The list of controls we are discussed:

- Label
- TextBox
- Button
- RadioButton
- CheckBox
- GroupBox
- ListBox
- ComboBox
- Timer
- MenuScript
- ContextMenuScript
- ColorDialog
- RichTextBox
- OpenFileDialog
- SaveFileDialog
- FontDialog

Control:

The control is a GUI component that is the primary source for windows application. Based on the functionality controls are classified into two types.

1. Active Controls
   A control that allows the user interaction during the execution of a form is an active control.
   Ex: Textbox, checkbox etc.
2. Passive Controls
   A control that does not allow the user to interact during the execution of a form is a passive control.
   Ex: Labels, Groupbox, Timer etc

The GUI is meaningful with the combination of active and passive controls.

Label

A passive control that is used to label the other control to make the interface more meaningful. It is an object reference to the class **System.Windows.Forms.Label.**

Properties:

Property	Description
Name	The property that uniquely identifies the label control. This is the most important property in every control in order to unique identification by the C# compiler. The name can be changed without giving space and it can be prefixed with “Lbl”.
BackColor	Sets the background color. Default gray.
Autosize	Determines if the label is automatically resized according to the text.
BorderStyle	Specifies the border for the label.
Font	Sets the font properties style, size etc.
ForeColor	Sets the forecolor. By default is the black.
Image	Specifies the path of the image that can be displayed within the label.
Imagealign	Specifies the alignment of the image
Text	Sets the text displayed over the label.
Dock	Specifies how the controls can be positioned within the form.
Textalign	Specifies the alignment of the text.

Visible	Determines if the label is visible or invisible by default it is true.

Textbox

An Interactive control that is used to enter the text and read the text. It can be used to accept maximum of 2 Billion characters. It is an object reference to the class **System.Windows.Forms.TextBox**. It is used in windows application as an input field.

Properties:

Property	Description
Name	The property that uniquely identifies textbox. It can be prefixed with Txt followed by any name.
Multiline	Determines if the textbox can accept multiple line of text or not. By default it is **false**.
CharacterCasting	Specifies how the character case is maintained within the textbox. Normal, Upper, Lower
PasswordChar	Specifies the character that masks the actual character. It can be used to make a textbox to be a password field.
Readonly	Determines if the text is available read as well as write and read only. By default it is **false**.
Scrollbars	Specifies the type of the scrollbars displayed when the multiline property is **true**.
Textalign	Specifies the alignment of the text. By default it is **left**.

Methods:

Name	Description
AppendText(string)	Appends the given text already existing text.
Clear()	Clears all the text.
Copy()	Copies the selected text.
Cut()	Moves the selected text
Hide()	Hides the control.
Paste()	Gets the copied/cut text.
Select(start,length)	Select the given text from the given starting position to the given length.
SelectAll()	Select the complex text.
Show()	Displays the hidden control.

Button

An Interactive control that is used to execute a set of commands when the button is selected by the user. It is also called as "Command Button". Most of the user interaction in windows based application is done through a button or menu. It is an object reference to the class **System.Windows.Forms.Button**.

Properties:

Property	Description
Name	The property that uniquely identifies button.
BackgroundImage	Specifies the path of the image that can be set as the background. It supports all valid picture format file.
Image	Specifies the path of the image that is displayed over the button.
ImageAlign	Specifies the alignment of the image.
Text	Specifies the text to be displayed over the button. The text can make use of '&' (ampersand) character. So that the respective character is underline and the user can interact access key(Alt followed by key)
TextAlign	Specifies the alignment of the text.

Methods:

Name	Description
Hide()	Hides the control.
Show()	Displays the hidden control.

MessageBox

A Predefined static class that is used to display an alert to the user giving information messages, warnings , errors and questions. Most of the window based applications make use of the MessageBox. It contains an overloaded method called show().

Syntax:

MessageBox.show(text,[Title],[MessageButtons],[MessageBoxIcon])

Where "text" is a valid string that can be up to 1024 characters. "Title" is a string that is displayed in the title bar. "MessageBoxButtons" specifies the type of the buttons to be placed over the messagebox. "MessageBoxIcon" specifies the type of the icon to be displayed over the messagebox indicating the type of alert.

MessageBoxIcon.Error

Error. An error or problem has occured

MessageBoxIcon.Exclamation

Warning icon. The UI is presenting a condition that might cause a problem in the future

MessageBoxIcon.Information

Information icon. The UI is presenting useful information.

MessageBoxIcon.Question

Question icon. Do not use this icon for questions.

Ex: Create an Interface to accept the student details by means of the messagebox

Radio Button

An Interactive control that is used to provide the list of options to the user so that the user can select only one option. Radio Button supports mutually exclusive case of option. It is an object reference to the class **System.Windows.Forms.RadioButton**

Properties:

Property	Description
Name	The property that is uniquely identifies ratio button
Checkalign	Specifies the alignment of the checkbox
Checked	Determines the state of the checkbox Return true, if the radio button is selected. Return false, if the radio button is not selected.
Text	Sets the label to be selected over the radiobutton
Textalign	Specifies the alignment of the text
Image	Specifies the path of the image that can be displayed along with the radio button.
Imagealign	Specifies the alignment of the image

CheckBox

An Interactive control that is used to provide the list of options to the user, so that the user can select more than one option. It is an object reference to the class **System.Windows.Forms.CheckBox**

Property	Description
Name	The property that is uniquely identifies checkbox
Checkalign	Specifies the alignment of the checkbox
Checked	Determines the state of the checkbox Return true, if the radio button is selected. Return false, if the radio button is not selected.
CheckState	Determines the state of the checkbox. It can be any one of the following constants. Unchecked- Displays an empty checkbox Checked- Displays a checkmark thick black color Interminate- Displays a light gray colored checkmark.

Note:

- When the check state is unchecked, checked property returns false.
- When the check state is check or interminate, check property returns true.

GroupBox

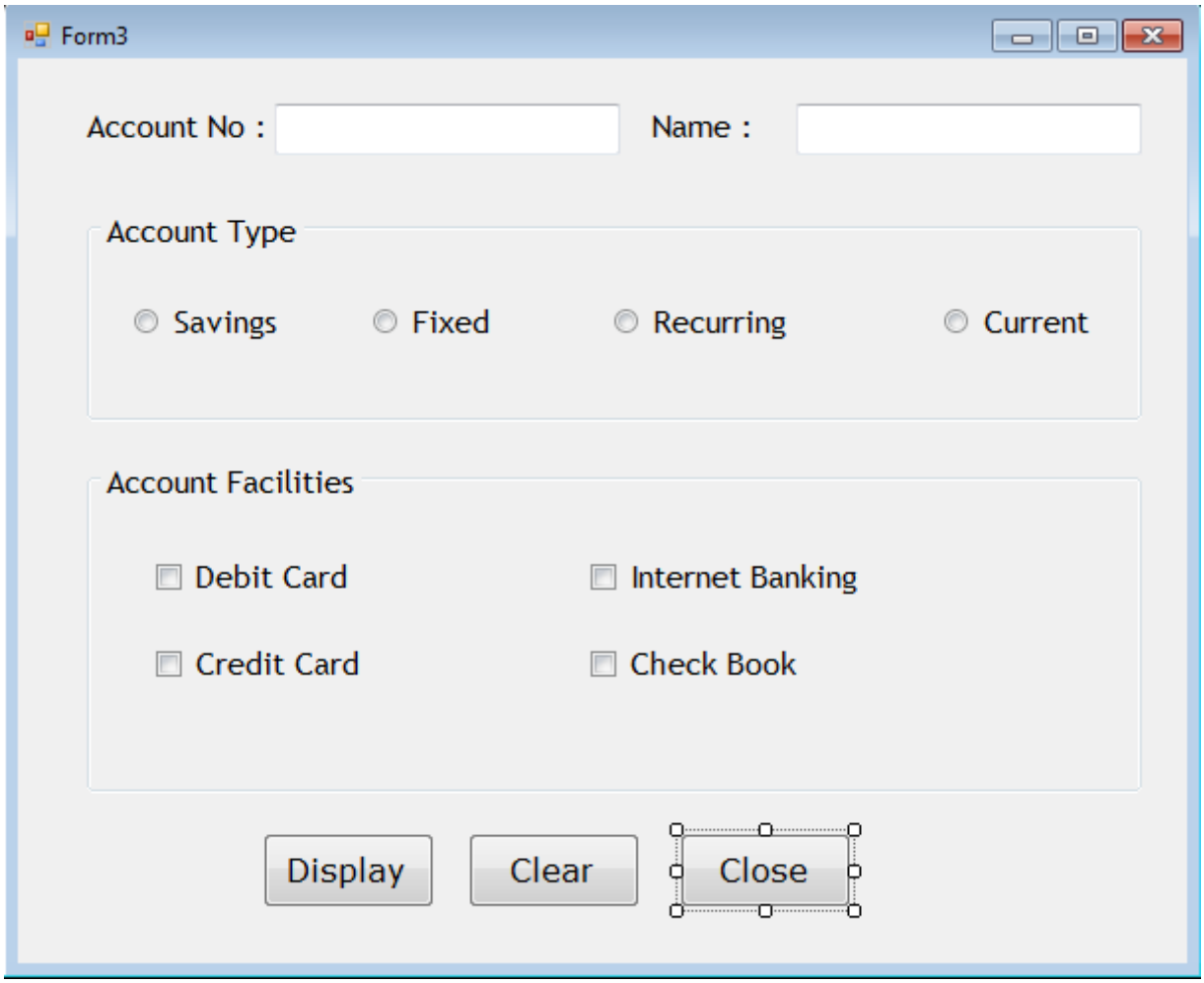
A passive control that is used to group the controls that fall into a particular category or group. It is an object reference to the class **System.Windows.Forms.GroupBox**

Property	Description
Name	The property that is uniquely identifies groupbox
Dock	Specifies the docking state of the control.
Text	Sets the caption for the groupbox

Example

Create an interface to work with radio button, checkbox and groupbox.

Step 1: Create the following interface using windows forms class



Step 2: Write the following logic to the events

```
//Display
        string s;
        s="Account No:"+ textBox1.Text+"\n";
        s = s + "Name:" + textBox2.Text + "\n";
        if (radioButton1.Checked == true)
           s = s + "Account Type:" + radioButton1.Text + "\n";
        else if(radioButton2.Checked==true)
           s = s + "Account Type:" + radioButton2.Text + "\n";
        else if(radioButton3.Checked==true)
           s = s + "Account Type:" + radioButton3.Text + "\n";
        else
           s = s + "Account Type:" + radioButton4.Text + "\n";

        s = s + "Account Facilities are..." + "\n";
        if (checkBox1.Checked == true)
           s = s + checkBox1.Text + "\n";
        if (checkBox2.Checked == true)
           s = s + checkBox2.Text + "\n";
        if (checkBox3.Checked == true)
           s = s + checkBox3.Text + "\n";
        if (checkBox4.Checked == true)
           s = s + checkBox4.Text + "\n";
        MessageBox.Show(s, "Account Details", MessageBoxButtons.OK, MessageBoxIcon.Information);

//Clear
            foreach (Control ctrl in this.Controls)
                if (ctrl is TextBox)
                    ctrl.Text = "";
            textBox1.Focus();

            radioButton1.Checked = false;
            radioButton2.Checked = false;
            radioButton3.Checked = false;
            radioButton4.Checked = false;

            checkBox1.Checked = false;
            checkBox2.Checked = false;
            checkBox3.Checked = false;
            checkBox4.Checked = false;

            textBox1.Focus();
//close
            this.Close();
```

Disadvantage of RadioButton:

- The user selects one option among the number of options.
- If we want to enter 100 options it is a little bit time taken process, Hence we go to “ListBox”.

ListBox

An Interactive control that is used to display the list of values to be selected by the user. It can be used to display maximum of 32768 values. All the items are treated as an array of the string with the starting index as zero.

	Property	Description
1	Name	The property that is uniquely identifies listbox.
2	Items	Specifies the collection of items to be displayed in the listbox.

The items collection has the following properties & methods.

Add(String)	Add the given string to already existing items.
Remove(string)	Removes an item based on a string value.
RemoveAt(index)	Removes an item list based on its index.
Clear()	Clear all the items in the listbox
Insert(index,string)	Inserts a new item in between the existing items.
IndexOf(string)	Returns the index of the given string value.
Count	Return the total number of items present in the list.

3	HorizontalScrollbar	Determines if a horizontal scrollbar is displayed when the list item goes beyond the right boundary.
4	SelectionMode	Indicates how the list items can be selected.
5	Multicolumn	Determines if the values are displayed in multicolumn.
6	Sorted	Determines if the list of items are arranged in alphabetical order.

Runtime Properties:

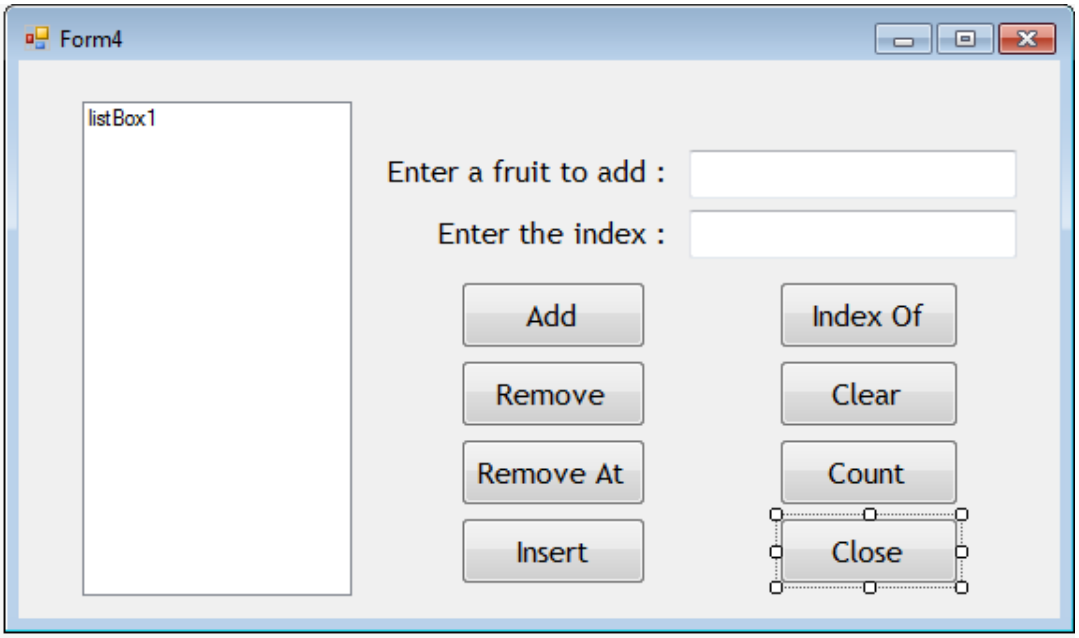
These properties are accessed while writing the code.

1. SelectedIndex: Returns the index of the selected item.
2. SelectedItem: Returns the string value of the item selected.

Example

Create an interface to work with listbox properties & methods

Step 1: Create an interface using forms class



Step 2: Add the click events for the above interface.

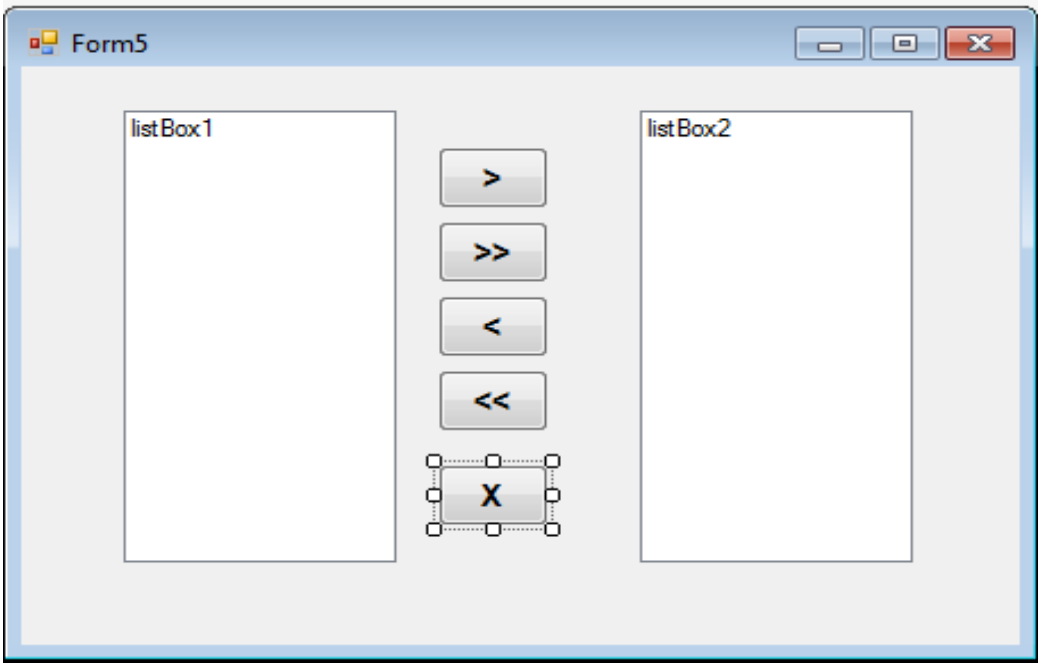
```
        string fruit = "";
         int i;

//Add
            fruit = textBox1.Text;
            listBox1.Items.Add(fruit);
//Remove
            fruit = listBox1.SelectedItem.ToString();
            listBox1.Items.Remove(fruit);
//Remove At
            i = listBox1.SelectedIndex;
            listBox1.Items.RemoveAt(i);
//insert
            fruit = textBox1.Text;
            i = int.Parse(textBox2.Text);
            listBox1.Items.Insert(i, fruit);
//index of
            fruit = textBox1.Text;
            i = listBox1.Items.IndexOf(fruit);
            MessageBox.Show("The index of fruit" + fruit + " is" + i.ToString());
//clear
            listBox1.Items.Clear();
//count
            i = listBox1.Items.Count;
            MessageBox.Show("The no.of fruits are "+i,"Total Fruits",
MessageBoxButtons.OK,MessageBoxIcon.Information);
//close
            this.Close();
```

Example 2

Create an interface to add items from one listbox to another listbox.

Step 1: Create an interface using forms class



Step 2: Add the click events for the above interface.

```
       int i;
        string s = "";
//load
            listBox1.Items.Add("Apple");
            listBox1.Items.Add("Banana");
            listBox1.Items.Add("Grape");
            listBox1.Items.Add("Orange");
            listBox1.Items.Add("Pineapple");
            listBox1.Items.Add("Strawberry");
//>
            s=listBox1.SelectedItem.ToString();
            listBox2.Items.Add(s);
            listBox1.Items.Remove(s);
//<
            s = listBox2.SelectedItem.ToString();
            listBox1.Items.Add(s);
            listBox2.Items.Remove(s);
//>>
            for (int i = 0; i < listBox1.Items.Count; i++)
            {
                listBox2.Items.Add(listBox1.Items[i]);
            }
            listBox1.Items.Clear();
//>>
            for (int i = 0; i < listBox2.Items.Count; i++)
            {
                listBox1.Items.Add(listBox2.Items[i]);
            }
            listBox2.Items.Clear();
//X
            this.Close();
```

ComboBox

An Interactive control that is used to display the list of items to be selected by the user. It can be used to store 32768 values. All the items are treated as an array of string with the starting index is zero. It is a combination of textbox & listbox. The user can type of character & search the values starting with the given character.

Property	Description
Name	The property that is uniquely identifies combobox.
Dropdownitems	Specifies the no.of items to be displayed in the drop down list.
Dropdownstyle	Specifies the appearance & control of the combobox it can be anyone of the following constants. **Simple**- Displays the control as the textbox and the user can select the list of item using up and down arrow keys. **Dropdown**- Displays an dropdownlist that allows the item for editing. This is a default setting. **Dropdownlist**- This makes the combobox behave like a listbox that does not allow the items for editing.
Text	Returns the text for the list item selected by the user.

ComboBox	ListBox
There is no horizontal scrollbar for combobox	There is a horizontal scrollbar for listbox.

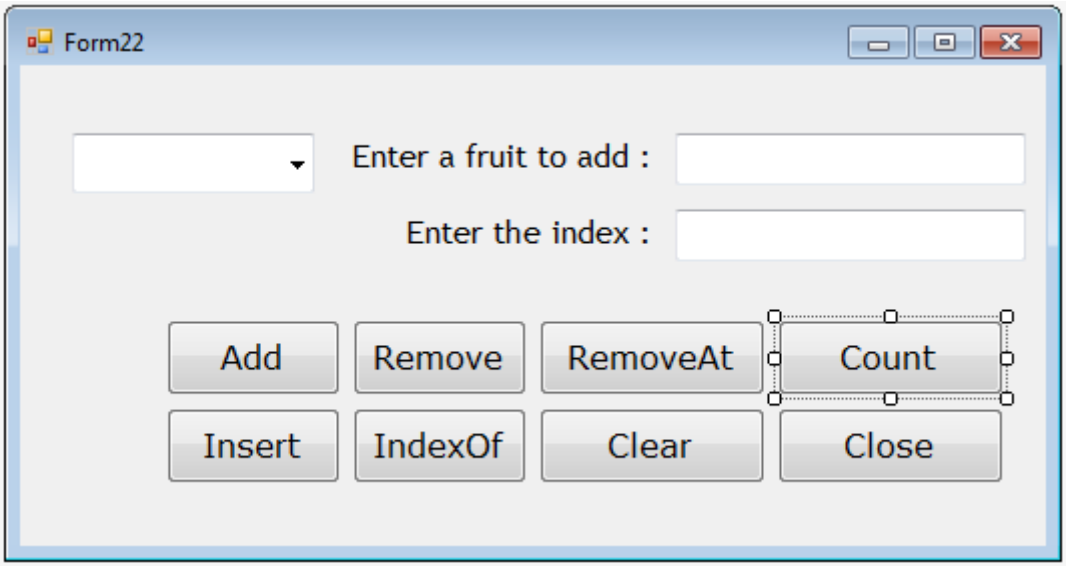
There is no multiline property for combobox

There is a multiline property for listbox.

Note: The rest of the properties & methods are same as the listbox.

Create an interface to add items from one listbox to another listbox.

Step 1: Create an interface using forms class



Step 2: Add the click events for the above interface.

```
//Add
            fruit = textBox1.Text;
            comboBox1.Items.Add(fruit);
//Remove
            fruit = comboBox1.SelectedItem.ToString();
            comboBox1.Items.Remove(fruit);
//RemoveAt
            i = comboBox1.SelectedIndex;
            comboBox1.Items.RemoveAt(i);
//Insert
            fruit = textBox1.Text;
            i = int.Parse(textBox2.Text);
            comboBox1.Items.Insert(i, fruit);
//Index of
            fruit = textBox1.Text;
            i = comboBox1.Items.IndexOf(fruit);
            MessageBox.Show("The Index of the fruit:" + fruit + " is" + i.ToString());
//Clear
            comboBox1.Items.Clear();
//Close
            this.Close();
//Count
            i = comboBox1.Items.Count;
            MessageBox.Show("The No.of players are :" + i, "Total Players", MessageBoxButtons.OK,
MessageBoxIcon.Information);
```

Timer

A control that is used to execute a set of statement at regular intervals of the time.

Property	Description
Name	The property that is uniquely identifies timer.
Enable	Determines if the timer is enabled or disabled.
Interval	Specifies the frequency of elapsed events. It takes the numeric value in the following manner. 1000ms=1sec 2000ms=2sec

Note:

- Timer control is not visible during the runtime.
- It has tick event, this event executes according to the interval property specified.

Create an interface to display the running time:

Step 1: Create an interface using forms class
Step 2: Select the timer and set the enabled property to true.
Step 3: Drag and drop one label and set the properties
Backcolor -Black
Forecolor -Lime
Step 4: Enter the following code inside timer_Tick block.

```
label1.Text = DateTime.Now.ToString();
this.Text = DateTime.Now.ToString();

int red, green, blue;
Random a = new Random();
Random r = new Random();
Random g = new Random();
Random b = new Random();
red = r.Next(0, 225);
green= r.Next(0, 225);
blue = r.Next(0, 225);
this.BackColor = Color.FromArgb(red, green, blue);
```

Menu

A menu is a list of options provided by the user so that the user selects menu option. In window based applications, menu play a very important role in handing the applications.
There are two types of menu.

1. Main menu or Pull down menu
2. Popup or context menu

1. Main menu or Pull down menu

This menu appears just below the title bar when selected display a pop list displaying the menu options to select by the user. This type of menu can be created in .NET using **menuscript** control. This control is available under menus & toolbar category of the toolbox.

MenuScript

This control is use to design a menu.

Property	Description
Name	The property that is uniquely identifies menu.
Image	Specifies the path of the image to be displayed along with the menu option.

Imagealign	Specifies the alignment of the image.
Shortcutkey	Specifies the shortcut to be attached with the menu option.
Text	Specifies the text for the menu option
Textalign	Specifies the alignment of the text.
Textdirection	Specifies the direction of the text within the menu.
TextImageRelation	Specifies the relative location of the image to the text no the item.

2. Popup or context menu

It is a floating menu that appears anywhere within the window, when the right button of the mouse is pressed. It can be created using **contextmenustript** control that is available under menus & toolbars of category of toolbox.

ContextMenuStript

This control is used to create a popup menu or context menu.

Property	Description
Name	The property that is uniquely identifies contextmenustript control.

Note: The rest of the properties are same as menustript control.

Color Dialog

This control is used to display a color dialog based on which the color can be applied for the controls.

Properties

Property	Description
Name	The property that is uniquely identifies color dialog.
AllowFullOpen	Enables and disable the define custom color buttons.
Color	Returns the color selected in the dialog box.
FullOpen	Controls weather the custom color selection of the dialog box is initially displayed.

Methods

ShowDialog()

This method is used to display the dialog box during runtime.

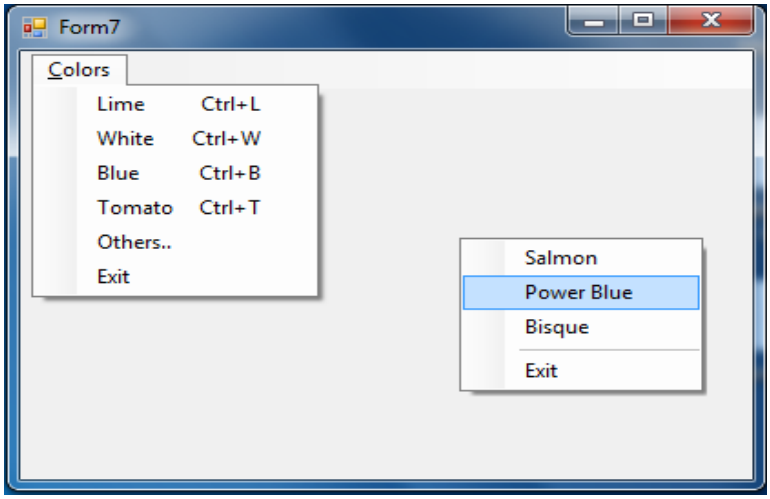
Example

Create a menu driven interface that changes the background color of the form when the menu option is selected.

Step 1: Create an interface using forms class.

Properties:

Form	ContextMenuStript-ContextMenuStript1 WindowState- Maximum
ContextMenuStript1	Salmon Power Blue Bisque - Exit
MenuStript	Orange Lime White Tomato Other.. Exit




```
//Orange
            this.BackColor = Color.Orange;
//White
            this.BackColor = Color.White;
//Blue
            this.BackColor = Color.Blue;
//Tomato
            this.BackColor = Color.Tomato;
//Others...
            colorDialog1.ShowDialog();
            this.BackColor = colorDialog1.Color;
//Exit
            this.Close();
//PowerBlue
            this.BackColor = Color.PowderBlue;
//Salmon
            this.BackColor = Color.Salmon;
//Bisque
            this.BackColor = Color.Bisque;
//Exit
            this.Close();
```

TextBox Vs RichTextBox

TextBox	RichTextBox
• It accept maximum 2 billion characters only	• It Accept unlimited characters
• Text is stored in ASCII format. (extension.txt)	• The Text is stored in Rich text format. (extension. Rft)
• Multiline property is set to false by default.	• Multiline property is set to true by default.
• It cannot allow any graphic content like images etc.	• It allows the graphic context along with text.
• It is used to develop notepad style applications	• It is used to develop MS Word style of applications.

RichTextBox

An interactive control that is used to enter the text and read the text from files.

Property	Description
Name	The property that is uniquely identifies RichTextBox.
Dock	Specifies the docking position of the control.
Multiline	Determines if the textbox can accept multiple lines or not. By default it is true.
Scrollbars	Specifies how the scrollbars are displayed for the control. By default it is both.
ShowSelectionMargin	Turns on or off the selection margin to be displayed with in the textbox.

OpenFileDialog

This control is used to File> open dialogbox. This is commonly available in all window based applications.

Property	Description
Name	The property that is uniquely identifiesOpenFileDialog.
CheckFileExists	Determines if the file is present or not
CheckPathExists	Determines if the path for the selected file exist or not. By default both are true.
Filename	Returns the name of the file selected.
Filter	Specfies the filter pattern for the files to be displayed in the dialogbox. Default (*.*)

ShowDialog()

A method that is used to display the dialogbox.

SaveFileDialog

This control is used to display file>save as dialogbox.

Property	Description
Name	The property that is uniquely identifies SaveFileDialogbox.

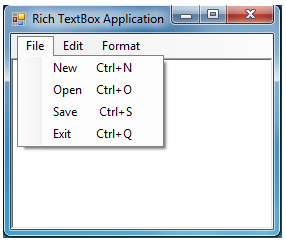
The rest of the properties and methods are same as OpenFileDialog.

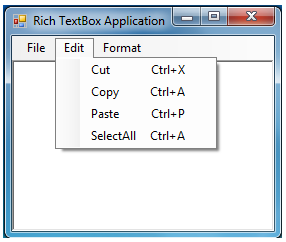
FontDialog

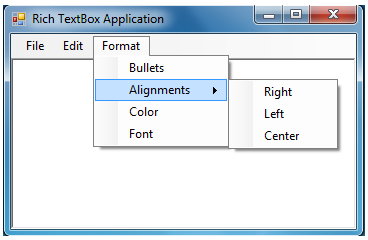
Property	Description
Name	The property that is uniquely identifies FontDialog.
Color	Determines the color of the font. By Default it is Black.
Font	Returns the font selected in the dialogbox.
ShowApply	Determines if the applied button is displayed in the dialogbox. By default it is false.
ShowColor	Determines if the color choice for the font is displayed. By default it is false.
ShowEffects	Determines if the font properties are affected in a preview. By default it is true.

Example

Create an interface to develop a menu driven interface, that can store or save the file and read the file using rich textbox contain







```
string Filename = "";
//Color
            colorDialog1.ShowDialog();
            richTextBox1.SelectionColor = colorDialog1.Color;
//New
            richTextBox1.Clear();
            richTextBox1.Focus();
//Open
            openFileDialog1.ShowDialog();
            Filename = openFileDialog1.FileName;
            richTextBox1.LoadFile(Filename);
//Save
            saveFileDialog1.ShowDialog();
            Filename = saveFileDialog1.FileName;
            richTextBox1.SaveFile(Filename);
            MessageBox.Show("File Saved", "Save", MessageBoxButtons.OK,MessageBoxIcon.Information);
//Exit
            this.Close();
//Cut
            richTextBox1.Cut();
//Copy
            richTextBox1.Copy();
//Paste
            richTextBox1.Paste();
//SelectAll
            richTextBox1.SelectAll();
//Bullets
            richTextBox1.SelectionBullet = true;
//right
            richTextBox1.SelectionAlignment = HorizontalAlignment.Right;
  //left
            richTextBox1.SelectionAlignment = HorizontalAlignment.Left;
//Center
            richTextBox1.SelectionAlignment = HorizontalAlignment.Center;
//font
            fontDialog1.ShowDialog();
            richTextBox1.SelectionFont = fontDialog1.Font;
```

Rich Textbox control has some common properties & methods just like a textbox.
Clear(), Cut(),Copy(),SelectAll(),Undo(),Redo() etc.

In addition to the above method, it has the following methods.

LoadFile(path)	Load the give filename by displaying contents of the respective file
SaveFile(path)	Saves the contents of the richtextbox into a file

There are some important properties associated with rich text box.

SelectionAlignment	Specifies the alignment for the selected text.
SlectionColor	Specifies the color for the selected text.
SelectionBullet	Determines if the selected text is displayed as bulleted text.
SelectionFont	Specifies the form for the selected text.