❖ MDI Form

MDI stands for **Multiple Document Interface** applications that allow users to work with multiple documents by opening more than one document at a time. Whereas, a **Single Document Interface (SDI)** application can manipulate only one document at a time.

The MDI applications act as the parent and child relationship in a form. A parent form is a container that contains child forms, while child forms can be multiple to display different modules in a parent form.

VB.NET has folowing rules for creating a form as an MDI form.

1. **MidParent:** The MidParent property is used to set a parent form to a child form.
2. **ActiveMdiChild:** The ActiveMdiChild property is used to get the reference of the current child form.
3. **IsMdiContainer:** The IsMdiContainer property set a Boolean value to True that represents the creation of a form as an MDI form.
4. **LayoutMdi():** The LayoutMdi() method is used to arrange the child forms in the parent or main form.
5. **Controls:** It is used to get the reference of control from the child form.

Let's create a program to display the multiple windows in the VB.NET Windows Forms.

Step 1: First, we have to open the Windows form and create the Menu bar with the use of MenuStrip control, as shown below.

Step 2: After creating the Menu, add the Subitems into the Menu bar, as shown below.

In the above image, we have defined two Subitems, First is the **Feedback Form**, and the Second is VB.NET.

Step 3: In the third step, we will create two Forms: The Child Form of the **Main Form** or **Parent Form**.

Here, we have created the first Child Form with the name Form2.

Form2.vb

```
1. Public Class Form2
2.     Private Sub Form2_Load(sender As Object, e As EventArgs) Handles MyBase.Load
```

```
3.        Me.Text = "Feedback form" ' Set the title of the form
4.        Label1.Text = " Fill the Feedback form"
5.        Button1.Text = "Submit"
6.        Button1.BackColor = Color.SkyBlue
7.        Button2.Text = "Cancel"
8.        Button2.BackColor = Color.Red
9.     End Sub
10.    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
11.       MsgBox(" Successfully submit the feedback form")
12.    End Sub
13.    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
14.       Me.Dispose() ' end the form2
15.    End Sub
16. End Class
```

Another **Child Form** with the name **Form3**.

Form3.vb

```
1.  Public Class Form3
2.     Private Sub Form3_Load(sender As Object, e As EventArgs) Handles MyBase.Load
3.        Label1.Text = "Welcome to JavaTpoint Tutorial Site"
4.        Label.BackColor = Color.Green
5.        Label2.Text = "This is the VB.NET Tutorial and we are learning the VB.NET MDI Form"
6.        Label2.BackColor = Color.SkyBlue
7.     End Sub
8.  End Class
```

Step 4: Now we write the programming code for the Main or Parent Form, and here is the code for our Main Form.

MDI_form.vb

```
1.  Public Class MDI_Form
2.     Private Sub MDI_Form_Load(sender As Object, e As EventArgs) Handles MyBase.Load
3.        IsMdiContainer = True 'Set the Boolean value to true to create the form as an MDI form
```

.

```
4.          Me.Text = "javatpoint.com" 'set the title of the form
5.          PictureBox1.Image = Image.FromFile("C:\Users\AMIT YADAV\Desktop\jtp2.png")
6.          PictureBox1.Height = 550
7.          PictureBox1.Width = 750
8.      End Sub
9.      Private Sub FeedbackFormToolStripMenuItem_Click(sender As Object, e As EventArgs) H
    andles FeedbackFormToolStripMenuItem.Click
10.         PictureBox1.Visible = False
11.         Dim fm2 As New Form2
12.         fm2.MdiParent = Me 'define the parent of form3, where Me represents the same form
13.         fm2.Show() 'Display the form3
14.     End Sub
15.     Private Sub VBNETToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles
    VBNETToolStripMenuItem.Click
16.         PictureBox1.Visible = False
17.         Dim fm3 As New Form3
18.         fm3.MdiParent = Me 'define the parent of form3, where Me represent the same form
19.         fm3.Show() 'Display the form3
20.     End Sub
21. End Class
```

❖ Differences between MDI and SDI

1. MDI stands for “Multiple Document Interface” while SDI stands for “Single Document Interface”.
2. One document per window is enforced in SDI while child windows per document are allowed in MDI.
3. MDI is a container control while SDI is not container control.
4. SDI contains one window only at a time but MDI contains multiple documents at a time appeared as child window.
5. MDI supports many interfaces means we can handle many applications at a time according to user’s requirement. But SDI supports one interface means you can handle only one application at a time.
6. For switching between documents MDI uses special interface inside the parent window while SDI uses Task Manager for that.
7. In MDI grouping is implemented naturally but in SDI grouping is possible through special window managers.
8. For maximizing all documents, parent window is maximized by MDI but in case of SDI, it is implemented through special code or window manager.

9. Switch focus to the specific document can be easily handled while in MDI but it is difficult to implement in SDI.

❖ MsgBox ( ) Function

The objective of MsgBox is to produce a pop-up message box and prompt the user to click on a command button before he /she can continues. This format is as follows:

yourMsg=MsgBox(Prompt, Style Value, Title)

The first argument, Prompt, will display the message in the message box. The Style Value will determine what type of command buttons appear on the message box, please refer to Table 12.1 for types of command button displayed. The Title argument will display the title of the message board.

Table 12.1: Style Values

Style Value	Named Constant	Buttons Displayed
0	vbOkOnly	Ok button
1	vbOkCancel	Ok and Cancel buttons
2	vbAbortRetryIgnore	Abort, Retry and Ignore buttons.
3	vbYesNoCancel	Yes, No and Cancel buttons
4	vbYesNo	Yes and No buttons
5	vbRetryCancel	Retry and Cancel buttons

We can use named constants in place of integers for the second argument to make the programs more readable. In fact, Visual Basic 2012 will automatically shows up a list of named constants where you can select one of them.

For example:

yourMsg=MsgBox("Click OK to Proceed", 1, "Startup Menu")

and

yourMsg=Msg("Click OK to Proceed". vbOkCancel,"Startup Menu")

are the same.

yourMsg is a variable that holds values that are returned by the MsgBox ( ) function. The values are determined by the type of buttons being clicked by the users. It has to be

declared as Integer data type in the procedure or in the general declaration section. Table 12.2 shows the values, the corresponding named constant and buttons.

Table 12.2 : Return Values and Command Buttons

Value	Named Constant	Button Clicked
1	vbOk	Ok button
2	vbCancel	Cancel button
3	vbAbort	Abort button
4	vbRetry	Retry button
5	vbIgnore	Ignore button
6	vbYes	Yes button
7	vbNo	No button

Example 12.1

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
        Dim testmsg As Integer
        testmsg = MsgBox("Click to test", 1, "Test message")
        If testmsg = 1 Then
                MessageBox.Show("You have clicked the OK button")
        Else
                MessageBox.Show("You have clicked the Cancel button")
        End If
End Sub
```

To make the message box looks more sophisticated, you can add an icon besides the message. There are four types of icons available in VB2012 as shown in Table 12.3

Table 12.3 Types of Icons

Value	Named Constant	Icon
16	vbCritical	
3	vbQuestion	
48	vbExclamation	
64	vbInformation	

Example 12.2

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
        Dim testMsg As Integer
        testMsg = MsgBox("Click to Test", vbYesNoCancel + vbExclamation, "Test
Message")
        If testMsg = 6 Then
                MessageBox.Show("You have clicked the yes button")
        ElseIf testMsg = 7 Then
                MessageBox.Show("You have clicked the NO button")
        Else
                MessageBox.Show("You have clicked the Cancel button")
        End If
End Sub
```

The first argument, Prompt, will display the message

❖ The InputBox( ) Function

An InputBox( ) function will display a message box where the user can enter a value or a message in the form of text. In VB2005, you can use the following format:

myMessage=InputBox(Prompt, Title, default_text, x-position, y-position)

myMessage is a variant data type but typically it is declared as string, which accept the message input by the users. The arguments are explained as follows:

Prompt - the message displayed normally as a question asked.

Title - The title of the Input Box.

default-text - The default text that appears in the input field where users can use it as his intended input or he may change to the message he wish to enter.

x-position and y-position - the position or tthe coordinates of the input box.

However, the format won't work in Visual Basic 2012 because InputBox is considered a namespace. So, you need to key in the full reference to the Inputbox namespace, which is

Microsoft.VisualBasic.InputBox(Prompt, Title, default_text, x-position, y-position)

The parameters remain the same.

Example 12.3

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
        Dim userMsg As String
        userMsg = Microsoft.VisualBasic.InputBox("What is your message?", "Message
Entry Form", "Enter your messge here", 500, 700)
        If userMsg <> "" Then
                MessageBox.Show(userMsg)
        Else
                MessageBox.Show("No Message")
        End If
End Sub
```

The inputbox will appear as shown in the figure below when you press the command button