UNIT 2

INTRODUCTION TO WINDOWS COMMON CONTROL

Outline of the Chapter

1. INTRODUCTION TO PROPERTIES, METHODS AND EVENTS:

- VB.Net contains a very rich amount of the controls for the windows application.
- Every control has three important elements which are Properties, Events and Methods.
- **Properties** means all the controls can resized, customized and moved. A property is a value or characteristics held by the particular control.
- **An Event means** a signal that informs an application that something important has occurred.
- Some of the common Properties, Method and Events of all controls are as below.

1.1 Common Properties:

- Each and every control has Name property and value of name property is unique in that module. Properties are similar to variable, like variable properties have its own data type. Variable is independent, while properties always associate control or objects.
- Depend on the data type of properties, only that type of data will accept that is string data type will accept string values, integer data type will accept integer values etc.
- The following are some of the commonly used properties:

Property Name	Description
Name	This is the actual name of the form.
Autosize	Gets or Sets a value indicating whether the control automatically resize based on its content.
AutoScroll	Gets or Sets a value indicating whether the container will allow automatically resize based on its content.
Text	The text, which will appear at the title bar of the form.
Width	This is the width of the form in pixel.
Font	It specifies font type, style and size.
Enabled	If True, allows the form to respond to mouse and keyboard events; if False, disables form.
ControlBox	By default, It is True and you can set it to False to hide the icon and disable the Control menu.
BackColor	Sets the form background color.
ForeColor	Sets the form foreground color.
Bottom	Get the distance value in pixels, between the bottom edge of the control and bottom edge of its container's area.
Top	Get the distance value in pixels, between the top edge of the control and top edge of its container's area.
Left	Get the distance value in pixels, between the left edge of the control and left edge of its container's area.
Right	Get the distance value in pixels, between the right edge of the control and

	right edge of its container's area.
Enable	Get or Set a value indicating whether the control can respond to user interaction or not.
Font	Get or Set the font of the text displayed by the control.
TabIndex	Get or Set the tab order of the control within its container.
TabStop	Get or Set a value indicating that whether the user can give the focus to this control using tab key.
Visible	Determines whether the control is visible or hidden.
Height & Width	It specifies height and width of the control.

1.2 Common Methods:

- Methods are block of code designed into controls that tell the control how to do things such as move to another location on a form.
- Methods are used to access or manipulate the characteristics of an object or a variable.
- Just as with properties not all controls have the same methods, some common methods do exist as shown in below figure.

Method Name	Description
Move	Changes an object's position on response to a code request
Drag	Handles the execution of a drag and drop operation by the user.
SetFocus	Gives focus to the object specified in the method call.
ToString()	A system.string containing the name of the component, and null if the component have unnamed

1.3 Common Events:

- An event is an automatic notification that some action has occurred, or event is action performed by system or user using keyboard or mouse.

Event Handler:

- An event handler is method that is bound to an event.
- When the event is raised, the code within the event handler is executed.
- Each event handler provides two parameters that allow you to handle the event properly.

Syntax:

```
Modifier Sub ObjectName_EventHandlerName(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
System.Object.EventHandlerName
        Code…
End Sub
```

Example:

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

```
                Code…
        End Sub
```

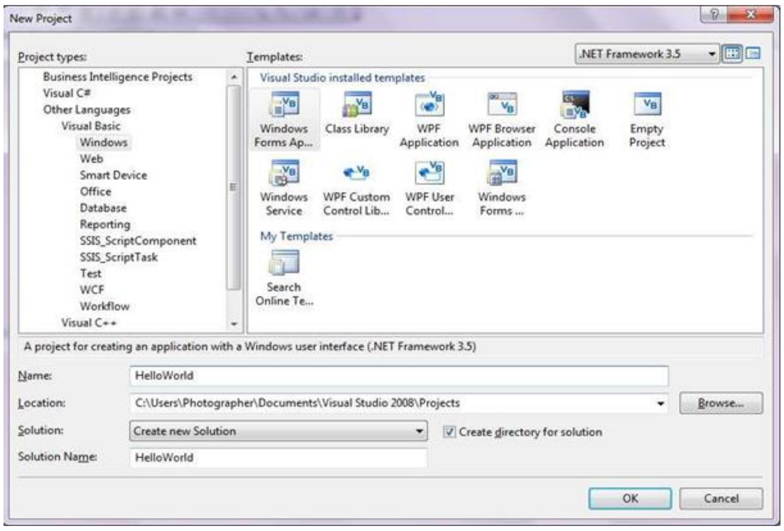
- The first parameter “sender” provides a reference to the object that rises the event.
- The second parameter “e” in the example above, passes an object specific to the event that is being handled.
- By referencing the objects properties you can get information such as the location of mouse for mouse event or data being transferred in drag drop event.
- Typically such event produced an event handler with a different event object type for second parameter.
- Some event handlers such as those for MouseDown and MouseUp events, have the same object type for their second parameter. For these types of event you can use the same event handler to handle both events.
- You can also use the same event handler to handle the same event for different controls.
- For Example, if you have group of RadioButton controls on a form, you could create a single event handler for the click event and have each control’s click event bound to the single event handler.
- Difference between event and methods is that to call a method you have to call explicitly in your code, while event is called automatically when user action is performed on object.

Event Name	Description
Click	Occurs when the control is clicked.
DoubleClick	Occurs when the control is double clicked.
GetFocus	Occurs when the control received focus.
LostFocus	Occurs when the control lost focus.
KeyDown	Occurs when a key is pressed.
KeyPress	Occurs when a key is pressed while the control has focused.
KeyUp	Occurs when a key is released.
MouseClick	Occurs when the control is clicked by mouse.
MouseDoubleClick	Occurs when the control is double clicked by mouse.
MouseDown	Occurs when the mouse pointer is over the control and mouse button is pressed.
MouseEnter	Occurs when the mouse pointer enter the control.
MouseHover	Occurs when the mouse pointer rests on the control.
MouseLeave	Occurs when the mouse pointer leaves the control.
MouseMove	Occurs when the mouse pointer is over the control.
MouseUp	Occurs when the mouse pointer is over the control and mouse button is released.
MouseWheel	Occurs when the mouse wheel moves while the control has focus.

Resize	Occurs when the control is resized.
CausesValidationChanged	Occurs when the value of Cause Validation property changes.
Validation	Occurs when the control is successfully Validated.
Validating	Occurs when the control is Validating.

2. FORM:

- A Form is a representation of any window displayed in the application.
- A Form object is considered as container for other control like button, textbox, label etc.
- When you create VB.NET windows application, Form object automatically added to your application.
- To create a New Window Forms Application, Select **File** > **New Project**. It will open one dialog box which is shown in below figure.



A project for creating an application with a Windows user interface (.NET Framework 3.5)

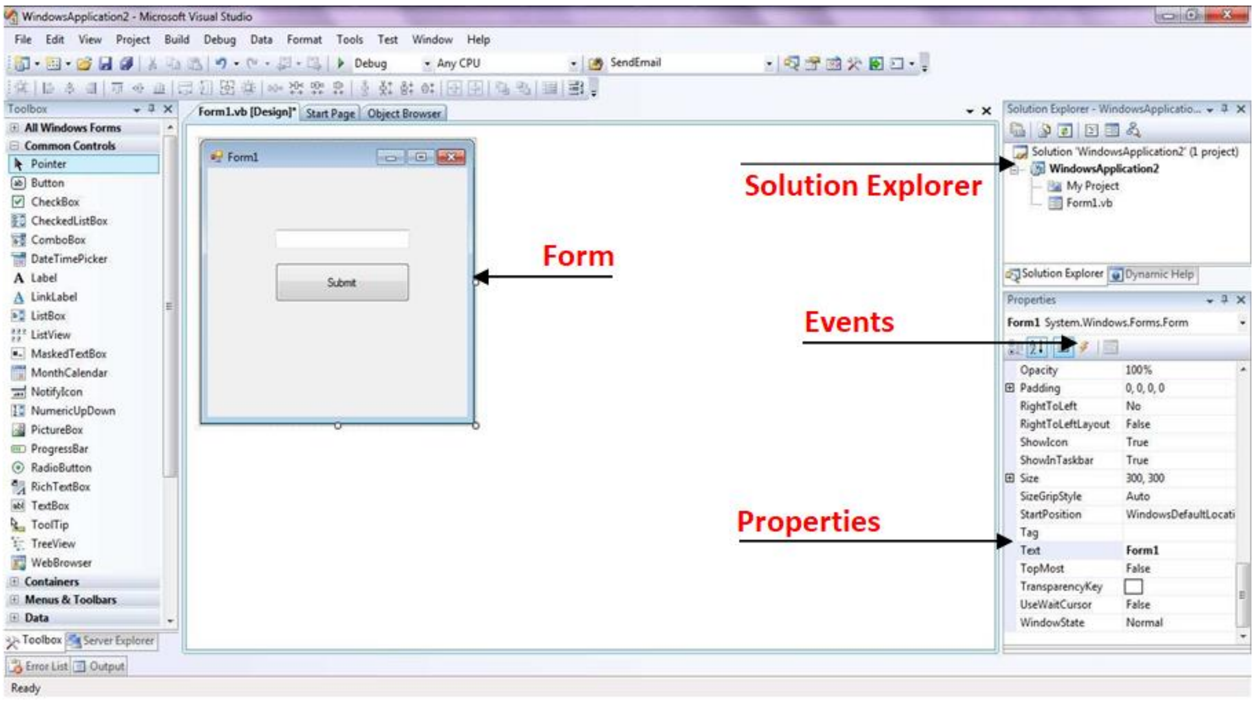
Name: HelloWorld

Location: C:\Users\Photographer\Documents\Visual Studio 2008\Projects

Solution: Create new Solution

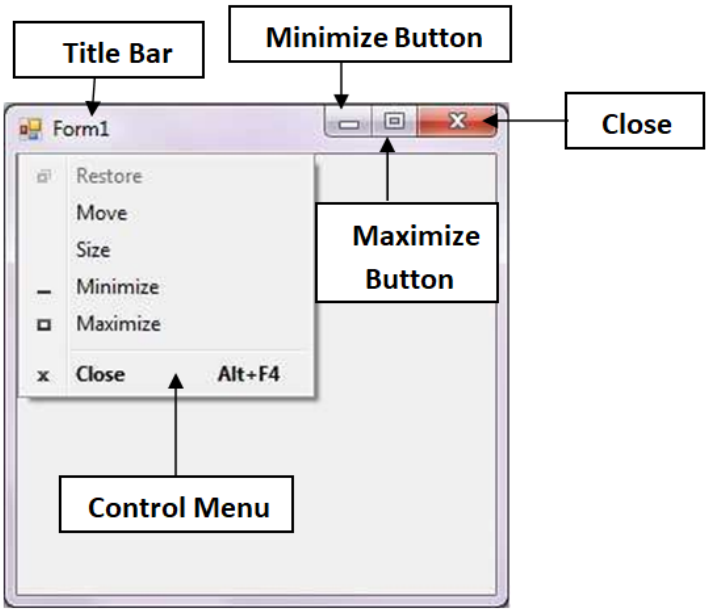
☑ Create directory for solution

Solution Name: HelloWorld



Opacity	100%
Padding	0, 0, 0, 0
RightToLeft	No
RightToLeftLayout	False
ShowIcon	True
ShowInTaskbar	True
Size	300, 300
SizeGripStyle	Auto
StartPosition	WindowsDefaultLocati
Tag	
Text	Form1
TopMost	False
TransparencyKey	☐
UseWaitCursor	False
WindowState	Normal

- This dialog box lets you select the type of project you want to create by choosing one of several templates.
- To create a **Windows Forms application**, Select Project types **windows in visual basic**. After that **select Windows Forms Application** from the project templates.
- Enter a name for the project and select the location for the project. A folder with the same name as the project is automatically added to the location you specify.
- Finally, select **OK**, Microsoft Visual Studio creates your project and displays window Form with a name Form1 which is shown in below figure.
- Visual Studio creates a default form for you when you create a Windows Forms Application. Every form will have title bar on which the form's caption is displayed and there will be buttons to close, maximize and minimize the form shown in below figure.



- If you click the icon on the top left corner, it opens the control menu, which contains the various commands to control the form like to move control from one place to another place, to maximize or minimize the form or to close the form.

2.1 Forms Properties:

The following are some of the commonly used properties:

Property Name	Description
Name	This is the actual name of the form.
Text	The text, which will appear at the title bar of the form.
Width	This is the width of the form in pixel.
Font	It specifies font type, style and size.
Enabled	If True, allows the form to respond to mouse and keyboard events; if False, disables form.
ControlBox	By default, It is True and you can set it to False to hide the icon and disable the Control menu.

BackColor	Sets the form background color.
ForeColor	Sets the form foreground color.
AcceptButton	The button that's automatically activated when you press Enter, no matter which control has the focus at the time.
CancelButton	The button that's automatically activated when you hit the Esc key.
StartPosition	It determines the initial position of the form when it's first displayed.
BorderStyle	It determines the style of the form's border and the appearance of the form.
MinimumSize	It specifies the minimum height and width of the window you can minimize.
MinimizeBox	It specifies that, minimize button to disable or not in the form.
MaximumSize	It specifies the maximum height and width of the window you maximize.
MaximizeBox	It specifies that, maximize button to disable or not in the form.
IsMDIContainer	By default, this property is False and you can set it to True for generating MDI application.
Opacity	It specifies the level of opacity of the form.
Icon	It represents the Icon to show at top left corner of the form.
Height & Width	It specifies height and width of the form.
Cursor	It specifies the design or cursor to be display in the form, while mouse pointer is moving in the form.

2.2 Forms Methods:

The following are some of the commonly used methods of the Form class:

Method Name	Description
Activate	Activates the form and gives it focus.
ActivateMdiChild	Activates the MDI child of a form.
CenterToScreen	Centers the form on the current screen.
Close	Closes the form.
Focus	Sets input focus to the control.
Show	Displays the control to the user.
Refresh	Forces the control to invalidate its client area and immediately redraw itself and any child controls.
ShowDialog	Shows the form as a modal dialog box.

2.3 Forms Events:

The following are various important events related to a form:

Event Name	Description
Activated	Occurs when the form is activated in code or by the user.
Click	Occurs when the form is clicked.
Enter	Occurs when the form is entered.
Move	Occurs when the form is moved.
Closed	Occurs before the form is closed.
Closing	Occurs when the form is closing.
Load	Occurs before a form is displayed for the first time.
Resize	Occurs when the control is resized.
Shown	Occurs whenever the form is first displayed.
DoubleClick	Occurs when the form control is double-clicked.
VisibleChanged	Occurs when the Visible property value changes.
HelpButtonClicked	Occurs when the Help button is clicked.

3. PROCEDURE ORIENTED, OBJECT ORIENTED AND EVENT DRIVEN PROGRAMMING:

Procedure oriented Programming (POP)	Object oriented Programming (OOP)	Event driven Programming (EDP)
POP main focus on Procedure / functions.	OOP main focus on data (object) rather than procedure.	EDP emphasis on events occurred by user.
The execution of the program starts with the first line and follows a predefined path.	The execution of the program is dependent on the basics of function calling.	User defines the flow of execution.
Large problems are divided into smaller functions.	Large problems are divided into smaller classes.	Large problems are divided into smaller procedures and functions.
Complier is used for the error checking or while compiling the code.	Compiler is used for the error checking or while compiling the code.	Interpreter is used for error checking or while compiling the code.
Whole program is checked and after that if any errors were generated during the compile process the error is displayed.	Whole program is checked and after that if any errors were generated during the compile process the error is displayed.	The program is checked line by line and generates errors when press enters after completion of line.
The functions are in independent which makes the program lengthy and complex.	It makes the program more easily than POP.	The program execution is much easier than OOP and POP.
Do not provides a good graphical user interface.	Do not provide a good graphical user interface.	Provides a good graphical user interface.

A particular or specific flow for the program execution.	Program execution starts from first line and ending at last line so follows a particular approach.	A particular direction is not followed is not followed in this kind of programs.
Examples: C, COBOL, FORTRAN etc.	Examples: C++, JAVA, C#.NET, VB.NET	Example: Visual Basic 6

4. INPUTBOX:

An inputbox is a specially designed dialog box that allows the programmer to request a value from the user and use that value.

InputBox will display a prompt in a dialog box, waits for the user to input text or click a button, and returns a string containing the contents of the text box.

Syntax:

Variable = InputBox (Prompt, [title], [default], [xpos], [ypos])

The arguments to the InputBox function are described below:

Argument	Description
prompt	**Required**, String expression displayed as the message in the dialog box.
Title	**Optional**, String expression displayed in the title bar of the dialog box.
Default	**Optional**, String expression displayed in the text box as the default response if no other input is provided.
xpos and ypos	**Optional**, Numeric expressions that specify custom positioning of the box on screen.

Example:

Dim myValue As String

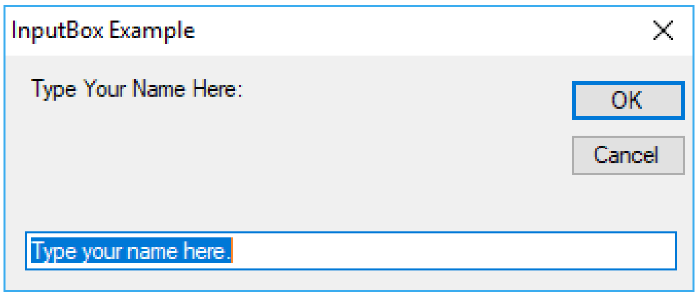
myValue = InputBox(“Type Your Name Here:”, “InputBox Example”, “Type your name here.”, 100, 100)

Example:

```
Public Class Inputboxx
     Private Sub Inputboxx_Load (ByVal sender As System.Object, ByVal e As
     System.EventArgs) Handles MyBase.Load

          Dim strUserName As String
          strUserName = Inputbox("Type Your Name Here:", "InputBox
          Example", "Type your name here.")

          If strUserName = "" Then
               'user cancelled the InputBox, so clear the label
               lbninfo.Text = ""
          Else
```

```
                lbninfo.Text = "Your Name is: " & strUserName
            End If
        End Sub
End Class
```

Output:



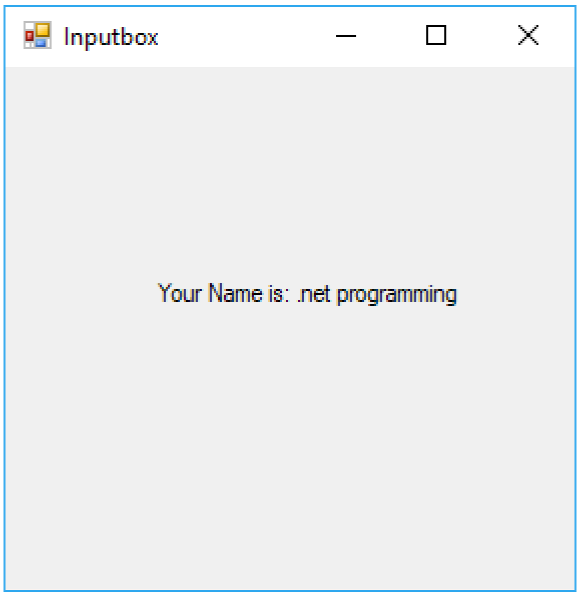
InputBox Example

Type Your Name Here:

OK

Cancel

Type your name here.



Inputbox

Your Name is: .net programming

5. MESSAGEBOX:

- A MessageBox is a predefined dialog box that enables you to displays application-related information or custom message to the user.
- Message boxes are also used to request information from the user or take any input from the user. To show message to user you have to call show( ) method of the Message Box as MessageBox.Show(). This will displays a dialog box.
- It interrupts the user. It immediately blocks further interaction. It requires only one Function call.

Syntax:

MessageBox.Show (Message Text, Caption, Buttons, Icon, Default Button)

Argument	Description
Message Text	Text is a compulsory argument. The String that you specify as a Text will display as a message in the Dialog Box.
Caption	**Optional**, The String that you specified as a caption will be display in the title bar of the Dialog Box. If omit this parameter then null value will be displayed on the title bar.
Button	**Optional**, Buttons is used to specify type of buttons to display in the message box. If omit this parameter, then only one button "OK" will displayed.
Icon	**Optional**, Icon is used to specify type of icon to display in the message box.

Default Button	**Optional,** It represents a value from the MessageBoxDefault enumeration. This parameter enables you to specify which buttons is set as the default button in the messagebox dialog box.

MessageBox Button Enumeration:

OK	The MessageBox Display OK Button.
OKCancel	The MessageBox Display OK and Cancel Buttons.
AbortRetryIgnore	The MessageBox Display Abort, Retry and Ignore Buttons.
YesNoCancel	The MessageBox Display Yes, No and Cancel Buttons.
YesNo	The MessageBox Display Yes and No Buttons.
CancelRetry	The MessageBox Display Cancel and Retry Buttons.

MessageBox Image Enumeration:

Member Name	Description
None	No icon is displayed.
Hand	The messagebox displays a hand icon.
Question	The messagebox displays a question mark icon.
Exclamation	The messagebox displays an Exclamation mark icon.
Asterisk	The messagebox displays an Asterisk mark icon.
Stop	The messagebox displays a Stop mark icon.
Error	The messagebox displays an Error mark icon.
Warning	The messagebox displays a Warning mark icon.
Information	The messagebox displays an Information mark icon.

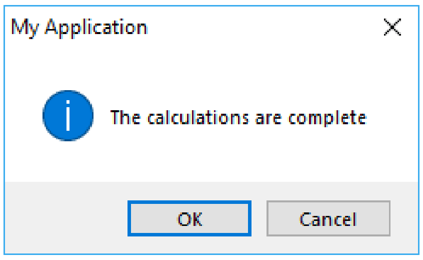
- To display information to the user in a message box:

Navigate to where you would like to add the code for the message box.

Add code using the Show method.

Example:

```
MessageBox.Show ("The calculations are complete" , "My Application",
MessageBoxButtons.OKCancel , MessageBoxIcon.Asterisk )
```


My Application

The calculations are complete

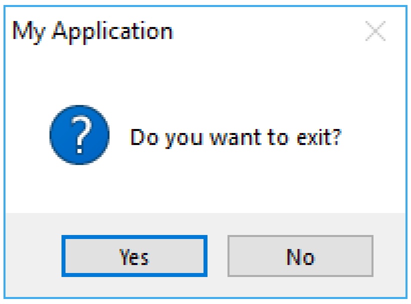
- To display a message box to request information:

Open the Code Editor for your class and navigate to where you would like to add the code for the message box.

Add code that uses the Show method of the MessageBox class to display a message box.

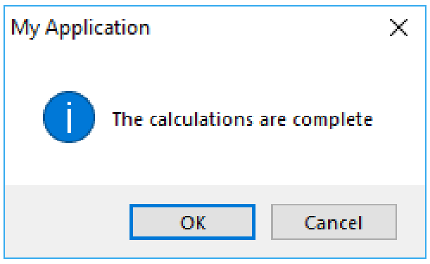
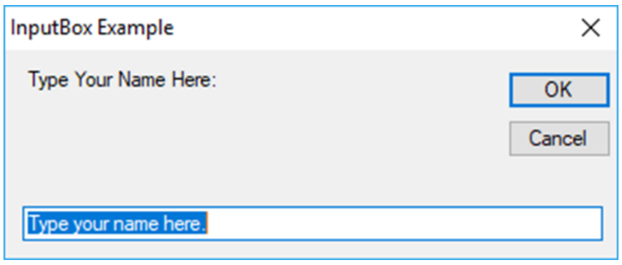
Example:

```
If MessageBox.Show ("Do you want to exit?",  "My Application",
MessageBoxButtons.YesNo,  MessageBoxIcon.Question) = DialogResult.Yes
Then
    Application.Exit ()
End If
```


My Application

Do you want to exit?

5.1 Difference between Message Box and Input Box:

Message Box	Input Box
A MessageBox is a predefined dialog box that enables you to displays application-related information or custom message to the user.	An InputBox is a specially designed dialog box that allows the programmer to request a value from the user and use that value.
MessageBox are also used to request information from the user or take any input from the user.	InputBox is only use to take input from user at runtime.
To show message to user you have to call show() method of the Message Box as MessageBox.Show(). This will display a dialog box.	There is no need of any method to display InputBox. It is directly displayed by writing its code in program.
Syntax: MessageBox.Show (Message Text, Caption, Buttons, Icon, Default Button)	Syntax: Variable = InputBox(Prompt, [title], [default], [xpos], [ypos])
It can return Boolean value that is true or false (Yes or No), Depending on Buttons of MessageBox	It can return string value, thus a variable is require to store the string value.
 My Application The calculations are complete 	 InputBox Example Type Your Name Here: Type your name here

6. WORKING WITH COMMON TOOLBOX CONTROL :

6.1 Label Control:

- The Label control represents a standard Windows label. It is generally used to display some text on the GUI which is not changed during runtime. A label participates in the tab order of a form, but does not receive any focus.

Properties:

Property Name	Description
BorderStyle	Gets or sets the border style for the control.
Autosize	It gets or sets a value specifying if the control should be automatically resized to display all its contents.
Text	Used to Display the text.
TextAlign	Gets or sets the alignment of text in the label.

Events:

Event Name	**Description**
Leave	Occurs when the input focus leaves the control.
LostFocus	Occurs when the control loses focus.
TextChanged	Occurs when the Text property value changes.

Other events are Click, MouseMove, MouseUp, MouseDown, MouseEnter events are used in Label control.

Methods:

Method Name	Description
Select()	Activates the control.
Show()	Displays the control to the user.
ToString()	Returns a String that contains the name of the control.

6.2 Button Control:

It represents a Windows button control. It is generally **used to generate a Click event by providing a handler for the Click event**.

If button has an underlined letter in its label, you can optionally hold down the Alt key and press the key that represents the underlined letter.

Properties:

Property Name	**Description**
BackColor	It gets or sets the background color of the control.

AutoSizeMode	Gets or Sets the mode by which the button automatically resized itself.
DialogResult	It gets or sets a value that is returned to the parent form when the button is clicked. This is used while creating dialog boxes.
Font	Sets the font.
Location	Gets or Sets the coordinates of the upper-left corner of the relative to the upper-left corner of its container.
Image	It gets or sets the image that is displayed on a button control.
ImageAlign	It gets or sets the alignment of the image on the button control
Size	Sets the size.
Enable	By default, this property is True and you can set it to False to disable the working of button.

Events:

Event Name	Description
GotFocus	Occurs when the control receives focus.
TextChanged	Occurs when the Text property value changes.
Click	Occurs when user clicks the Button.
TabIndexChanged	Occurs when the text property value changes.
VisibleChanged	Occurs when the visible property value changes.

Other events are GotFocus, LostFocus, MouseMove, MouseUp, MouseDown, MouseEnter events are used in Button control.

Methods:

Method Name	Description
Select()	Activates the control.
ToString()	Returns a String containing the name of the Component,
NotifyDefault()	Notifies the Button whether it is the default button so that it can adjust its appearance accordingly.
GetPrefferedSize	Retrieves the size of a rectangular area into which a control can be fitted.

6.3 TextBox Control:

Text box controls allow entering text on a form at runtime.

By default, it takes a single line of text but you can make it accept multiple line texts and also can add scroll bars to it.

Before you can type in a text box, however the blinking curser needs to be in the text box. To move the cursor into the text box, just click the text box.

TextBox control can also be used to accept passwords and other sensitive information. You can use the PasswordChar property to mask characters entered in a single line version of the control.

You can limit the amount of text entered into a TextBox control by setting the MaxLength property to a specific number of the character.

Use the CharacterCasting property to enable the user to type only uppercase, only lowercase, or a combination of Uppercase and Lowercase characters into the TextBox control.

Properties:

Property Name	Description
AutoSize	Gets or sets a value indicating whether the height of the control automatically adjusts when the font assigned to the control is changed.
MaxLength	Gets or sets the maximum number or characters the user can type or paste into the TextBox control.
ReadOnly	Gets or sets a value indicating whether the text in the textbox is read only.
CharacterCasing	Gets or sets whether the TextBox control modifies the case of characters as they are typed.
Multiline	Gets or sets a value indicating whether this is a multiline TextBox control.
PasswordChar	Gets or sets the character used to mask characters of a password in a single-line.
Text	Gets or sets the current text in the TextBox control.
TextAlign	Gets or sets how text is aligned in a textbox control.

Events:

Event Name	Description
TextAlignChanged	Occurs when the TextAlign property value changes.
TextChanged	Occurs when the text in the textbox changes.
FontChanged	Occurs when the font property value change.
MultilineChanged	Occurs when the value of the multiline property has changed.

Other events are GotFocus, LostFocus, MouseMove, MouseUp, MouseDown, MouseEnter events are used in TextBox control.

Methods:

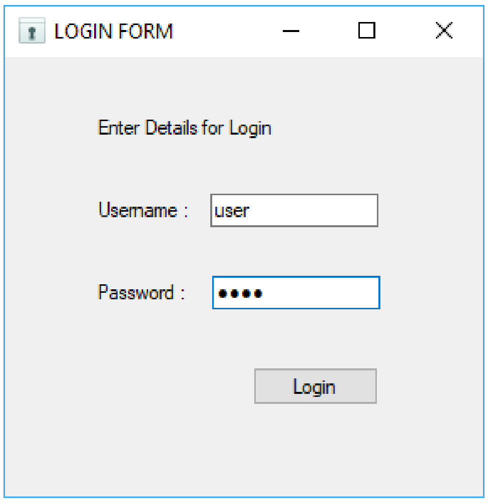
Method Name	Description
AppendText()	Appends text to the current text of a text box.
Clear()	Clears all text from the text box control.
ResetText()	Resets the Text property to its default value.
Copy()	Copies the current selection in the textbox to the clipboard.
Cut()	Moves the current selection in the textbox to the clipboard.
Paste()	Replace the current selection in the textbox with the contents of the clipboard.
Paste(String)	Sets the selected text to the specified text without clearing the undo buffer.

ResetText	Resets the Text property to its default value.
ToString	Returns a string that represents the TextBoxBase control.
Undo	Undoes the last edit operation in the TextBox.

6.4 Example of Label, Button and TextBox Control (Login Application):

```
Public Class LABEL_BUTTON_TEXTBOX
      Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
      System.EventArgs) Handles btn_login.Click
            If txtbox_username.Text = "user" Then
                  If txtbox_password.Text = "1234" Then
                        MessageBox.Show("Login is successfull")
                  Else
                        MessageBox.Show("PASSWORD doesnot match!!")
                  End If
            Else
                  MessageBox.Show("USERNAME doesnot match!!")
            End If
      End Sub
End Class
```

Output:

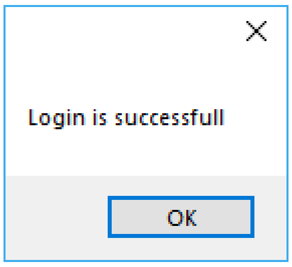

LOGIN FORM

Enter Details for Login

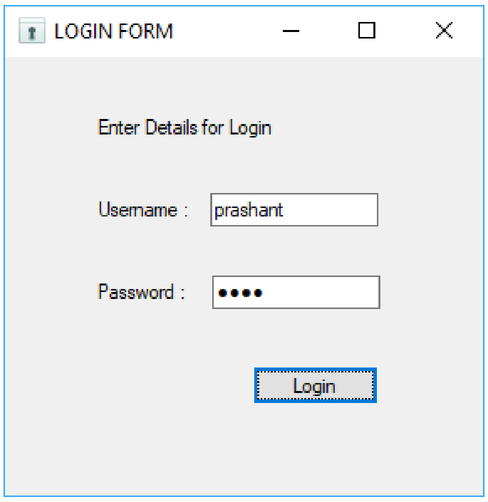
Username : user

Password : ●●●●

Login


Login is successfull

OK

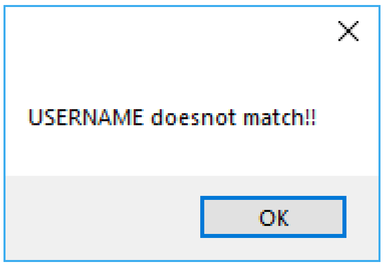

LOGIN FORM

Enter Details for Login

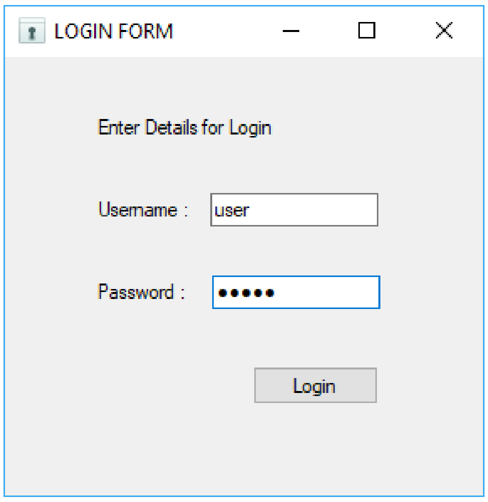
Username : prashant

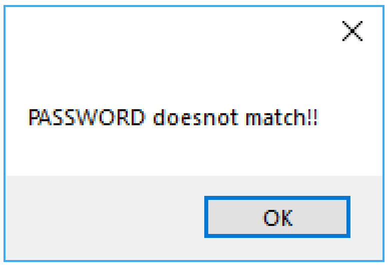
Password : ●●●●

Login


USERNAME doesnot match!!

OK




PASSWORD doesnot match!!

6.5 NumericUpDown Control:

- A NumericUpDown control allows users to provide a spin (up/down) interface to move through pre- defined numbers using up and down arrows.
- With many different properties and event handlers, the NumericUpDown control is ideal for certain user interfaces where a number should be entered or adjusted.
- It contains a single numeric value that can be incremented or decremented by clicking the up or down buttons of the control.
- User can also enter a value, unless the **ReadOnly** property is set to true.
- Its display can be formatted by setting the DecimalPlaces, Hexadecimal or ThousandsSeparator properties.
- To specify the allowable range of values for this control, set the **Minimum and Maximum** properties.
- You can also increase speed that the control moves through numbers when the user continuously presses the up or down arrow by setting the **accelerations property**.
- When the **UpButton and DownButton methods are called**, either in code or by the click of the up or down buttons, the new value is **validated and the control is updated** with the new value in appropriate format.

Properties:

Property Name	Description
AutoSize	Gets or sets a value indicating whether the control should automatically resize based on its contents.
UpDownAlign	Gets or sets the alignment of the up and down buttons on the spin box.
InterceptArrowKeys	Gets or sets a value indicating whether the user can use the UP ARROW and DOWN ARROW keys to select values.
DecimalPlaces	Gets or sets the number of decimal places to display in the spin box.
HexaDecimalPlaces	Gets or sets a value indicating that whether the value display in hexadecimal format or not.

ThousandSeparator	Gets or sets a value indicating whether a thousand separator is displayed in the spin box.
Minimum	Gets or sets the minimum value for the spin box.
Maximum	Gets or sets the maximum value for the spin box.
Increment	Gets or sets the value to increment the spin box, when the Up and Down button are clicked.

Events:

Event Name	Description
Scroll	Occurs when the user or code scrolls through the client area.
Clicked	Occurs when the control is clicked.
TextChanged	Occurs when the value of the Text Property changes.

Other events are DoubleClick, KeyUp, KeyDown, KeyPress, GotFocus, LostFocus, MouseUp, MouseDown events are used in NumericUpDown control.

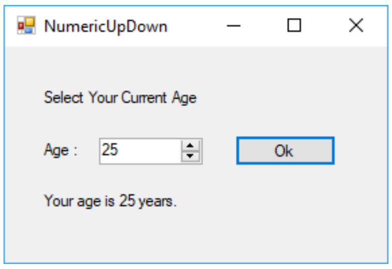
Methods:

Method Name	Description
UpButton()	Increments the value of the spin box (also known as an up-down control).
DownButton()	Decrements the value of the spin box (also known as an up-down control).
OnValueChanged()	Raises the ValueChanged event.
ParseEditText()	Converts the text displayed in the spin box to a numeric value and evaluate it.
UpdateEditText()	Displays the current value of the SpinBox in the appropriate format.

Example:

```
Public Class NumericUpDown
     Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
     System.EventArgs) Handles btnOk.Click
          lblAnswer.Text = "Your age is " + numUDAge.Value.ToString() + "
          years."
    End Sub
End Class
```

Output:



Age : 25

Your age is 25 years.

6.6 CheckBox Control:

- The CheckBox control allows the user to set true/false or yes/no type options. The user can select or deselect it.
- When a check box is selected it has the value True, and when it is cleared, it holds the value False.
- You can select any combination of checkbox you wants.
- When a checkbox is selected, the option is turned on. When the checkbox is empty(clear), the option is turned off.
- The appearance property determines whether the CheckBox appears as a typical CheckBox or a Button. The three state properties determine whether the control supports two or three states.
- Use the **checked property** to get or set the value of a two-state CheckBox control and use the **CheckState property** to get or set the value of three-state CheckBox.

Properties:

Property Name	Description
AutoCheck	Gets or sets a value indicating whether the Checked or CheckedState value and the appearance of the control automatically change when the check box is selected.
Appearance	Property used to set or get the appearance of a checkbox.
Checked	Gets or sets a value indicating whether the check box is selected.
CheckState	Default value is Unchecked. Set it to True if you want a check to appear. When set to Indeterminate it displays a check in gray background.
FlatStyle	Default value is Standard. Select the value from a predefined list to set the style of the checkbox.
Text	Gets or sets the caption of a check box.
ThreeState	Gets or sets a value indicating whether or not a check box should allow three check states rather than two.

Events:

Event Name	Description
AppearanceChanged	Occurs when the value of the Appearance property of the check box is changed.
CheckedChanged	Occurs when the value of the Checked property of the CheckBox control is changed.
CheckStateChanged	Occurs when the value of the CheckState property of the CheckBox control is changed.

Methods:

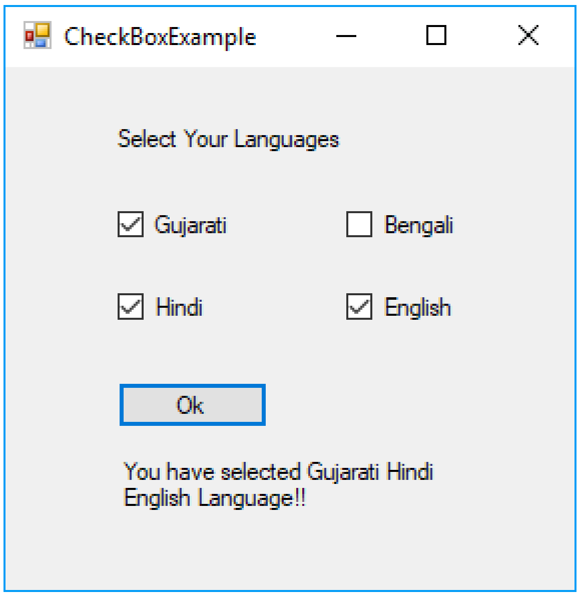
Method Name	Description
OnClick()	Raises the OnClick event.

OnCheckedChanged()	Raises the CheckedChanged event.
OnCheckStateChanged()	Raises the CheckStateChanged event.

Example:

```
Public Class CheckBoxExample
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles btnOk.Click

        lblLanguage.Text = "You have selected "

        If chkboxGujarati.Checked = True Then
            lblLanguage.Text &= "Gujarati "
        End If

        If chkboxHindi.Checked = True Then
            lblLanguage.Text &= "Hindi "
        End If

        If chkboxBengali.Checked = True Then
            lblLanguage.Text &= "Bengali "
        End If

        If chkboxEnglish.Checked = True Then
            lblLanguage.Text &= "English "
        End If

        lblLanguage.Text &= "Language!!"

    End Sub
End Class
```

Output:


CheckBoxExample

Select Your Languages

☑ Gujarati ☐ Bengali

☑ Hindi ☑ English

Ok

You have selected Gujarati Hindi English Language!!

6.7 RadioButton Control:

- RadioButton are set of two or mutually exclusive options.
- Selecting one will deselect other one automatically. It means they are always selecting only one out of all the option buttons
- All RadioButton controls in a given container, such as a Form, constitute a group. To create multiple groups on one form, place each group in its own container, such as GroupBox or Panel Control.

Properties:

Property Name	Description
Appearance	Gets or sets a value determining the appearance of the radio button.
CheckAlign	Gets or sets the location of the check box portion of the radio button.
Checked	Gets or sets a value indicating whether the control is checked.
FlatStyle	Gets or sets the flat style appearance of the button control.
Text	Gets or sets the text associated with the radio button.
AutoCheck	Gets or sets a value indicating whether the Checked value and the appearance of the control automatically change when the control is clicked.

Events:

Event Name	Description
AppearanceChanged	Occurs when the value of the Appearance property of the RadioButton control is changed.
CheckedChanged	Occurs when the value of the Checked property of the RadioButton control is changed.

Other events are Clock, MouseUp, MouseDown and MouseMove.

Methods:

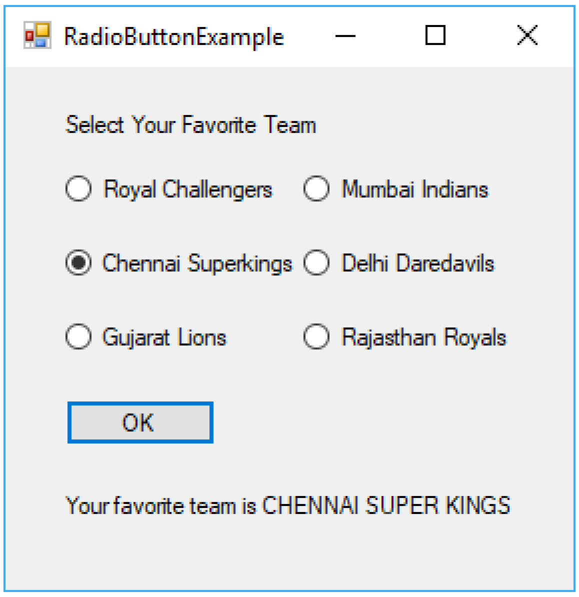
Method Name	Description
PerformClick	Generates a Click event for the control.

Example:

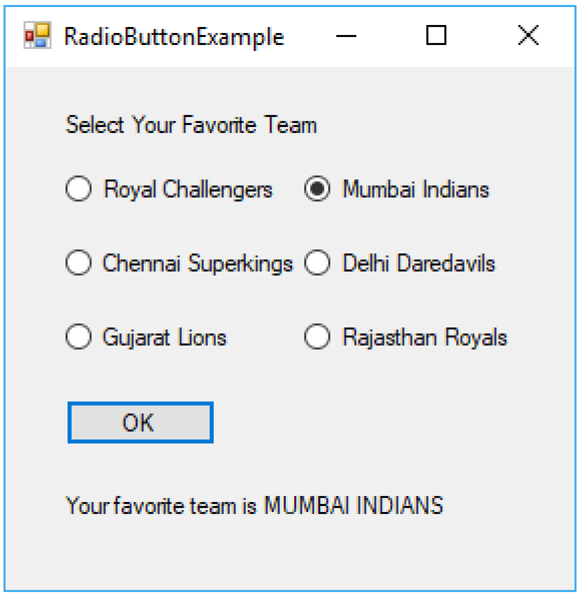
```
Public Class RadioButtonExample
     Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
     System.EventArgs) Handles btnOK.Click
          lblTeam.Text = "Your favorite team is "
          If rdRCB.Checked = True Then
               lblTeam.Text &= "ROYAL CHALLENGERS BANGLORE "
          End If
          If rdCSK.Checked = True Then
               lblTeam.Text &= "CHENNAI SUPER KINGS "
          End If
          If rdGL.Checked = True Then
               lblTeam.Text &= "GUJARAT LIONS "
```

```
            End If
            If rdDD.Checked = True Then
                lblTeam.Text &= "DELHI DAREDEVILS "
            End If
            If rdMI.Checked = True Then
                lblTeam.Text &= "MUMBAI INDIANS "
            End If
            If rdRR.Checked = True Then
                lblTeam.Text &= "RAJASTHAN ROYALS "
            End If
        End Sub
End Class
```

Output:



Your favorite team is CHENNAI SUPER KINGS



Your favorite team is MUMBAI INDIANS

6.8 GroupBox Control:

- A GroupBox control is a container control that is used to place Windows Forms child controls in a group.
- The GroupBox displays a frame around a group of control with or without a caption.
- Use a GroupBox to logically group a collection of controls on a form.
- The GroupBox is a container control that can be used to define group of control.
- Typical use of GroupBox is to contain a logical group of RadioButton controls.
- The purpose of a GroupBox is to define user interfaces where we can categories related controls in a group.

Properties:

Property Name	Description
AutoSize	Gets or sets a value that indicates whether the GroupBox resizes based on its contents.
FlatStyle	Gets or sets the flat style appearance of the group box control.

Location	Gets or sets the coordinates of the upper-left corner of the control relative to the upper-left corner of its container.
Margin	Gets or sets the space between controls.
Padding	Gets or sets padding within the control.
Text	Gets or sets the text associated with this control.
Dock	Gets or sets which control borders are docked to its parent control and determines how a control is resized with its parent.
Width	Sets the width of the control.
Height	Sets the Height of the control.
Background	Sets the background color of the control.

Events:

Event Name	Description
Scroll	Occurs when the user or code scrolls through the client area
Layout	Occurs when a control should reposition its child controls.
PaddingChanged	Occurs when the control's padding changes.
TextChanged	Occurs when the Text property value changes.

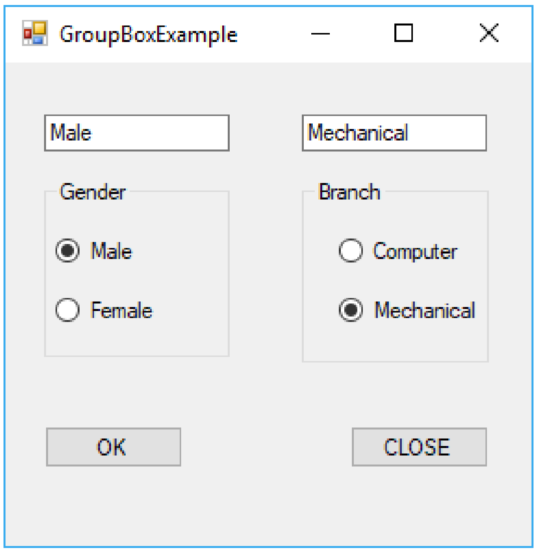
Methods:

Method Name	Description
Select()	Activates the control.
ToString()	Returns a String containing the name of the Component
OnFontChanged()	Raises the FontChanged event.

Example:

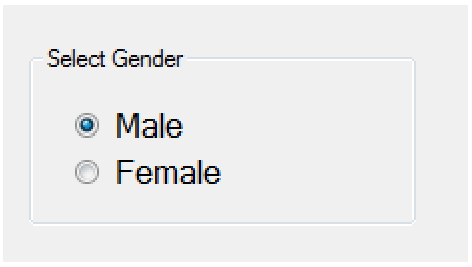
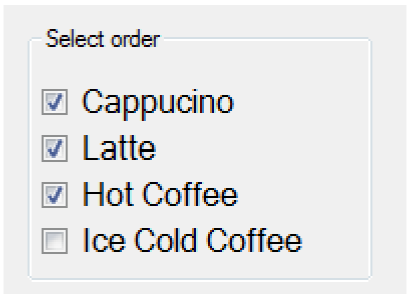
```
Public Class GroupBoxExample

    Private Sub btnOK_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles btnOK.Click
        MessageBox.Show("Data Selected")
    End Sub
    Private Sub btnCLOSE_Click(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles btnCLOSE.Click
        Me.Close()
    End Sub

    Private Sub rdMale_CheckedChanged(ByVal sender As System.Object, ByVal
    e As System.EventArgs) Handles rdMale.CheckedChanged
        txtGender.Text = rdMale.Text
    End Sub
    Private Sub rdFemale_CheckedChanged(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles rdFemale.CheckedChanged
        txtGender.Text = rdFemale.Text
    End Sub
    Private Sub rdComputer_CheckedChanged(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles rdComputer.CheckedChanged
        txtBranch.Text = rdComputer.Text
    End Sub
    Private Sub rdMechanical_CheckedChanged(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles rdMechanical.CheckedChanged
        txtBranch.Text = rdMechanical.Text
    End Sub
End Class
```

Output:



6.9 Difference between Radio Button and Check Box:

Radio Button	Check Box
Radio button or option button, is a circular control that comes in a group with other controls of the same type.	CheckBox is quite opposite and similar to Radio Button, it is square in shape, because using CheckBox you can do a multiple selection
Usually a radio button like an empty circle or a donut.	Usually a CheckBox like an empty square.
If it is clicked, it will be filled with a big dot.	If it is clicked, it will be filled with a right arrow symbol in square box.
There are only two states of radio button that is checked or unchecked.	Depending on user requirement, there are two or three states in checkbox that is checked, unchecked or intermediate.
And in every group of radio button only one can be selected by the user.	And in every group of checkbox you can select one or more checkbox.
Example in a membership form, there is always a radio button for Gender for Male and Female.	Example their properties, they have the same Checked property that when selected, will

	toggle the true or false value.
	 Select order Cappucino Latte Hot Coffee Ice Cold Coffee

7. GTU QUESTIONS:

Sr. No	Questions	Marks
	18/02/2021	
1	Define Event.	2
2	Define properties and explain in brief.	2
3	Define GotFocus event.	2
4	Explain important properties and events of Text box control.	3
5	Differentiate between Radio button and Checkbox.	3
6	Develop program using Scrollbar control.	4
7	Give Example of List Box control with Add(), AddRange(),Insert() method.	3
8	Describe POPUP menu with example.	3
9	Explain Masked Textbox control with its important properties and event.	4
10	Explain Input Box with example.	4
11	Explain Selected index changed event of Combobox with suitable example.	4
12	Explain RichTextBox, ProgressBar and PictureBox with example.	7
13	Explain Menu control with suitable example.	4
	29/10/2020	
1	List Timer control properties, method and events.	2
2	Define GotFocus event.	2
3	Compare Textbox and Richtextbox controls.	3
4	Define DropDownStyle property of ComboBox.	3
5	Develop small program using Radio button.	3
6	Develop small program using Button and Textbox.	3
7	Differentiate between Radio button and Checkbox.	4
8	Differentiate between Object oriented and Event driven programming.	4
9	Describe Listbox control with properties, method and event.	4
10	Develop program using Scrollbar control.	4
11	Describe InputBox with example.	3
12	Describe Masked Text box with example.	3
13	Write step to create MENUS in vb.net.	3
14	Describe Numeric Updown control with properties, method and event.	4
15	List properties and event of Form control. Explain any two properties.	4
	16/11/2019	
1	Explain DropDownStyle property of a combobox control.	2
2	Explain important property and event of Timer control in brief.	2
3	Explain Text and Value Properties of DateTime Picker Control with Example	3
4	Design simple calculator using numeric updown control.	3
5	Define Input box and Message box and demonstrate the use of these controls in a program.	4
6	Explain different methods of TextBox Control with example.	4
7	Write a program to change the background colour of label using three radio buttons named red, green and blue.	4
8	Give Example of ListBox control with Add(),AddRange(),Insert() method.	3
9	Differentiate between Radio button control and Check Box control.	3
10	Write down code for stop watch using Timer control	4
11	List out properties of Richtext box and explain any two with example.	4

12	Design one application using scroll bars and write code.	4
13	Explain Selected index changed event of Combobox with suitable example.	4
14	Explain Menu control with suitable example.	4
15	Explain Masked TextBox Control with suitable example.	3
	17/05/2019	
1	Write syntax of MessageBox and InputBox control.	2
2	List properties, methods and events of Windows Form object.	4
3	Differentiate between Procedure Oriented and Event Driven programming languages.	4
4	Differentiate between ListBox and ComboBox.	4
5	Create an application using text box, label and button control.	4
6	Explain CheckBox and RadioButton control.	4
7	Explain Timer control with example.	3
8	Distinguish between Textbox and Richtextbox control.	3
9	Design a calculator to find simple interest making use of VB.Net controls.	3
	28/11/2018	
1	Define Event.	2
2	Explain important properties and events of Text box control.	3
3	Write difference between Listbox control and Combo box control	3
4	Write difference between Check box control and Radio button control	3
5	Explain Timer control with its important property and event.	3
6	Explain NumericUpDown control with its important property and event.	3
7	Explain Date Time Picker control with its important property and event.	3
8	Explain scroll bar control with its important properties and event.	3
9	Explain Masked Textbox control with its important properties and event.	4
10	Explain Message Box.	4
11	Explain Input Box	4
12	Write a code for simple calculator.	7
13	Explain difference between Context menu and Main menu.	2
	04/05/2018	
1	Define and explain the syntax of Message box.	2
2	Explain DateTimePicker control.	2
3	Differentiate between TextBox and RichTextBox.	2
4	Explain Mask textbox in brief.	2
5	Explain the important Properties of timer.	2
6	Define (i) Properties (ii) Methods	2
7	Differentiate between POP and OOP.	3
8	Explain Numeric up down control with Example.	3
9	Explain GotFocus event.	4
10	Explain Button and Textbox control with example.	4
11	Explain I) visible ii) Auto size iii) Enable Properties.	3
12	Write a program making use of following controls (i) Label (ii)Textbox (iii) Button (iv) ListBox.	4
13	Differentiate Between radio button and checkbox.	4
	03/05/2017	
1	Write Code to display a message box with three buttons Yes, No and Help. Assume other parameters of message box of your own.	2

2	Write code to read one integer value from the user using InputBox.	2
3	What is the use of DateTimePicker Control.	2
4	Write four differences between TextBox and RichTextBox controls.	2
5	When should we use the Timer Control?	2
6	Differentiate procedure oriented and event driven programming.	3
7	Differentiate object-oriented and event driven programming.	3
8	Describe Button control with proper example.	3
9	Describe TextBox control with proper example.	3
10	Write one program that uses CheckBox control.	3
11	Write one program that uses RadioButton control.	3
12	Design a form with two Listboxes with multiple selection and two command buttons. Write code to move all selected items from one listbox to other listbox with help of command buttons.	4
13	Create a form with one textbox and one Listbox. Add command buttons for Add, Count and Exit. Write appropriate code for command buttons.	4
14	Differentiate between ListBox and ComboBox.	4
15	Give some examples of Formatting DateTime.	3
16	How does Context Menu differ from Main Menu?	2