Working with Form: properties, methods and events

- To create a New Window Forms Application, Select **File→New Project**. It will open one dialog box which is shown in Fig 2.1.

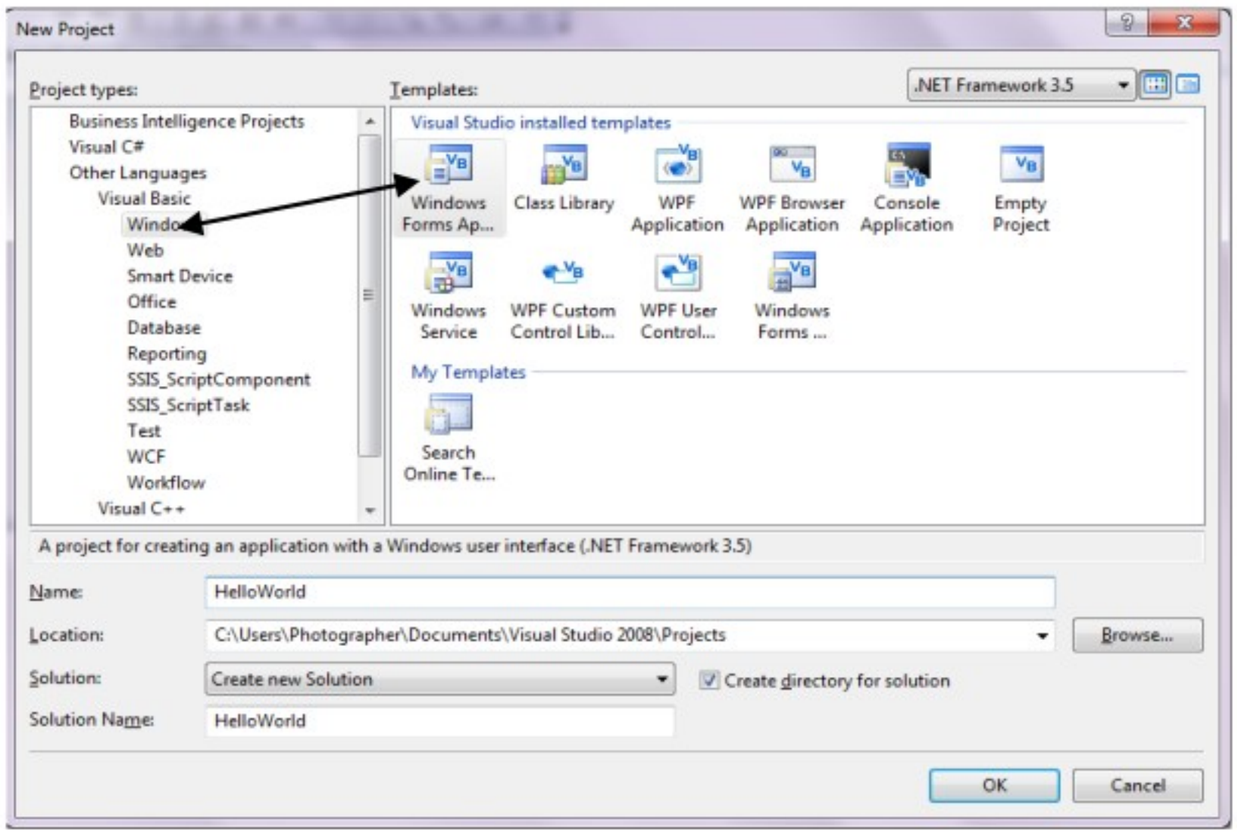


Fig 2.1 The New Project dialog box

- 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 Fig 2.2.

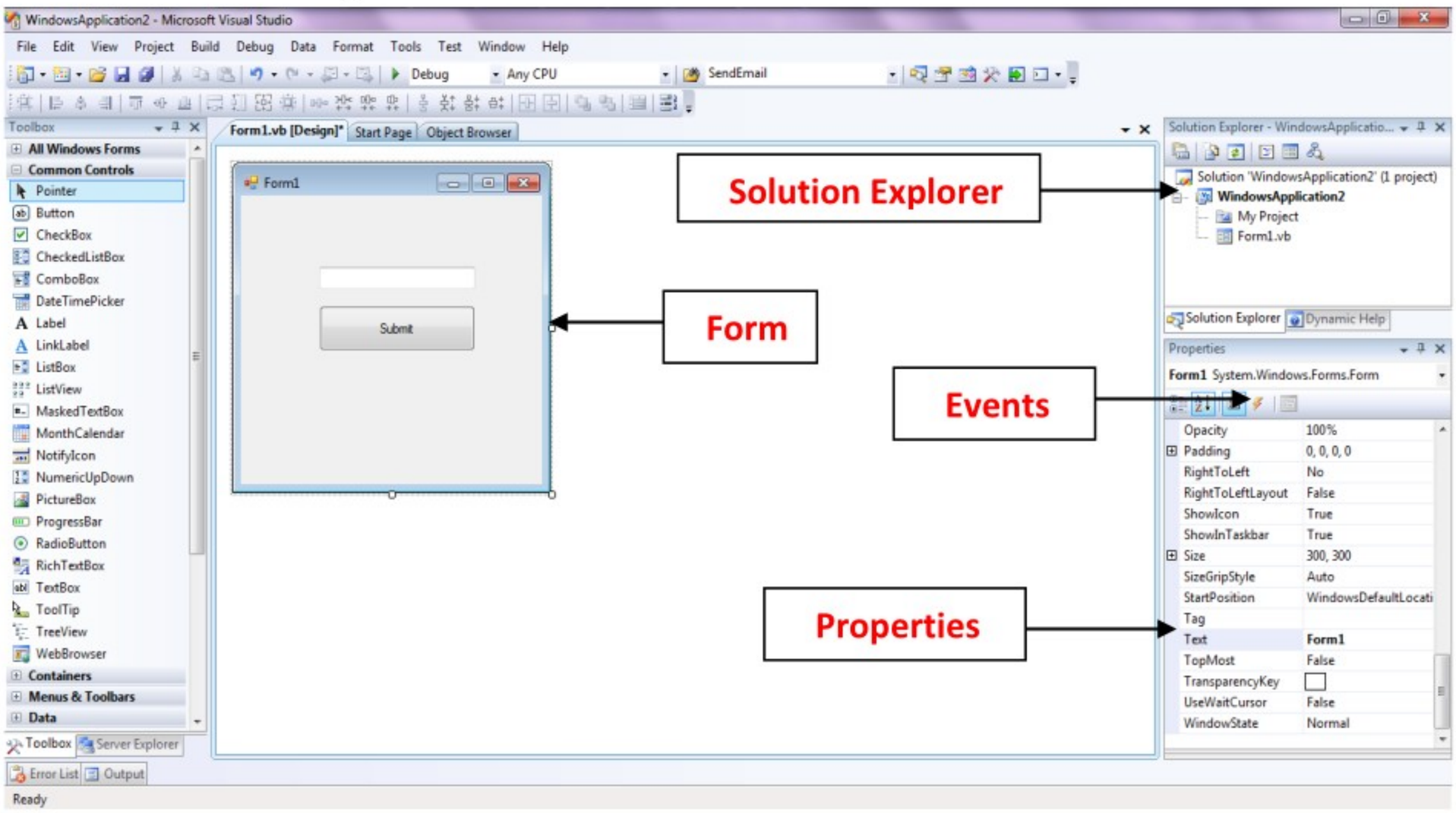


Fig 2.2 Window Form with a name Form1

- 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 Fig 2.3:

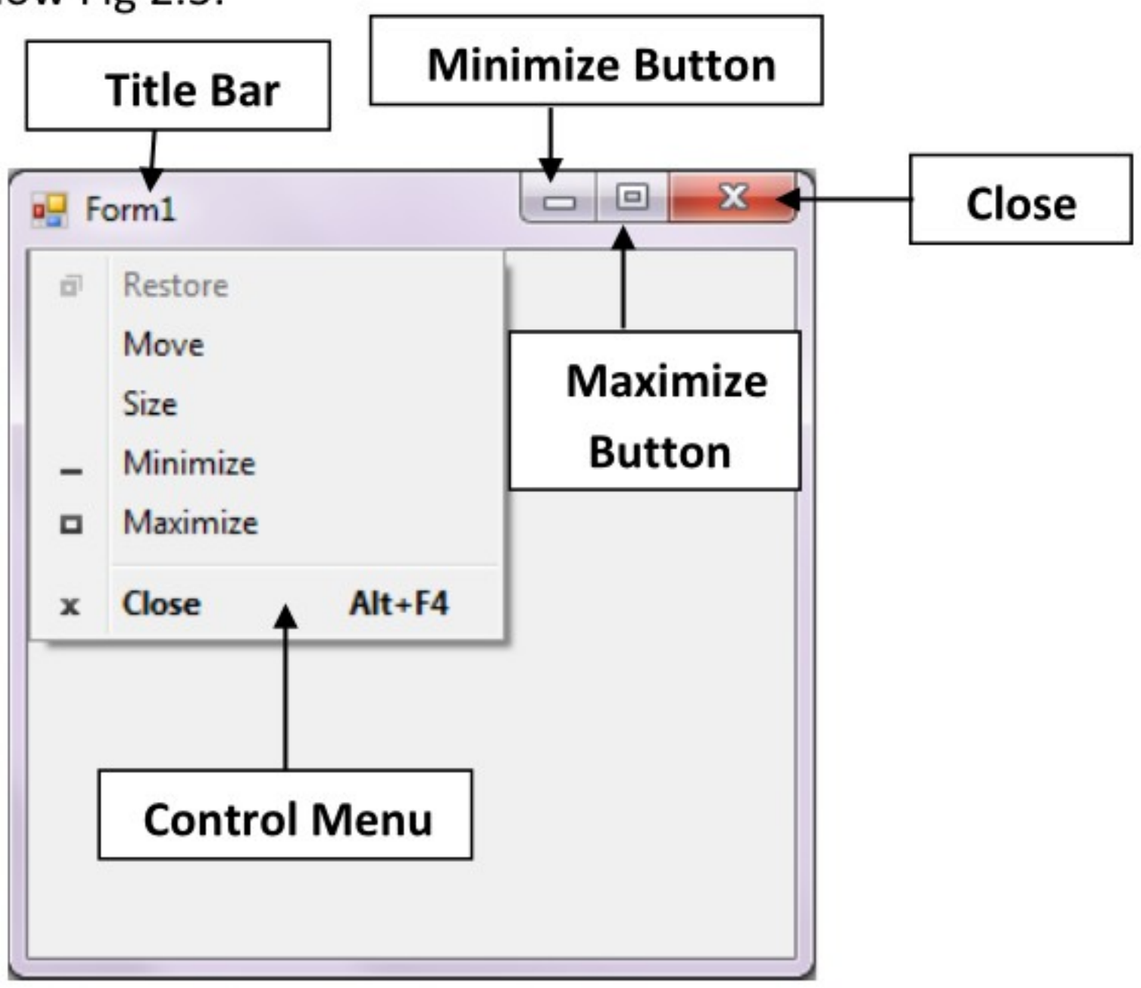


Fig 2.3 Default window form designer

- 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.

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 specify 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.
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.
MaximumSize	It specifies the maximum height and width of the window you maximize.
IsMDIContainer	By default, this property is False and you can set it to True for generating MDI application.

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.
Hide	Conceals the control from 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.
Select	Activates the control.

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.

Differentiate POP, OOP and Event Driven:

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

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 displays 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("InputBox Message", "InputBox title", "1", 100, 100)
```

- Example of the InputBox:
  - Design a form by drag and drop one label control on the form (see Fig 2.4) and double click on the form to write a code which is given below:

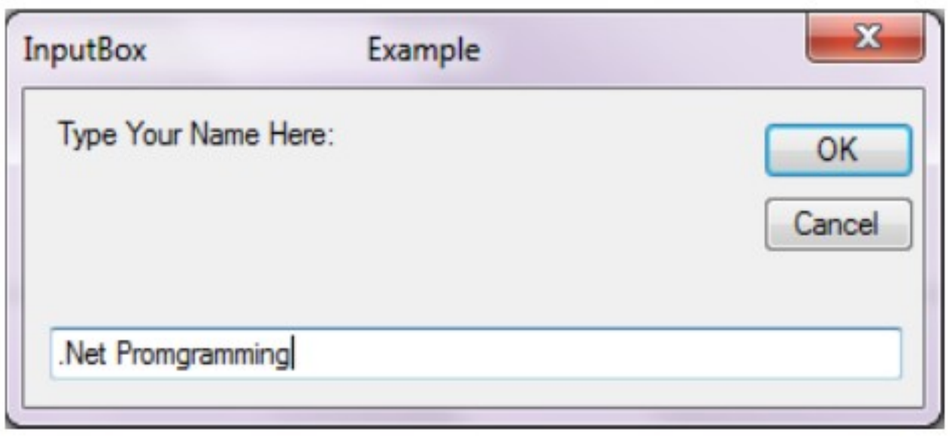


Fig 2.4 Input Dialog Box

Fig 2.5 Form Dialog Box with Output

```
Private Sub Form1_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
            lblInfo.Text = ""
        Else
            lblInfo.Text = "Your Name is: " & strUserName
        End If
End Sub
```

MessageBox

- A MessageBox is a predefined dialog box that displays application-related information to the user. Message boxes are also used to request information from the user. **MessageBox.Show()** displays a dialog box.

- It interrupts the user. It immediately blocks further interaction. It requires only one Function call.
  - 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)
  - 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
```

Common Tool Box Controls in VB.NET:

- **VB.Net** contains a very rich amount of the controls for the windows application.
- Every controls have three important elements which are **Properties, Methods** and **Events**
- **Properties** means all the controls can resized, customized and moved. A property is a value or characteristics held by the particular control.
- **Methods** cause an object to do something. Methods are blocks of code designed into controls that tell the control how to do things.
- An **Event** means a signal that informs an application that something important has occurred.
- Some of the **common Properties** and **Events** of all controls are as below. **Common Properties of all controls** which are listed below:

Property Name	Description
Text	Display the name of the control on the windows forms.
Name	Contains the name of the controls by which the control can be identified.
Minimum Size	With the help of this property we can manage the minimum size of the control.
Maximum Size	With the help of this property we can manage the maximum size of the control.
Size	With the help of this property we can manage the size of the control.
Modifier	To handle the access level of control public, private, protected or friend.
Font	To set the font style of that control.
Anchor	To set anchor property of the control.
TabIndex	To set the tab index of the particular control.
Visible	Control should be visible or not visible we can set with this property.

- **Common Events of all controls** which are listed:

Event Name	Description
Click	When we click on the control and we have to execute any code that code will be written in this event.
MouseClick	When mouse will be clicked on the control and we have to execute any code that code will be written in this event.
DoubleClick	When we double click on the control and we have to execute any code that code will be written in this event.
DragLeave	When we remove the mouse from the control and at that time if any code have to be executed that code will be written in this event.
SizeChanged	When we changed the size of the control and at that time if any code have to be executed that code will be written in this event.
StypeChanged	When we changed the style of the control and at that time if any code have to be executed that code will be written in this event.

- Some other controls of windows forms are as following:

Button Control

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

Properties:

Property Name	Description
BackColor	It gets or sets the background color of the control.
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.
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
FlatStyle	It gets or sets the flat style appearance of 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.
Validated	Occurs when the control is finished validating.
Click	Occurs when user clicks the Button.
FontChange	Occurs when font is changed.

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.

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.

Properties:

Property Name	Description
Autosize	It gets or sets a value specifying if the control should be automatically resized to display all its contents.
BorderStyle	Gets or sets the border style for the control.
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.

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.

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.

Properties:

Property Name	Description
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

	TextBox control.
WordWrap	Indicates whether a multiline text box control automatically wraps words to the beginning of the next line when necessary.
Readonly	By default, this property is True and you can set it to False to make the text Readonly.

Events:

Event Name	Description
TextAlignChanged	Occurs when the TextAlign property value changes.
TextChanged	Occurs when the text in the textbox changes.

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.

Common Example of Textbox, Label and Button:

Example:

```
Private Sub btnShowMessage_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnShowMessage.Click
        lblMessage.Text = txtMessage.Text
End Sub
```

Output:

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.**

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

Width	Sets the width of the control
Height	Sets the Height of the control
Background	Sets the background color of the control
CurrentValue	Represents the currently selected value in a control.
Minimum and Maximum Range	Used to set the minimum and maximum values of a control
Increment	Used to set the value that is increased or decreased when up and down buttons are clicked

Events:

Event Name	Description
Scroll	Occurs when the user or code scrolls through the client area

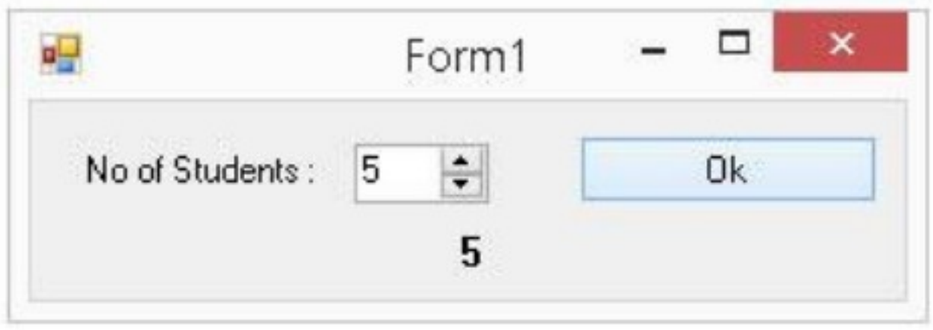
Methods:

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.

Example:

```
Private Sub btnShow_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles btnShow.Click
        lblAnswer.Text = NumericUpDown1.Value.ToString()
End Sub
```

Output:



Checkbox Control

- The CheckBox **control allows the user to set true/false or yes/no type options**.
- When a check box is **selected** it has the value **True**, and when it is **cleared**, it holds the value **False**.

Properties:

Property Name	Description
CheckAlign	Gets or sets the horizontal and vertical alignment of the check mark on the check box.
Checked	Gets or sets a value indicating whether the check box is selected.
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:

```
Private Sub btnOK_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnOK.Click
        lblCity.Text = "You have selected "
        If chkRajkot.Checked = True Then
                lblCity.Text &= "Rajkot "
        End If
        If chkJunagadh.Checked = True Then
                lblCity.Text &= "Junagadh "
        End If
        If chkBaroda.Checked = True Then
                lblCity.Text &= "Baroda "
        End If
        If chkAhmedabad.Checked = True Then
                lblCity.Text &= "Ahmedabad "
        End If
End Sub
```

Output:

RadioButton Control

- It enables the user to select a single option from a group of choices.

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.

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.

Methods:

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

Example:

```
Private Sub btnOK_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnOK.Click
        lblCity.Text = "Your favorite city is "
        If rdRajkot.Checked = True Then
                lblCity.Text &= "Rajkot "
        End If
        If rdJunagadh.Checked = True Then
                lblCity.Text &= "Junagadh "
        End If
        If rdBaroda.Checked = True Then
                lblCity.Text &= "Baroda "
        End If
        If rdAhmedabad.Checked = True Then
                lblCity.Text &= "Ahmedabad "
        End If
End Sub
```

Output:

GroupBox Control

- A GroupBox control is a container control that is **used to place Windows Forms child controls in a group**.
- 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.
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

Methods:

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

Example of all controls are shown below, design a Student Registration form with the help of the all controls which we had learned.

- Open the **visual studio→ Click on the new project → select the windows application → select the proper destination to save the project** and give proper title to the project
- Now drag and drop the following the controls to the form and set its text properties and name of the controls as shown below:

Control Name	Text Property	Name
Form	Registration Form	frmStudentRegistration
Group Box	Registration Form	gbRegistration
Label	Name of Student:	lblName
	Number languages Known:	lblNumberofLanguages
	Gender:	lblGender
	Select Languages:	lblLanguages
Numeric Updown		nudNumbers

Radio Button	Male	rbmale
	Female	rbFemale
Check Box	.Net	chkDotnet
	Java	chkJava
	Php	chkPhp
	C/C++	chkC
	Phython	chkPhython
	Ruby	chkRuby
Button	Save	btnSave
	Cancel	btnCancel

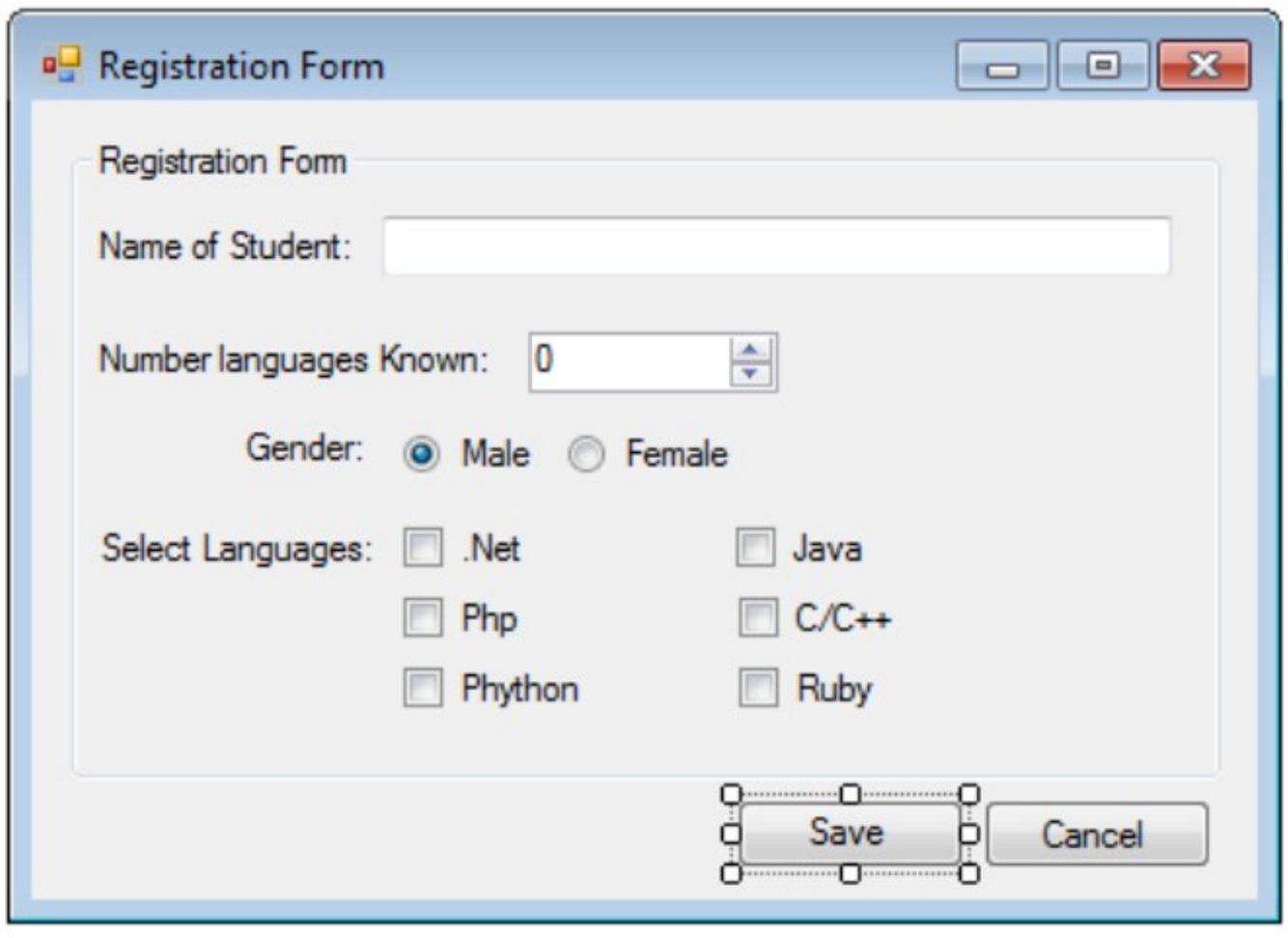


Fig 2.6 Registration Form

Write the following code on the click event of the Save button,

```
Private Sub btnSave_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
btnSave.Click
        If txtName.Text = "" Then
                MessageBox.Show("Please Enter the Student Name", "Error Occured",
                MessageBoxButtons.OK, MessageBoxIcon.Error)
        Else
                MessageBox.Show("Resistered Successfully", "Thank You", MessageBoxButtons.OK,
                MessageBoxIcon.Information)
            Me.Close()
        End If
End Sub
```