BCA Semester-5
US05CBCA01-Visual Programming through VB.NET

UNIT-3 VB.NET Controls and Error Handling

Button

Button is most widely used control in VB applications. It used to perform a task or take an action when the user clicks on it, generally on completion of input.

Important properties

No.	Property	Use
1	Text	Text to be displayed on the button.
2	TextAlign	Alignment of text on the button.
3	BackColor	Background color of the button.
4	ForeColor	Color of the text to be displayed on the button.
5	Image	Image to be displayed on the button
6	ImageAlign	Alignment of the image
7	UseMnemonic	Whether the button can be pressed using Alt+Key
8	FlatStyle	Appearance of the button
9	Enabled	Whether the button can be pressed or not.
10	Visible	Whether the button is visible or not.
11	AcceptButton	Whether the button can be pressed on pressing the Enter key on the keyboard.
12	CancelButton	Whether the button can be pressed on pressing the E key on the keyboard.

Important Event

Buttons provide the most popular way of creating and handling an event in your code-every Visual Basic Programmer is familiar with the button Click event. Buttons can be clicked with the mouse or with the Enter key if the button has the focus.

TextBox

TextBox is also a widely used control, just like Button, in VB applications. It is mainly used to take input from the user. However, it is also sometimes used to display the output.

Important properties

No.	Property	Use
1	Text	Text in the TextBox.
2	ReadOnly	If True, text in the textbox can not be changed, but can only be selected.
3	Enabled	Enables/Disables the textbox. You can not select the text if the TextBox is disabled.
4	BackColor	Background color of the button.
5	ForeColor	Color of the text to be displayed on the button.
6	Multiline	Allows multiple lines of text in the TextBox.
7	PasswordChar	Sets character to be displayed instead of actual text, when the TextBox is used to take password from the user.
8	MaxLength	Maximum no. of character that can be entered in the textbox. Default:32767
9	RightToLeft	If set Yes, allows text to move from right to left. This can be used in calculator/for Arabic language. Default: No
10	AutoCompleteCustomSource	Is used to provide list of suggestions.

11	AutoCompleteMode	Determines AutoComplete behavior.
12	AutoCompleteSource	Specifies Source of suggestions.

Important Methods

No.	Method	Use
1	Focus()	Gives focus (i.e. places cursor in) the TextBox
2	Clear()	Removes entire text.
3	Append()	Appends to existing contents.

Important Events

No.	Event	Use
1	GotFocus()	This event triggers when the TextBox receives focus.
2	LostFocus()	This event triggers when the TextBox loses focus.
3	TextChanged()	This event triggers when the text in the TextBox is changed.
4	KeyPress()	This event triggers when a key pressed on the keyboard and TextBox has the focus.

Label

Label is a frequently used control, mostly with TextBox, in VB applications. It is mainly used to display some information or mention type of what kind of information which is expected in the following TextBox.

Important properties

No.	Property	Use
1	Text	Text to be displayed on the label.
2	TextAlign	Alignment of text within the label.
3	BackColor	Background color of the button.
4	ForeColor	Color of the text to be displayed on the button.
5	BorderStyle	Type of border around the label.

Important Methods

No.	Method	Use
1	Hide()	Hides the label.
2	Show()	Shows the label.

Important Events

Label is mostly used as a passive control, just to display some text. However, it reacts to mouse click and mouse hover.

No.	Event	Use
1	Click()	This event triggers when mouse is clicked on the label.
2	MouseHover()	This event triggers when mouse pointer is placed on the label.

RadioButton/CheckBox

A RadioButton is a control which is circular in shape and toggle a dot in a circle. It allows the user to select any one from the available options. Therefore, it is used to specify choice in situations where only one of the choice is possible. For example, gender, relationship status, category etc.

A CheckBox is a control which is square in shape and toggle a tick mark in a square. It allows the user to select one or more from the available options. So, it is used to specify multiple selections like languages known, hobbies and interests, skills etc.

Following properties are common to RadioButton and CheckBox:

Important properties

No.	Property	Use
1	Text	Text to be displayed on the control
2	Checked	Specifies if the radio button is selected or not
3	Appearance	Normal or toggle button
4	CheckAlign	Location of radio button with respect to text
5	Image	Image to be displayed next to control

Important Methods

No.	Method	Use
1	Hide()	Hides the control.
2	Show()	Shows the control.

Important Events

No.	Event	Use
1	CheckedChanged()	Occurs first of all, when the Checked property changes either by code or by user interaction
2	CheckedStateChanged()	Occurs after CheckedChanged event, when the Checked property changes either by code or by user interaction.
3	Click()	Occurs last, when the Checked property changes only by user interaction.

GroupBox

Group boxes are used to provide a grouping for other controls. Group boxes display frames around their contained controls and can display text in a caption.

The most common use of group box is to create a group of radio buttons. When you group radio buttons using group box, each set works independently. This means, one radio button can be selected from each group.

Important properties

No.	Property	Use
1	Text	Title of the group box.
2	Controls	Collection of controls which are placed in the group box.

Important Methods

No.	Method	Use
1	Hide()	Hides the control.
2	Show()	Shows the control.

Important Events

No.	Event	Use
1	Enter()	Occurs when any control within the group receives the focus.

ListBox

List box displays a list of items from which the user can select one or more. If there are too many items to display at once, a scroll bar automatically appears to let the user scroll through the list.

Important properties

No.	Property	Use
1	Items	Collection of items to be displayed in the ListBox.
2	SelectionMode	None-No selection at all is allowed. One- (Default) Only a single item can be selected. MultiSimple- Simple multiple selection: A mouse click (or pressing the spacebar) selects or deselects an item in the list. You must click all the items you want to select. MultiExtended-Extended multiple selection: Press Shift and click the mouse (or press one of the arrow keys) to expand the selection. This will highlight all the items between the previously selected item and the current selection. Press Ctrl and click the mouse to select or deselect.
3	SelectedItem	Returns the item which is selected.
4	SelectedItems	Returns all items which are selected. This is applicable in case multiple selection is enabled.
5	SelectedIndex	Returns index of selected item, -1 no item is selected.
6	SelectedIndices	Returns indices of all selected items.
7	MultiColumn	If true, displays items in multiple columns.
8	Sorted	If true, items in the ListBox are sorted.

Important Methods

No.	Method	Use
1	SetSelected(index, T/F)	If second argument is true, selects the item at position specified by index,
2	GetSelected(index)	Returns True if item at specified index is selected, False otherwise.
3	ClearSelected()	Clears selection of (all items) ListBox.
4	FindString(str)	Returns index at which match is found, -1 otherwise.

Important Events

No.	Event	Use
1	SelectedIndexChanged()	Occurs whenever an item is selected/de-selected thorugh mouse/keyboard/code.

ComboBox

Combo box is a combination of TextBox and ListBox. It is used to display data in a drop-down list. The combo box is made up of two parts: The top part is a text box that allows the user to type in all or part of a list item. The other part is a list box that displays a list of items from which the user can select one or more. You can allow the user to select an item from the list, or enter their own data.

Important properties

No.	Property	Use
1	Items	Collection of items to be displayed in the ComboBox.
2	DropDownStyle	DropDown: (Default) The control is made up of a drop-down list and a text box. The user can select an item from the list or type a new one in the text box. DropDownList: This style is a drop-down list, from which the user can select one of its items but can't enter a new one. Simple: The control includes a text box and a list that doesn't drop down. The user can select from the list using arrow keys or type in the text box.

		The DropDown and Simple ComboBox controls allow the user to select an item from the list or enter a new one in the edit box of the control. The DropDownList ComboBox is similar to a ListBox control in the sense that it restricts the user to selecting an item. The user can not type in a new item.
3	Text	Returns selected/typed value
4	Sorted	If true, items in the ComboBox are sorted.

Important Methods

No.	Method	Use
1	BeginUpdate()	Turns off visual updating of the combo box until the EndUpdate method is called.
2	EndUpdate()	Resumes visual updating of the combo box.
3	FindString(str)	Returns index at which match is found, -1 otherwise.
4	GetItemText()	Gets an item's text.

Important Events

No.	Event	Use
1	TextChanged()	This event occurs FIRST OF ALL when either the user types a text or clicks to select an item
2	SelectedValueChanged()	This event occurs BEFORE SelectedIndexChanged when the user click on one of the item in the list to select it.
3	SelectedIndexChanged()	This event occurs AFTER SelectedValueChanged when the user click on one of the item in the list to select it.

CheckedListBox

CheckedListBox is similar to a ListBox, but it provides a check box before each item in list.

Important properties

No.	Property	Use
1	Items	Collection of items to be displayed in the CheckedListBox.
2	CheckOnClick	If true, Single-click should select an item. By default, a double-click select an item.
3	CheckedItems	Holds collection of items selected.
4	Sorted	If true, items in the ComboBox are sorted.

Important Methods

No.	Method	Use
1	GetItemChecked(index)	Returns true if item at specified index is checked, false otherwise.
2	SetItemChecked(index, True/False)	Sets/Clears the check mark of item at specified index.

Important Events

No.	Event	Use
1	ItemCheck()	When the check in front of an item in checked list box changes, an ItemCheck event occurs.
2	SelectedIndexChanged()	This event occurs when the user clicks on one of the item in the list to select/deselect it.

Timer

Timer is not a control, it is a component. It is used to perform a task at every specific interval of time. When a Timer is added to the form, it does not appear on the form canvas, but it appear in the component tray, located just below the form window. It is also not visible at runtime.

Important properties

No.	Property	Use
1	Enabled	True enables the timer, particularly the tick() event of the timer. False disables the tick() event until Enabled property is set to True.
2	Interval	No. of milliseconds after which the tick() event is triggered.

Important Methods

No.	Method	Use
1	Start()	Starts the timer. Same as setting Enabled property to True.
2	Stop()	Stops the timer. Same as setting Enabled property to False.

Important Events

No.	Event	Use
1	Tick()	This event triggers after every milliseconds specified in Interval property of the timer.

PictureBox

Picture box is used to display graphics from a bitmap, icon, JPEG, GIF or other image file type.

Important properties

No.	Property	Use
1	Image	Image to display in the picture box. Image can be set at either design time or at runtime. PictureBox1.Image = Image.FromFile("<PATH TO IMAGE FILE>")
2	SizeMode	Size and position of the image. Options are: Normal/ StretchImage/ AutoSize/ CenterImage
3	ClientSize	Used at runtime. PictureBox1.ClientSize = New Size(width, height)

Important Methods

No.	Method	Use
1	Show()	Shows the picture box if it is hidden.
2	Hide()	Hides the picture box.

Important Events

No.	Event	Use
1	Click()	This event triggers on single click of mouse.
2	MouseHover()	This event triggers when the mouse pointer is placed on the picture box.

HScrollBar/VScrollBar

Scroll bars are—those vertical or horizontal controls that display a scroll box or thumb that you can manipulate, and when you drag it to a new position, the value of the scroll bar changes, causing a corresponding action in the program.

Important properties

No.	Property	Use
1	Value	Integer – Current value of the scrollbar.
2	Minimum	Minimum value which can be selected using the scrollbar. Default is zero.
3	Maximum	Maximum value which can be selected using the scrollbar. Default is 100. Actual Max = (Maximum+1) – LargeChange
4	SmallChange	Change in the value when the user clicks on arrows at the ends of the scrollbar. Default:1
5	LargeChange	Change in the value when the user clicks in the scrollbar outside the scroll button. Default: 100 It is actually the width of the scroll button.

Important Methods

No.	Method	Use
1	Show()	Shows the scroll bar box if it is hidden.
2	Hide()	Hides the scroll bar.

Important Events

No.	Event	Use
1	Scroll()	This event triggers when the scroll bar slides.
2	ValueChanged()	Occurs when the Value property has changed, either by a Scroll event or programmatically.

DateTimePicker

You can set a date and time in a date-time picker just by editing the displayed values in the control; if you click the arrow in the date-time picker, it displays a month calendar, just as a combo box would display a drop-down list; you can make selections just by clicking the calendar.

Important properties

No.	Property	Use
1	Value	Currently selected date. By default, system's date is returned.
2	MinDate	Minimum date that can be selected from the control. Default (and earliest value which can be assigned is) : 1/1/1753
3	MaxDate	Maximum date that can be selected from the control. Default (and latest value which can be assigned is) : 31/12/9998
4	Format	Options are: Long/Short/Time/Custom. If Custom is used, specify format in CustomFormat property of the control.
5	CustomFormat	Customized format like dd/mm/yyyy.

How to get more details about the DateTime selected from a DateTimePicker?

Dim dtm As **DateTime**
dtm = DateTimePicker1.Value

dtm.Day : Returns day number between 1 and 31.
dtm.Month : Returns month number between 1 and 12.
dtm.Year : Returns current year.

dtm.Hour : Returns hour from current time.
dtm.Minute : Returns minute from cuurent time.

Important Event

No.	Event	Use
1	ValueChanged()	When the selected value changes in the control.

TreeView

A TreeView control is used to display hierarchical relationship. Examples of hierarchy include file system of computers (Drive->Directories->Files), employees in an organization (Manager->Asst. Manager->Executive), Book (Chapters->Paragraphs->Sentences) etc.

A Treeview displays objects in the form of an upside down tree. Thus, the root i.e. the top-most object appears at the top or beginning of the tree. The root expands to other nodes, known as child nodes. These child nodes further expands to their child nodes and so on.

The following figure shows a Treeview control:

Important properties

No.	Property	Use
1	Nodes	Collection of nodes
2	CheckBoxes	Gets/sets whether checkboxes should be displayed next to tree nodes
3	FullRowSelect	Gets/sets whether a selection should select the whole width of the tree view.
4	HideSelection	Gets/sets whether the selected tree node stays highlighted when the tree view loses the focus.
5	LabelEdit	Gets/sets whether tree node text can be edited.
6	PathSeparator	Gets/sets the string the tree node uses as a path delimiter. Default is \.
7	Scrollable	Gets/sets whether the tree view should display scroll bars as needed.
8	SelectedNode	Gets/sets the node that is selected.
9	ShowLines	Gets/sets whether lines are drawn between tree nodes.
10	ShowPlusMinus	Gets/sets whether plus-sign (+) and minus-sign (-) buttons are shown next to tree nodes with child tree nodes.
11	ShowRootLines	Gets/sets whether lines should be drawn between the tree nodes and the root node.
12	Sorted	Gets/sets if the tree nodes should be sorted.
13	TopNode	Gets the first visible tree node.
14	VisibleCount	Gets the number of nodes that can be seen currently.

Important Methods

No.	Method	Use
1	CollapseAll	Collapses all nodes.
2	ExpandAll	Expands all the nodes.
3	GetNodeAt	Gets the node that is at the given location.
4	GetNodeCount	Gets the number of nodes.

Important Events

No.	Event	Use
1	AfterSelect	Occurs after a noode in the tree is selected.
2	AfterCheck	Occurs when a node checkbox is checked.
3	AfterLabelEdit	Occurs when a tree node label text is edited.

The default event is the AfterSelect event, which occurs after a node has been selected. This event is an event of the tree view control, not of the TreeNode object that was selected, but you can determine which node was selected with the TreeViewEvent Args object that is passed to you, because it has a Node property that holds the selected node. For example, here's how you display the text of a selected node in a text box:

```
Private Sub TreeView1_AfterSelect(ByVal sender As System.Object, _
ByVal e As System.Windows.Forms.TreeViewEventArgs) Handles _
TreeView1.AfterSelect
TextBox1.Text = "You clicked: " & e.Node.Text
End Sub
```

Using Checkboxes and AfterCheck event in Tree Views

You can make checkboxes appear in a tree view by setting the tree view's CheckBoxes property to True. To display which checkboxes have been checked or unchecked you can use the following code:

```
Private Sub TreeView1_AfterCheck(ByVal sender As Object, ByVal e As
System.Windows.Forms.TreeViewEventArgs) Handles TreeView1.AfterCheck

If e.Node.Checked Then
TextBox1.Text = "You checked: " & e.Node.Text
Else
TextBox1.Text = "You unchecked: " & e.Node.Text
End If

End Sub
```

Exception or Error Handling

Exceptions occur when a program is running (as opposed to syntax errors, which will prevent VB .NET from running your program at all). You can trap such exceptions and recover from them, rather than letting them bring your program to an end.

Exception handling is really runtime error handling in VB .NET (although other languages make a distinction between exceptions and errors).

There are two ways to handle such exceptions: structured and unstructured exception handling.
Structured exception handling uses the same Try…Catch…Finally type of construct that Java does; unstructured exception handling is uses the On Error GoTo statement.

Unstructured Exception Handling

The old error-handling mechanism in VB6 and older versions is now called unstructured exception handling, and it uses **On Error Goto** statement. You use this statement to tell VB .NET where to transfer control to **in case there's been an exception**, as in this case, where I'm telling Visual Basic to jump to the label "Handler" if there's been an exception.

On Error { GoTo [line | 0 | -1] | Resume Next }

Here are the parts of this statement:
GoTo line—Enables the exception-handling code that starts at the line specified in the required line argument. The line argument is any line label or line number. If an exception occurs, program execution

goes to the given location. (Note that the specified line must be in the same procedure as the On Error statement.)

GoTo 0—Disables enabled exception handler in the current procedure and resets it to Nothing.

GoTo -1—Same as GoTo 0.

Resume Next—Specifies that when an exception occurs, execution skips over the statement that caused the problem and goes to the statement immediately following. Execution continues from that point.

You create labels in your code with the label name followed by a colon, and the exception-handling code will follow that label (note that I've added an Exit Sub statement to make sure the code in the exception handler is not executed by mistake as part of normal program execution):

```
Module Module1
Sub Main()
On Error Goto Handler
Exit Sub
Handler:
MsgBox ("Exception occurred!")

End Sub
End Module
```

Now I can execute some code that may cause an exception, as here, where the code performs a division by zero, which causes an exception. When the exception occurs, control will jump to the exception handler, where I'll display a message and then use the **Resume Next** statement to transfer control back to the statement immediately after the statement that caused the exception:

```
Module Module1
Sub Main()

Dim int1 = 0, int2 = 1, int3 As Integer

On Error Goto Handler
int3 = int2 / int1
System.Console.WriteLine("The answer is {0}", int3)
Handler:
System.Console.WriteLine("Divide by zero error")
Resume Next

End Sub

End Module
```

When you run this code, you see this message: `Divide by zero error`

You can also use an **On Error Resume Next** or **On Error Resume line** statement to make Visual Basic continue program execution after an exception has occurred. This form is sometimes preferable to the On Error GoTo form if you don't want to write an explicit exception handler:

Structured Exception Handling

The unstructured exception handling approach suffers from two limitations: using On Error statement just sets the internal exception handler in Visual Basic; it certainly doesn't add any structure to your code, and if your code extends over procedures and blocks, it can be hard to figure out what exception handler is working when.

Visual Basic also supports structured exception handling. In particular, Visual Basic uses an enhanced version of the Try…Catch…Finally syntax already supported by other languages, such as Java.

Here's an example that follows our previous example handling a division by zero exception; I start by creating a Try block—**you put the exception-prone code in the Try section and the exception-handling code in the Catch section.** After the rest of the statement finishes, execution is always passed to the **Finally** block, if there is one.

```
Module Module1
Sub Main()
Try
Catch e As Exception
End Try
End Sub
End Module
```

Note the syntax of the Catch statement, which catches an Exception object that I'm naming e. When the code in the Try block causes an exception, I can use the e.ToString method to display a message:

```
Module Module1
Sub Main()
Dim int1 = 0, int2 = 1, int3 As Integer
Try
int3 = int2 / int1
System.Console.WriteLine("The answer is {0}", int3)
Catch e As Exception
System.Console.WriteLine(e.ToString)
End Try
End Sub
End Module
```

Here's what you see when you run this code:

```
System.OverflowException: Exception of type System.OverflowException was thrown.
at Microsoft.VisualBasic.Helpers.IntegerType.FromObject(Object Value)
at ConsoleHello.Module1.Main() in
C:\vbnet\ConsoleHello\Module1.vb:line 5
```

Besides using the e.ToString method, you can also use the e.message field, which contains this message:

```
Exception of type System.OverflowException was thrown.
```

The code in the Finally block, if there is one, is always executed in a Try…Catch…Finally statement, even if there was no exception, and even if you execute an Exit Try statement. This allows you to deallocate resources and so on.

LinkLabel

Link labels are new in VB .NET. They're based on the Label class, but also let you support Web-style hyperlinks to the Internet and other Windows forms. In other words, you can use a link label control for everything that you can use a label control for, and you can also make part of the text in this control a link to a Visual Basic object or Web page.

Besides functioning as a full label control, you can display multiple hyperlinks in a single link label control. Each hyperlink is an object of the LinkLabel.Link class and is stored in a collection called Links.

Important properties

No.	Property	Use
1	Links	Collection of Links in the LinkLabel control.
2	LinkArea	Sets/Gets the range in the text to treat as a link.

3	LinkColor	Sets/Gets the color for a normal link.
4	ActiveLinkColor	Sets/Gets the color for an active link.
5	DisabledLinkColor	Sets/Gets the color for a disabled link.
6	VisitedLinkColor	Sets/Gets the color for links that have been visited.

Important Events

No.	Event	Use
1	LinkClicked()	This event triggers when a link is clicked inside the link label.

To navigate to a particular URL on web, System.Diagnostics.Process.Start( ) method is used with URL as a string argument. For example, to redirect to www.google.com, you write:

System.Diagnostics.Process.Start("www.google.com")

To create multiple hyperlinks within a single LinkLabel control, use Add() method of Links collection.

```
LinkLabel1.Text = "You can visit our website1 or website2"
LinkLabel1.Links.Add(17,8,"web1")
' w in website1 is at position 17 in the Text of LinkLabel1, 8 is the length of the word website1
LinkLabel1.Links.Add(28,8,"web1")
 ' w in website1 is at position 28 in the Text of LinkLabel1, 8 is the length of the word website2
```

In the LinkClicked() event of the link label, you must identify the exact link which is clicked. It can be done as follows:

```
If (e.Link.LinkData.ToString() = "web1") Then
        System.Diagnostics.Process.Start("URL1")
End If

If (e.Link.LinkData.ToString() = "web2") Then
        System.Diagnostics.Process.Start("URL2")
End If
```

RichTextBox

RichTextBox control is used for displaying, entering, and manipulating rich text with formatting. The RichTextBox control does everything the TextBox control does, but in addition, it can display fonts, colors, and links; load text and embedded images from a file; undo and redo editing operations; and find specified characters.

Rich text format (RTF) text supports a variety of formats. For example, you can color text in a rich text box, underline it, bold it, or make it italic. You can select fonts and fonts sizes, as well as write the text out to disk file or read it back in. Rich text boxes also can hold a great amount of data, unlike standard text boxes.

Important properties

No.	Property	Use
1	Rtf	Sets/Gets the text of the RTB including all formatting information.
2	Text	Sets/Gets the text of the RTB excluding all formatting information.
3	SelectedRtf	Sets/Gets the selected text wthin RTB
4	SelectionLength	Sets/Gets the no. of characters selected
5	SelectionStart	Sets/Gets the starting point of text selected
6	WordWrap	Whether a multiline RTB control automatically wraps Words.
7	DetectUrls	Whether an RTB should detect URLs when typed into RTB.
8	Modified	Whether the RTB control has been modified.
9	Multiline	Whether the RTB should have more than one line.

Important Methods

No.	Method	Use
1	AppendText()	Appends text at the end of existing text
2	Clear()	Removes all text from textbox & makes it empty
3	Find()	Searches for text within the contents of the RTB
4	LoadFile()	Loads contents of a file into the RTB
5	SaveFile()	Saves the contents of RTF to a file
6	Select()	Selects specified part of text
7	SelectAll()	Selects entire text

Important Events

No.	Event	Use
1	LinkClicked()	Occurs when the user clicks on a link within the text of the RichTextBox control.
2	ModifiedChanged()	Occurs when the value of the Modified property is changed.

ColorDialog

Color dialogs let the user select a color, which is returned in the dialog object's Color property.

Important properties

No.	Property	Use
1	AllowFullOpen	Gets/sets whether the user can use the dialog box to define custom colors.
2	AnyColor	Gets/sets whether the dialog box displays all available colors in the set of basic colors.
3	Color	Gets/sets the color selected by the user.
4	FullOpen	Gets/sets whether the controls used to create custom colors are visible when the dialog box is opened
5	ShowHelp	Gets/sets whether a Help button appears in the color dialog box.
6	SolidColorOnly	Gets/sets whether the dialog box will restrict users to selecting solid colors only.

Important Methods

No.	Method	Use
1	Reset	Resets all dialog options to their default values.
2	ShowDialog	Shows the dialog.

Important Events

No.	Event	Use
1	HelpRequest	Occurs when the user clicks the Help button.

FontDialog

Font dialogs let the user select a font size, face, color, and so on.
To display the font dialog box, call the ShowDialog method. This dialog shows list boxes for Font, Style, and Size, checkboxes for effects like Strikeout and Underline, a drop-down list for Script (Script refers to different character scripts that are available for a given font—for example, Hebrew), and a sample of how the font will appear. You can recover these settings using properties of the same names of the Font object returned by the Font property.

Important properties

No.	Property	Use
1	Font	Returns font selected by the user.
2	MaxSize	Maximum size of font that can be selected.
3	MinSize	Minimum size of font that can be selected.
4	ShowApply	Whether the apply button should be shown.
5	ShowColor	Whether the Color button should be shown to change font color.
6	ShowEffects	Whether to show underline, strikeout and font color selections.

Important Methods

No.	Method	Use
1	Reset	Resets all dialog options to their default values.
2	ShowDialog	Shows the dialog.

Menus

IMPORTANT NOTE: MainMenu control is replaced by MenuStrip control whereas MenuItem control is replaced by ToolStripMenuItem in VB.NET. However, you can still create aand use objects of MainMenu/MenuItem class in code.

Menus are those controls that allow the user to make selections and also hide away those selections when they're not needed, saving space in Windows applications.

In Visual Basic, the MainMenu control represents the container for the menu structure of a form; you can assign a control of this type to a form's Menu property at run time. Menus are made up of MenuItem objects that represent the individual parts of a menu—menu items can be a parent menu or a menu item in a menu.

There are all kinds of options here—you can add submenus to menus that will pop up when the user clicks an arrow in a menu item, display check marks, create menu separators (horizontal bars used in menus to group menu items), assign shortcut keys (like Ctrl+H) to menu items, even draw the appearance of menu items yourself. These actions are actually supported by MenuItem objects, not MainMenu objects.

Menu Items
Menus like File or Edit and the actual items in such menus are supported with the MenuItem class. This class supports the actual controls in your menu system, and it's their Click event that you add code to in order to make that menu system active.

Important properties

No.	Property	Use
1	Text	Text to be displayed on menu item.
2	Checked	Whether a chek-mark should be displayed next to Text in menu item.
3	Shortcut	A keyboard shortcut that can be pressed to select the menu item.
4	ShowShortcut	Whether shortcut key should be displayed in menu item.
5	Enabled	Whether the menu item should be enabled.

* Setting the Text property to a hyphen (-) converts the menu item into a menu separator, one of those horizontal bars that help group menu items together. (You can even have separators in menu bars, in which case they're vertical.)

Prefacing a character in a menu item's caption with an ampersand (&) underlines that character and makes it into an access key, which means the user can select that item by pressing Alt and that character. For example, giving a menu item the caption "E&xit" makes X into the access key for this menu item.

The most common menu item event that you handle is Click, which means the user has clicked a menu item and your code should respond to it. However, there are other events here as well—the Popup event lets you to perform tasks before a menu is displayed, because it happens just before a menu item is displayed. And the Select event happens when a menu item is selected (that is, highlighted). This enables you to perform tasks like displaying help for menu items when the user places the mouse cursor over those items.

Debugging

Bug means an error in the code. Debugging is the process of execution of the code under inspection, with the aim of removing such bugs (i.e. errors).

The Visual Studio IDE has following built-in features for debugging:

- Breakpoints
- Stepping through the program using Step into (F11) and Step over (F10)
- Watch

Setting Breakpoints

When a program is running in the debugger, a breakpoint will halt execution and give the developer control of the debugger.
Lines with a breakpoint are highlighted in red. You can remove a breakpoint by right-clicking the line again and selecting Remove Breakpoint.

Stepping Thru a Program

Once a breakpoint is set, the program can be run in the debugger.
In the Debug menu, select Start instead of Start Without Debugging
This starts your program in the debugger, and the breakpoints will be enabled.
To step through one line of code, select Debug | Step Over and watch the cursor move to the next line.
The Debug | Step Into command allows you to step into a function that is about to be called.

While in the debugger mode, you can use the immediate window to check/change the values of variables.

Debugging Using the Watch Window and QuickWatch Dialog Box

The Watch window provides a method for you to observe variables and expressions easily while the code is executing — this can be invaluable when you are trying to debug unwanted results in a variable. You can even change the values of variables in the Watch window. You can also add as many variables and expressions as needed to debug your program. This provides a mechanism for watching the values of your variables change without any intervention on your part. This is an easy place to watch many variables.

The QuickWatch dialog box is best for watching a single variable or expression. You can add or delete variables or expressions in the QuickWatch dialog box only when your program is in break mode.