

CHAPTER :5

Menus and dialog boxes

- **Menus and Dialog boxes**
 - ❖ Drop down menu
 - ❖ Pop-up menu
 - ❖ Common Dialog boxes
 - ❖ Message Box and Input Box

Menus

- A list of commands or options from which you can choose
- a set of options presented to the user of a computer application to help the user find information or execute a program function.
- Menus can be added to a form using the Menu Editor(menustrip), selected from the "Tools" menu or from the button on the toolbar.

Cont.,

- Selecting a menu item at design-time takes you to the code editor for the specific menu item.
- The three-letter mnemonic for a menu item is `mnu` (eg, `mnuFile`).
- For example, a menu name for a File menu might be `mnuFile`.
- The `&` operator may be used to create a shortcut to a menu name.
- Whichever letter is preceded by an `&` will appear underlined

Drop Down Menu

- In a graphical drop-down menu, clicking on an item (text word, button or icon) causes a list of new items to appear below that item.
- An example can be seen by clicking on one of the text words such as "File" or "Edit" in the horizontal list at the top of the screen in a Windows application.
- Clicking on an item in the menu executes the indicated function or generates another menu.
- In a variant of the drop-down menu called a fly-out, the list appears to the side of the clicked item.

Pop-up Menu

- A pop-up menu appears above the clicked item when the main list is at the bottom of the window or screen.
- A menu may occasionally appear with apparent spontaneity at a random location in a window.
- This type of menu, also known as a pop-up, is usually meant to forestall a potential problem or prevent a user from making a mistake.

Cont.,

- VB allows you to easily create a popup menu like those you see in Explorer when you right click.
- All you need to do is create a menu, containing all the items you want displayed on the popup menu.

Option available in menu script

- **MenuItem** - Creates a sub-menu (also known as a pull-right menu) which essentially pops up a new menu to the right of the currently displayed menu.
- **ComboBox** - Adds a ComboBox control to the menu. Although you have the option of adding a ComboBox to your menu, you should resist the urge to do so.
- The placing of a ComboBox in a menu is not considered to be good user interface design.

Cont..

- **Separator** - Places a separator after the last menu item to be added.
- Separators in menus are useful for distinguishing between groups of menu items.
- **TextBox** - Adds a TextBox control to the menu.
- As with the ComboBox, you should resist the temptation to add such a control to a menu as it violates GUI design convention

MsgBox () Function

- The objective of MsgBox is to produce a message box and prompt the user to click on a command button before he /she can continues.
- This message box 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 example given below for types of command button displayed.
- The Title argument will display the title of the message board

example

```
Dim userMsg As String
userMsg = InputBox("What is your message?",
    "Message Entry Form", "Enter your message here")
If (userMsg <> "") Then
    TextBox.Text = userMsg
Else
    TextBox.Text = "No Message"
End If
```

Common Dialog boxes

- The Common Dialog Control provides a standard interface for operations such as opening, saving, and printing files or selecting colours and fonts using the Microsoft Windows dynamic link library `COMMDDL.DLL`.
- The Control is visible on the form as an icon at design-time but not at run-time.

Cont.,

- The icon cannot be resized on the form and it's not possible to specify where the Dialog will be displayed when opened.
- To use a Common Dialog Control, add "Microsoft Common Dialog Control" from the "Project", "Components" menu item.
- The three-letter mnemonic for a Common Dialog Control is "cdl".
- The Control's Methods are used to determine which dialog is displayed.

ColorDialog

- The ColorDialog control class represents a common dialog box that displays available colors along with controls that enable the user to define custom colors.
- It lets the user select a color.
- The main property of the ColorDialog control is *Color*, which returns a **Color** object.

example

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

```
Dim ColorDialog1 As New ColorDialog
```

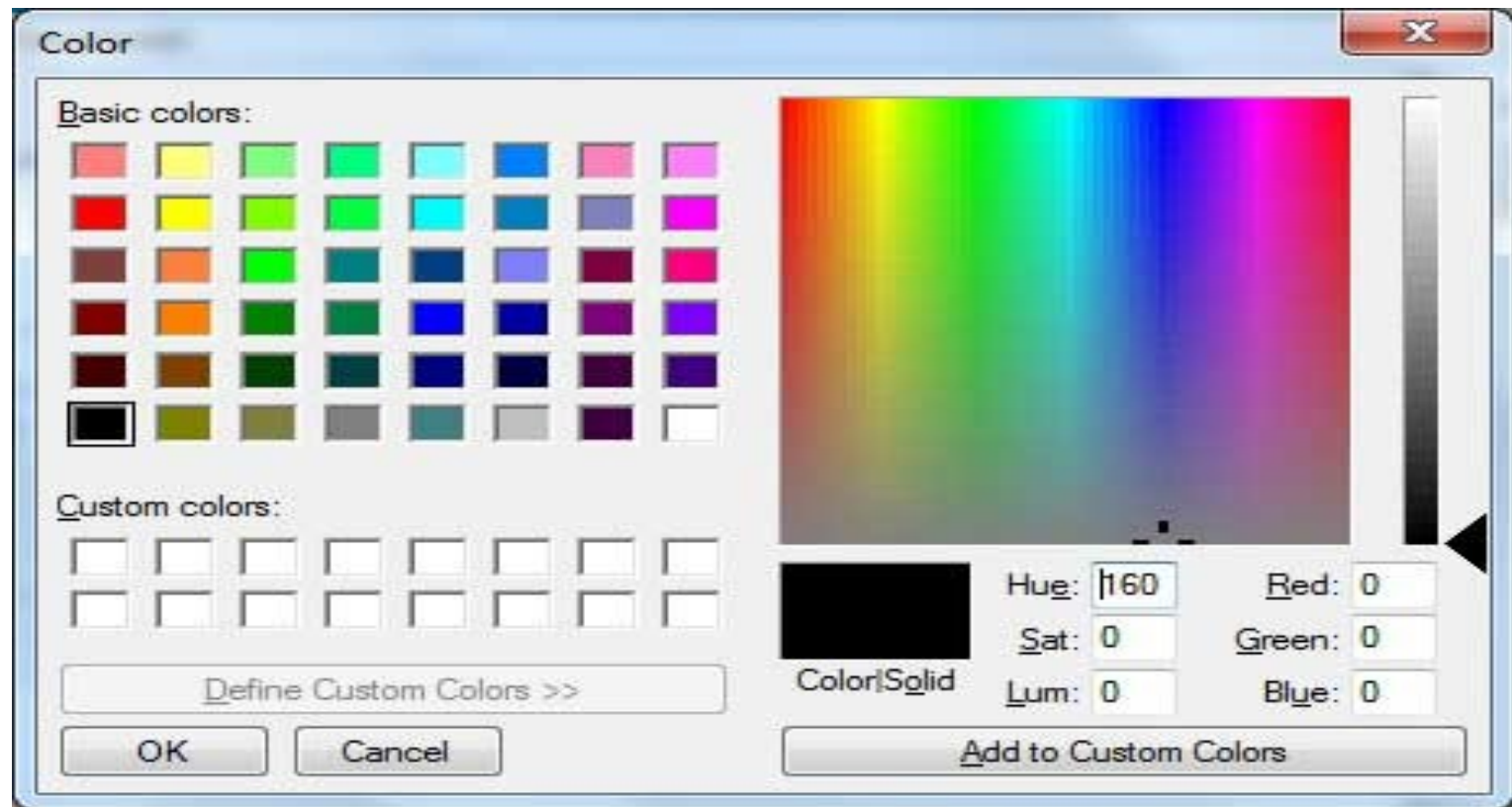
```
    If ColorDialog1.ShowDialog <> -  
Windows.Forms.DialogResult.Cancel Then
```

```
        Label1.ForeColor =  
ColorDialog1.Color
```

```
    End If
```

```
End Sub
```

...ColorDialog



FontDialog

- It prompts the user to choose a font from among those installed on the local computer and lets the user select the font, font size, and color.
- It returns the Font and Color objects.
- Following is the Font dialog box:

Example

```
Dim FontDialog1 As New FontDialog
If FontDialog1.ShowDialog <>
Windows.Forms.DialogResult.Cancel
Then
    RichTextBox1.ForeColor =
FontDialog1.Color
    RichTextBox1.Font =
FontDialog1.Font
End If
```

Cont.,



OpenFileDialog

- The **OpenFileDialog** control prompts the user to open a file and allows the user to select a file to open.
- The user can check if the file exists and then open it. The OpenFileDialog control class inherits from the abstract class **FileDialog**.
- If the ShowReadOnly property is set to True, then a read-only check box appears in the dialog box.
- You can also set the ReadOnlyChecked property to True, so that the read-only check box appears checked.
- Following is the Open File dialog box:

example

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

```
    If OpenFileDialog1.ShowDialog <>  
    Windows.Forms.DialogResult.Cancel  
    Then
```

```
        PictureBox1.Image =  
        Image.FromFile(OpenFileDialog1.FileName  
e)
```

```
    End If
```

```
End Sub
```

SaveFileDialog

- The **SaveFileDialog** control prompts the user to select a location for saving a file and allows the user to specify the name of the file to save data.
- The SaveFileDialog control class inherits from the abstract class FileDialog.

example

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

Dim SaveFileDialog1 As New SaveFileDialog

SaveFileDialog1.Filter = "TXT Files (*.txt)|*.txt"

If SaveFileDialog1.ShowDialog =
Windows.Forms.DialogResult.OK Then

My.Computer.FileSystem.WriteAllText _
(SaveFileDialog1.FileName, RichTextBox1.Text,
True)

End If

End Sub

Cont.,

- Write some text in the RichTextBox and click on the Save Comment button to save the text as a text file in your computer.

PrintDialog

- The PrintDialog control lets the user to print documents by selecting a printer and choosing which sections of the document to print from a Windows Forms application.
- There are various other controls related to printing of documents.
- Let us have a brief look at these controls and their purpose.
- These other controls are:

The **PrintDocument** control

- it provides support for actual events and operations of printing in Visual Basic and sets the properties for printing.

Cont.,

The **PrinterSettings** control

- it is used to configure how a document is printed by specifying the printer.

The **PageSetUpDialog** control

- it allows the user to specify page-related print settings including page orientation, paper size and margin size.

Cont.,

The **PrintPreviewControl** control

- it represents the raw preview part of print previewing from a Windows Forms application, without any dialog boxes or buttons.

The **PrintPreviewDialog** control

- it represents a dialog box form that contains a PrintPreviewControl for printing from a Windows Forms application.