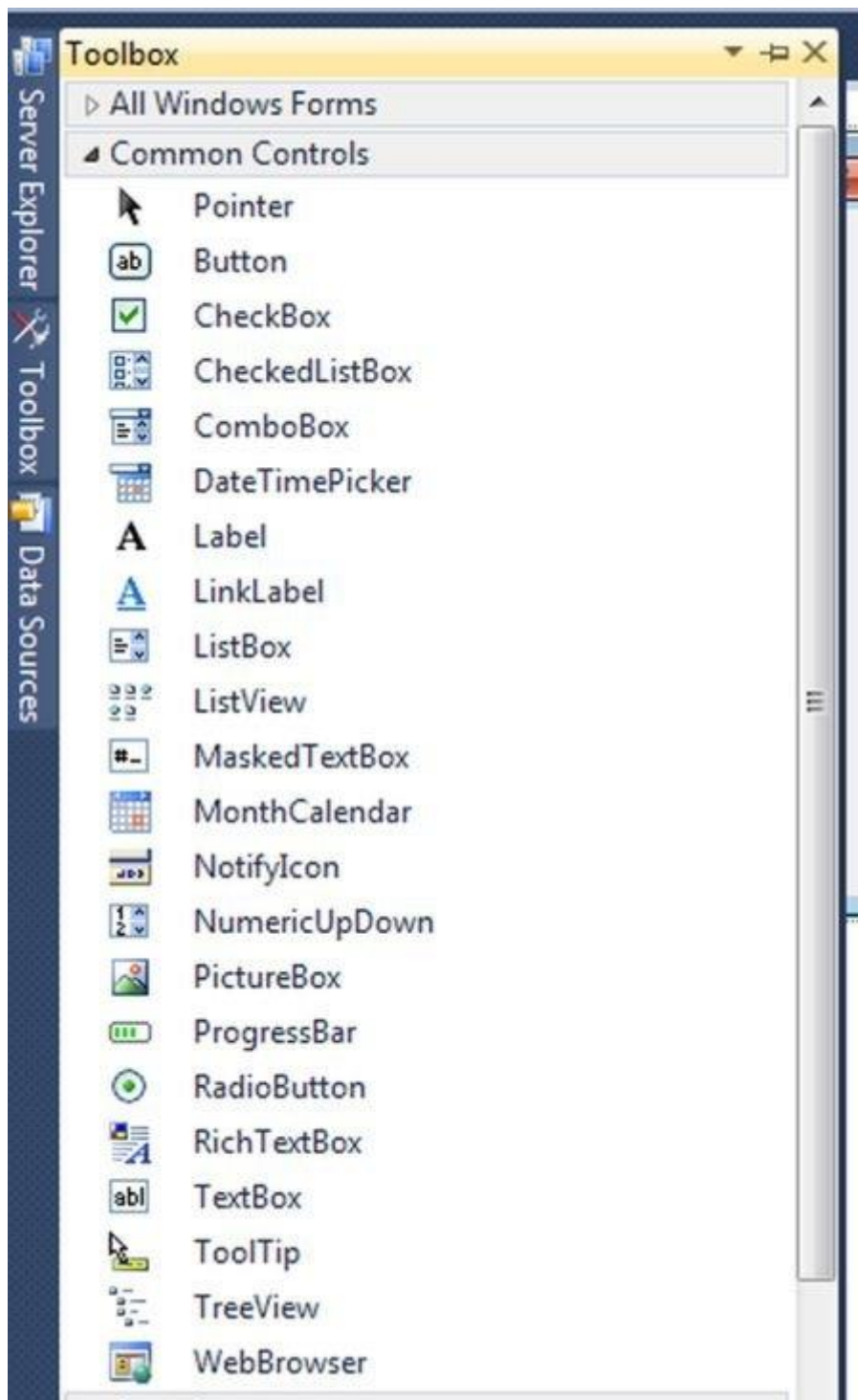
**Step1:** Open the Visual Studio 2010, click on new project.



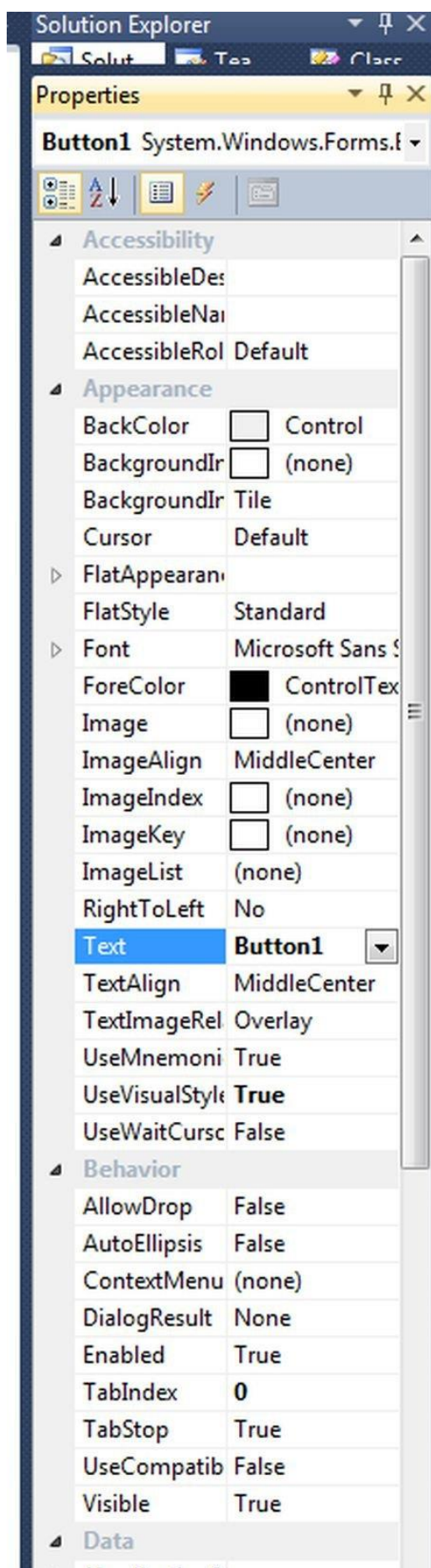
**Step2:** Click on VB and select the windows category, a page will open with a form.



**Step3:** Now open a toolbox and controls on the form i.e 5LabelBox, 4TextBox,2Button.



**Step4:** Edit the Buttons by Calculate and Exit by clicking on the properties of it.



**Step5:** Double click on the Buttons, it generates default code in which you can enter the rest of the code.

Coding for the Event is shown below.

#### Public Class Total

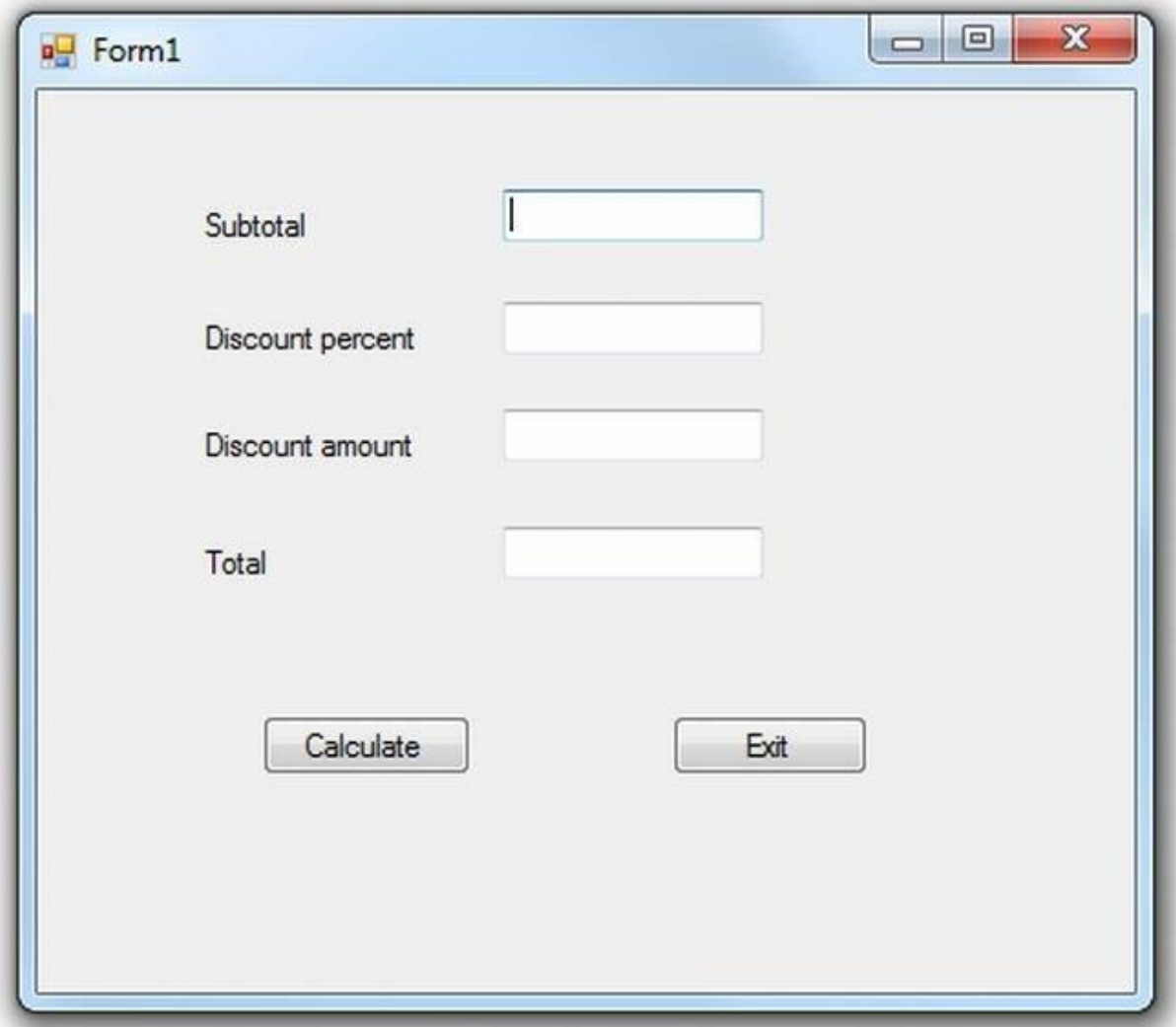
```
Private Sub Button1_Calculate(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    Dim discountPercent As Decimal
    If TextSubtotal.Text >= 400 Then
        discountPercent = 0.4
        Label5.Text = "<400 will be accept"
    Else
        If TextSubtotal.Text >= 200 And TextSubtotal.Text < 400 Then
            discountPercent = 0.3
        Else
            If TextSubtotal.Text >= 100 And TextSubtotal.Text < 200 Then
                discountPercent = 0.2
                Label5.Text = ""
            Else
                discountPercent = 0
            End If
        End If

        Dim discountAmount As Decimal = TextSubtotal.Text * discountPercent
        Dim Total1 As Decimal = TextSubtotal.Text - discountAmount
        TextDiscountpercent.Text = FormatPercent(discountPercent, 1)
        TextDiscountamount.Text = FormatNumber(discountAmount)
        TextTotal.Text = FormatNumber(Total1)
        TextSubtotal.Select()

    End If
End If
End Sub

Private Sub Button2_Exit(sender As System.Object, e As System.EventArgs) Handles Button2.Click
    Me.Close()
End Sub
End Class
```

Press f5 for the output you will see the first page like this.



The image shows a screenshot of a Windows application window titled "Form1". The window has a standard Windows XP-style title bar with minimize, maximize, and close buttons. The main area of the form is light gray and contains four labels on the left, each followed by a text input field on the right:

- Subtotal
- Discount percent
- Discount amount
- Total

At the bottom of the form, there are two buttons: "Calculate" and "Exit". The "Subtotal" input field contains a single vertical line cursor. The other input fields are empty.

When you enter the value more than 400 it will not accept.

The image shows a Windows application window titled "Form1". Inside the window, there are four labels on the left: "Subtotal", "Discount percent", "Discount amount", and "Total". To the right of each label is a text input field. The "Subtotal" field contains the number "500". To the right of the "Subtotal" field, there is a red text label that says "<400 will be accept". At the bottom of the form, there are two buttons: "Calculate" and "Exit".

When you enter the value less than 400, it will accept the value and will show all values.

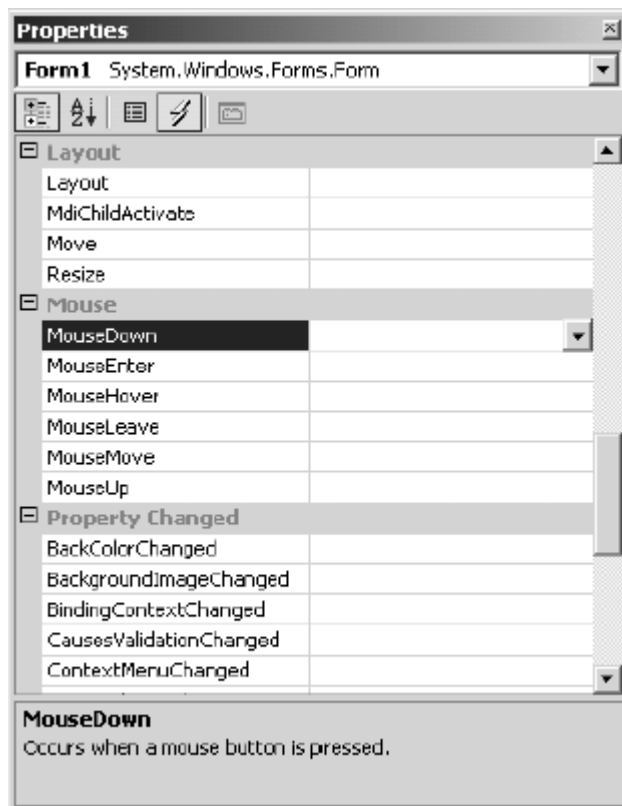


Form1

Subtotal	<input type="text" value="150"/>
Discount percent	<input type="text" value="20.0%"/>
Discount amount	<input type="text" value="30.00"/>
Total	<input type="text" value="120.00"/>

## Handling Mouse Events

The window in Figure 9.18 lists multiple mouse events. Event handlers can be generated simply by double-clicking the desired event.



The preceding code results in the `OnMouseDown` method being called if a mouse button is pressed and the `OnMouseMove` method being called whenever the mouse moves over the control, which in this case is the form.

To carry out some action after the mouse event occurs, we need to write the event handlers. The second parameter of the event handler method is a `System.Windows.Forms.MouseEventArgs` object, which details the mouse's state.

### Listing 9.7: Mouse Event Handlers

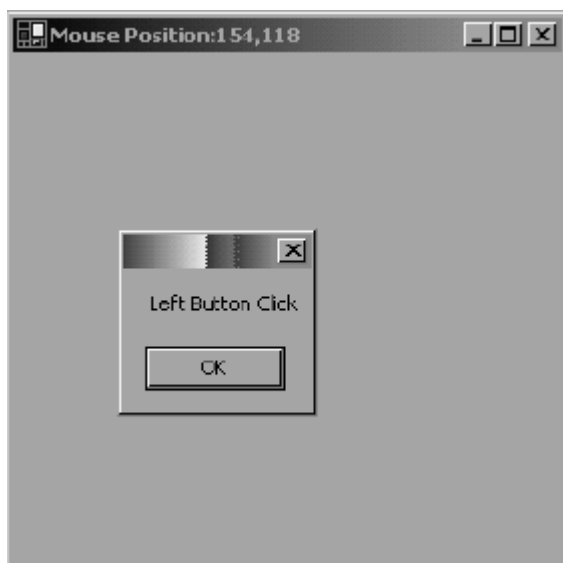
```
Public Sub OnMouseDown(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs)
    Select Case e.Button
        Case MouseButton.Left
            MessageBox.Show(Me, "Left Button Click")
        Exit Select
        Case MouseButton.Right
            MessageBox.Show(Me, "Right Button Click")
        Exit Select
    End Select
End Sub
```

```

Case MouseButton.Middle
    Exit Select
Case Else
    Exit Select
End Select
End Sub
Private Sub OnMouseMove(ByVal sender As Object, ByVal e As System.Windows.Forms.MouseEventArgs)
    Me.Text = "Mouse Position:" & e.X.ToString() & "," & e.Y.ToString()
End Sub

```

Figure 9.19 shows the output of Listing 9.7. A mouse click displays, in a message box, the mouse button clicked, while a mouse move shows the mouse's coordinates as the title of the form.



## Conclusion

Hope this article would have helped you in understanding Event Handling in windows programming using VB.NET.

## Writing code

The Code Editor As we saw in the previous lesson, getting to the Code Editor is as simple as hitting the proper button. You may have discovered that you can also call-up the Editor by double-clicking on an object. It is also possible to select "View code" with the right mouse button.

Codes must be written within chosen events

## Control Events

### Form Load event example

An event is a signal that informs an application that something important has occurred. For example, when a user clicks a control on a form, the form can raise a **Click** event and call a procedure that handles the event. There are various types of events associated with a Form like click, double click, close, load, resize, etc.

Following is the default structure of a form **Load** event handler subroutine. You can see this code by double clicking the code which will give you a complete list of the all events associated with Form control –

```
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    'event handler code goes here
End Sub
```

Here, **Handles MyBase.Load** indicates that **Form1\_Load()** subroutine handles **Load** event. Similar way, you can check stub code for click, double click. If you want to initialize some variables like properties, etc., then you will keep such code inside Form1\_Load() subroutine. Here, important point to note is the name of the event handler, which is by default Form1\_Load, but you can change this name based on your naming convention you use in your application programming.

### **Button Click Event Example**

Let's create a program to display a message on the Windows Forms using the button control in VB.NET. Public Class Button Control

```
Private Sub Button1_Click (ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles Button1.Click
    MessageBox.Show("Hello, World")
End Sub
```

### **ANOTHER EXAMPLE**

```
Private Sub Button_Control_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    Button1.Text = "Click Me" ' Set the name of button
End Sub
```

```
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
    MsgBox(" Visit here: https://www.javatpoint.com")
    ' Display the message, when a user clicks on Click me button
End Sub
```

```
End Class
```

### **TextBox GotFocus Event Example**

```
void textBox1_GotFocus(object sender, System.EventArgs e)
{
    MessageBox("cant be empty")
}
```

## SETTING PROPERTIES AT RUNTIME AND DESIGN TIME

Properties can be set at design time by using the Properties window or at run time by using statements in the program code.

```
Object.Property = Value
```

For example,

```
Form1.Caption = "Hello"
```

You can set any of the form properties using Properties Window. Most of the properties can be set or read during application execution. You can refer to Microsoft documentation for a complete list of properties associated with different controls and restrictions applied to them.

Let's create a program that displays the login details.

This program will set the Labels, Textbox and Button Properties at Runtime. When the Form is loaded the properties will be set .

### JavatPoint1.vb

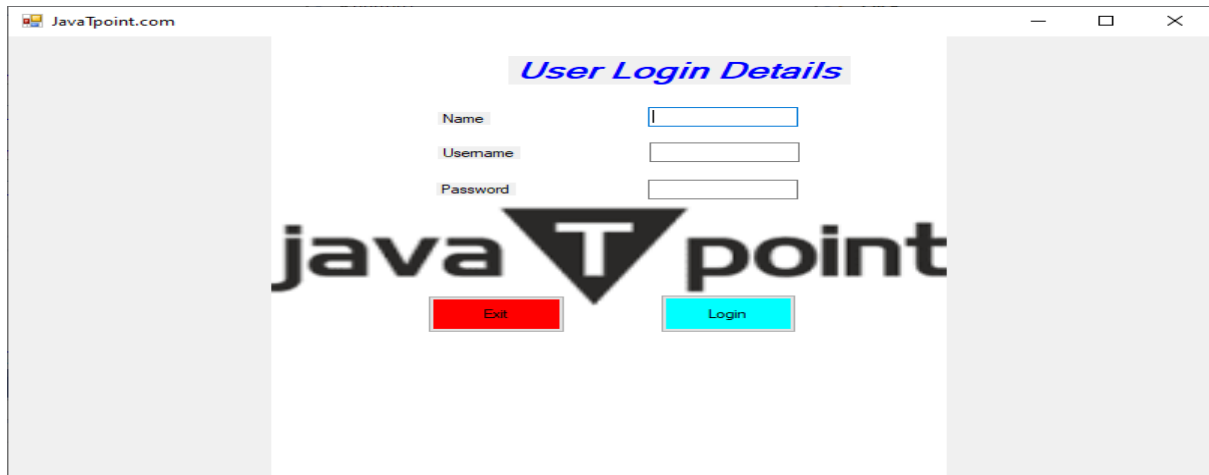
```
Public Class JavatPoint1
    Private Sub JavatPoint1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
        Me.Text = "JavaTpoint.com" ' title name
        Label1.Text = "User Login Details" ' Set the title name for Label1
        Label2.Text = "Name" ' Set the name for label2
        Label3.Text = "Username" ' Set the username for label2
        Label4.Text = "Password" ' Set the label name Password
        Text3.PasswordChar = "*"
        Button1.Text = "Login" ' Set the name of Button1 as Login
        Button2.Text = "Exit" ' Set the name of Button2 As Exit
    End Sub

    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
        End ' terminate the program when the user clicks button 2
    End Sub

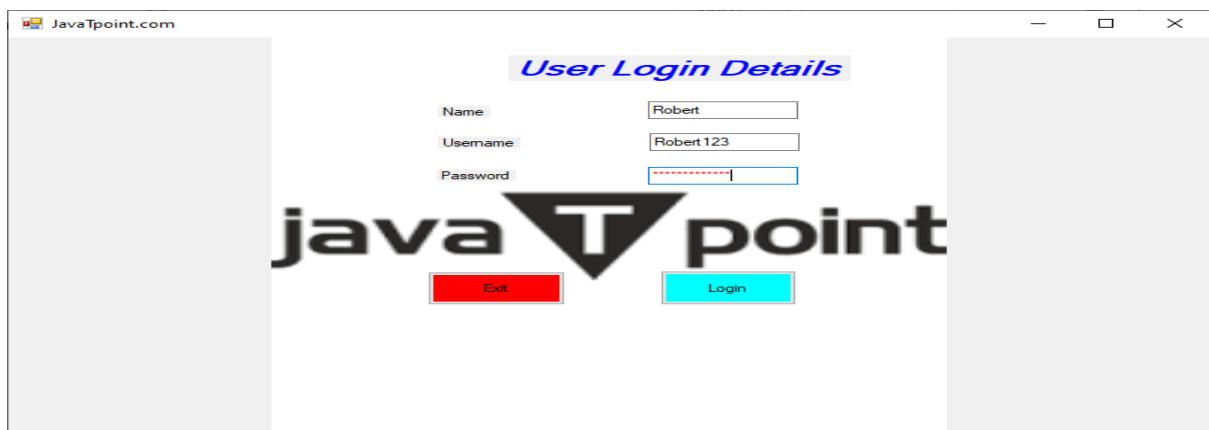
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim name As String
        Dim Uname As String
        Dim pass As String
        name = text1.Text
```

```
Uname = Text2.Text
pass = Text3.Text
' Display the user details, when the Button1 is clicked
MsgBox(" Your Name: " & name + vbCrLf + "Your UserName: " & Uname + vbCrLf + "Your Password: " & pass)
End Sub
End Class
```

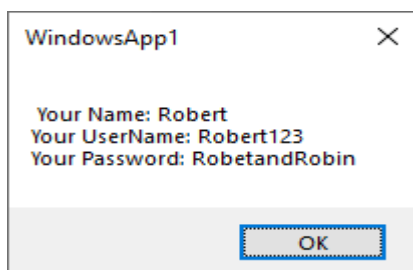
### Output:

A screenshot of a Windows application window titled "JavaTpoint.com". The window displays a form titled "User Login Details" in blue text. The form contains three input fields: "Name" (with a cursor), "Username", and "Password". Below the fields is a large "java T point" logo. At the bottom of the form are two buttons: a red "Exit" button and a cyan "Login" button.

Now enter all the details of the User Login form, it shows the following image, as shown below.

A screenshot of the same "User Login Details" form, but now with data entered. The "Name" field contains "Robert", the "Username" field contains "Robert123", and the "Password" field contains "RobetandRobin" (masked with red dots). The "Exit" and "Login" buttons remain at the bottom.

Now, click on the **Login** button. It shows all the details filled by the user in the form.

A screenshot of a Windows message box titled "WindowsApp1". The message box contains the following text: "Your Name: Robert", "Your UserName: Robert123", and "Your Password: RobetandRobin". At the bottom right of the message box is an "OK" button.

The Exit button in the form used to terminate the program.