C# Label Control

Labels are one of the most frequently used C# control. We can use the Label control to display text in a set location on the page. Label controls can also be used to add descriptive text to a Form to provide the user with helpful information. The Label class is defined in the System.Windows.Forms namespace.

Add a Label control to the form - Click Label in the Toolbox and drag it over the forms Designer and drop it in the desired location.

If you want to change the display text of the Label, you have to set a new text to the Text property of Label.

```
label1.Text = "This is my first Label";
```

In addition to displaying text, the Label control can also display an image using the Image property, or a combination of the ImageIndex and ImageList properties.

```
label1.Image = Image.FromFile("C:\\testimage.jpg");
```

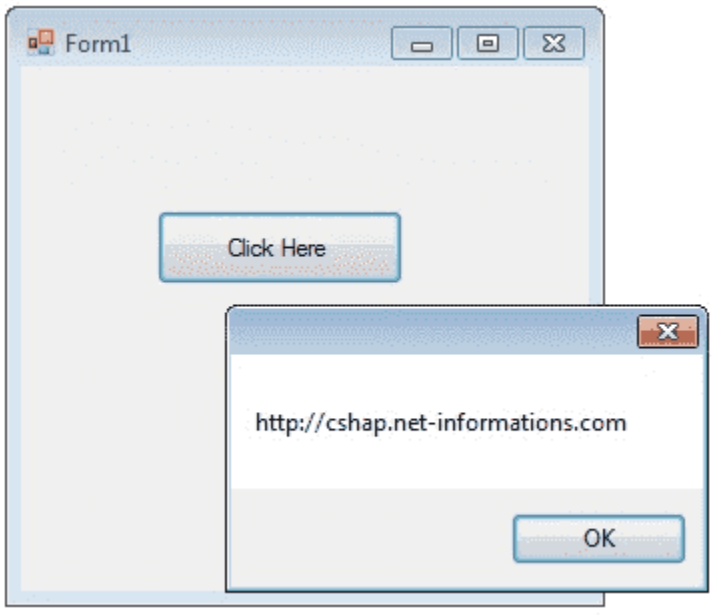
The following C# source code shows how to set some properties of the Label through coding.

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
```

```
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            label1.Text = "This is my first Lable";
            label1.BorderStyle = BorderStyle.FixedSingle;
            label1.TextAlign = ContentAlignment.MiddleCenter;
        }
    }
}
```

C# Button Control

Windows Forms controls are reusable components that encapsulate user interface functionality and are used in client side Windows applications. A button is a control, which is an interactive component that enables users to communicate with an application. The Button class inherits directly from the ButtonBase class. A Button can be clicked by using the mouse, ENTER key, or SPACEBAR if the button has focus.



When you want to change display text of the Button , you can change the Text property of the button.

```
button1.Text = "Click Here";
```

Similarly if you want to load an Image to a Button control , you can code like this.

```
button1.Image = Image.FromFile("C:\\testimage.jpg");
```

How to Call a Button's Click Event Programmatically

The Click event is raised when the Button control is clicked. This event is commonly used when no command name is associated with the Button control. Raising an event invokes the event handler through a delegate.

```
private void Form1_Load(object sender, EventArgs e)

{

   Button b = new Button();

   b.Click += new EventHandler(ShowMessage);

   Controls.Add(b);

}

private void ShowMessage(object sender, EventArgs e)

{

   MessageBox.Show("Button Click");

}
```

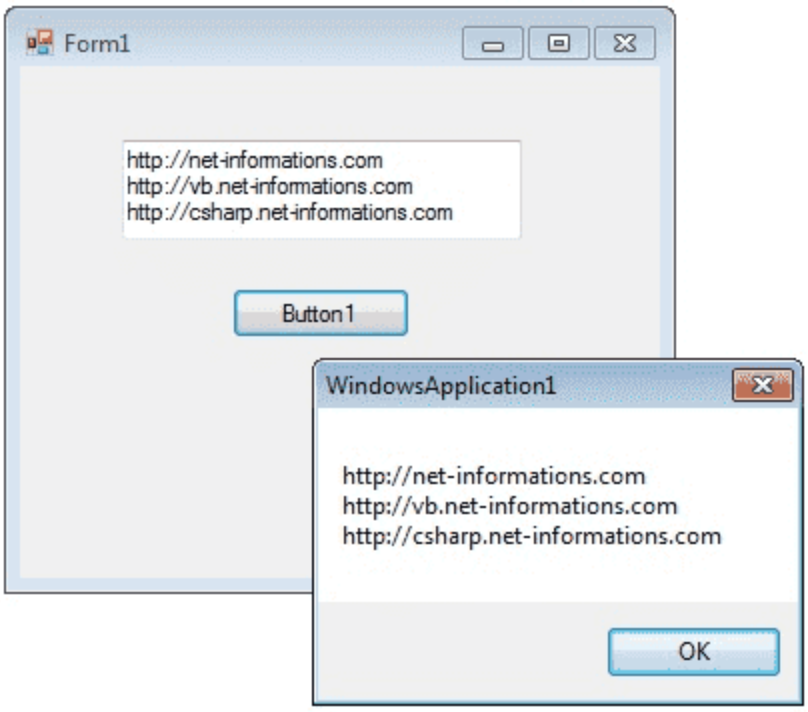
The following C# source code shows how to change the button Text property while Form loading event and to display a message box when pressing a Button Control.

```
using System;
using System.Drawing;
using System.Windows.Forms;
```

```
namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            button1.Text = "Click Here";
        }

        private void button1_Click(object sender, EventArgs e)
        {
            MessageBox.Show("http://cshap.net-informations.com");
        }
    }
}
```

C# TextBox Control

A TextBox control is used to display, or accept as input, a single line of text. This control has additional functionality that is not found in the standard Windows text box control, including multiline editing and password character masking.



A text box object is used to display text on a form or to get user input while a C# program is running. In a text box, a user can type data or paste it into the control from the clipboard.

For displaying a text in a TextBox control , you can code like this.

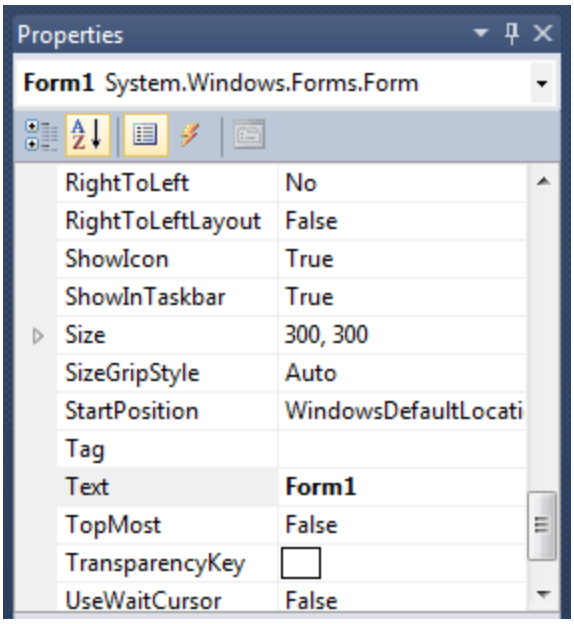
```
textBox1.Text = "http://csharp.net-informations.com";
```

You can also collect the input value from a TextBox control to a variable like this way.

```
string var;

var = textBox1.Text;
```

C# TextBox Properties

You can set TextBox properties through Property window or through program. You can open Properties window by pressing F4 or right click on a control and select Properties menu item.



The below code set a textbox width as 250 and height as 50 through source code.

```
textBox1.Width = 250;

textBox1.Height = 50;
```

Background Color and Foreground Color

You can set background color and foreground color through property window and programmatically.

```
textBox1.BackColor = Color.Blue;

textBox1.ForeColor = Color.White;
```

Textbox BorderStyle

You can set 3 different types of border style for textbox, they are None, FixedSingle and fixed3d.

```
textBox1.BorderStyle = BorderStyle.Fixed3D;
```

TextBox Events

Keydown event

You can capture which key is pressed by the user using KeyDown event

e.g.

```
private void textBox1_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Enter)
    {
        MessageBox.Show("You press Enter Key");

    }
    if (e.KeyCode == Keys.CapsLock)
    {
        MessageBox.Show("You press Caps Lock Key");
    }
}
```

TextChanged Event

When user input or setting the Text property to a new value raises the TextChanged event

e.g.

```
private void textBox1_TextChanged(object sender, EventArgs e)
{
    label1.Text = textBox1.Text;
}
```

Textbox Maximum Length

Sets the maximum number of characters or words the user can input into the text box control.

```
textBox1.MaxLength = 40;
```

Textbox ReadOnly

When a program wants to prevent a user from changing the text that appears in a text box, the program can set the controls Read-only property is to True.

```
textBox1.ReadOnly = true;
```

Multiline TextBox

You can use the Multiline and ScrollBars properties to enable multiple lines of text to be displayed or entered.

```
textBox1.Multiline = true;
```

Textbox password character

TextBox controls 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

```
textBox1.PasswordChar = '*';
```

The above code set the PasswordChar to * , so when the user enter password then it display only * instead of typed characters.

How to Newline in a TextBox

You can add new line in a textbox using many ways.

```
textBox1.Text += "your text" + "\r\n";
```

or

```
textBox1.Text += "your text" + Environment.NewLine;
```

How to retrieve integer values from textbox ?

```
int i;

i = int.Parse (textBox1.Text);
```

Parse method Converts the string representation of a number to its integer equivalent.

String to Float conversion

```
float  i;

i = float.Parse (textBox1.Text);
```

String to Double conversion

```
double   i;

i = float.Parse (textBox1.Text);
```

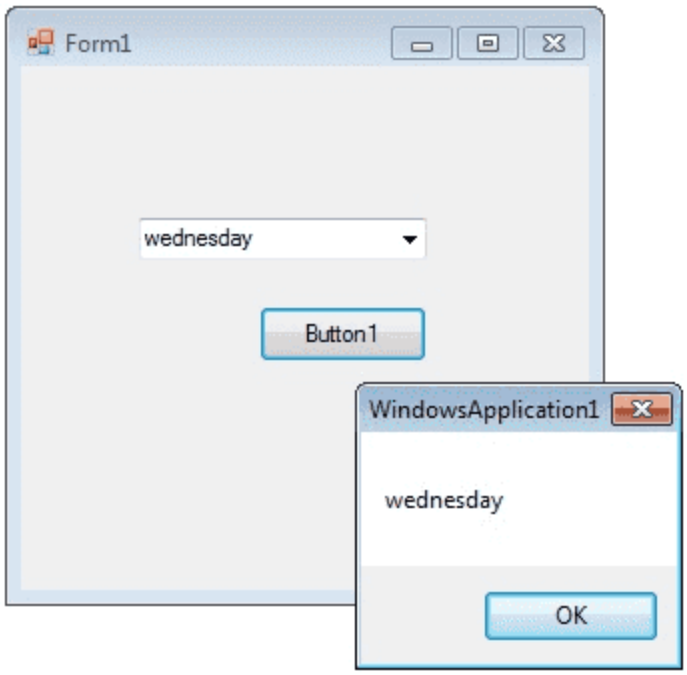
From the following C# source code you can see some important property settings to a TextBox control.

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            textBox1.Width = 250;
            textBox1.Height = 50;
            textBox1.Multiline = true;
            textBox1.BackColor = Color.Blue;
            textBox1.ForeColor = Color.White;
            textBox1.BorderStyle = BorderStyle.Fixed3D;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            string var;
            var = textBox1.Text;
            MessageBox.Show(var);
        }
```

```
    }
}
```

C# ComboBox Control

C# controls are located in the Toolbox of the development environment, and you use them to create objects on a form with a simple series of mouse clicks and dragging motions. A ComboBox displays a text box combined with a ListBox, which enables the user to select items from the list or enter a new value.



The user can type a value in the text field or click the button to display a drop down list. You can add individual objects with the Add method. You can delete items with the Remove method or clear the entire list with the Clear method.

How add a item to combobox

```
comboBox1.Items.Add("Sunday");
comboBox1.Items.Add("Monday");
comboBox1.Items.Add("Tuesday");
```

ComboBox SelectedItem

How to retrieve value from ComboBox

If you want to retrieve the displayed item to a string variable , you can code like this

```
string var;
var = comboBox1.Text;
                    Or
var item = this.comboBox1.GetItemText(this.comboBox1.SelectedItem);
MessageBox.Show(item);
```

How to remove an item from ComboBox

You can remove items from a combobox in two ways. You can remove item at a the specified index or giving a specified item by name.

```
comboBox1.Items.RemoveAt(1);
```

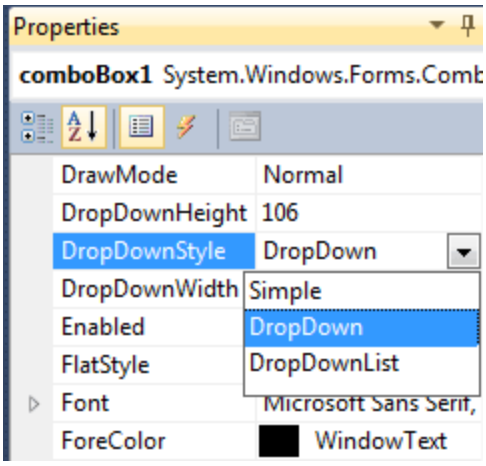
The above code will remove the second item from the combobox.

```
comboBox1.Items.Remove("Friday");
```

The above code will remove the item "Friday" from the combobox.

DropDownStyle

The DropDownStyle property specifies whether the list is always displayed or whether the list is displayed in a drop-down. The DropDownStyle property also specifies whether the text portion can be edited.




```
comboBox1.DropDownStyle = ComboBoxStyle.DropDown;
```

comboBox Selected Value

How to set the selected item in a comboBox

You can display selected item in a combobox in two ways.

```
comboBox1.Items.Add("test1");
comboBox1.Items.Add("test2");
comboBox1.Items.Add("test3");
comboBox1.SelectedItem = "test3";
or
comboBox1.SelectedIndex = comboBox1.FindStringExact("test3");
```

ComboBox DataSource Property

How to populate a combo box with a DataSet ?

You can Programmatically Binding DataSource to ComboBox in a simple way..

Consider an sql string like...."select au_id,au_lname from authors";

Make a datasource and bind it like the following...

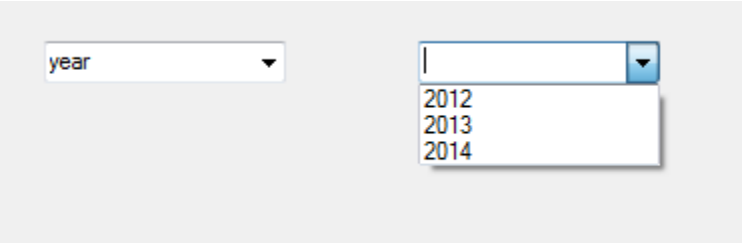
```
comboBox1.DataSource = ds.Tables[0];
comboBox1.ValueMember = "au_id";
comboBox1.DisplayMember = "au_lname";
```

Combobox SelectedIndexChanged event

The SelectedIndexChanged event of a combobox fire when you change the slected item in a combobox. If you want to do something when you change the selection, you can write the program on SelectedIndexChanged event. From the following code you can understand how to set values in the SelectedIndexChanged event of a combobox. Drag and drop two combobox on the Form and copy and paste the following source code.

```
using System;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_Load(object sender, EventArgs e)
        {
            comboBox1.Items.Add("weekdays");
            comboBox1.Items.Add("year");
        }
        private void comboBox1_SelectedIndexChanged(object sender,
EventArgs e)
        {
            comboBox2.Items.Clear();
            if (comboBox1.SelectedItem == "weekdays")
            {
                comboBox2.Items.Add("Sunday");
                comboBox2.Items.Add("Monday");
                comboBox2.Items.Add("Tuesday");
            }
            else if (comboBox1.SelectedItem == "year")
            {
                comboBox2.Items.Add("2012");
                comboBox2.Items.Add("2013");
                comboBox2.Items.Add("2014");
            }
        }
    }
}
```

Output



ComboBox Default Value

How to set a default value for a Combo Box

You can set combobox default value by using SelectedIndex property

```
comboBox1.SelectedIndex = 6;
```

ComboBox readonly

How to make a combobox read only

You can make a ComboBox readonly, that means a user cannot write in a combo box but he can select the given items, in two ways. By default, DropDownStyle property of a Combobox is DropDown. In this case user can enter values to combobox. When you change the DropDownStyle property to DropDownList, the Combobox will become read only and user can not enter values to combobox. Second method, if you want the combobox completely read only, you can set comboBox1.Enabled = false.

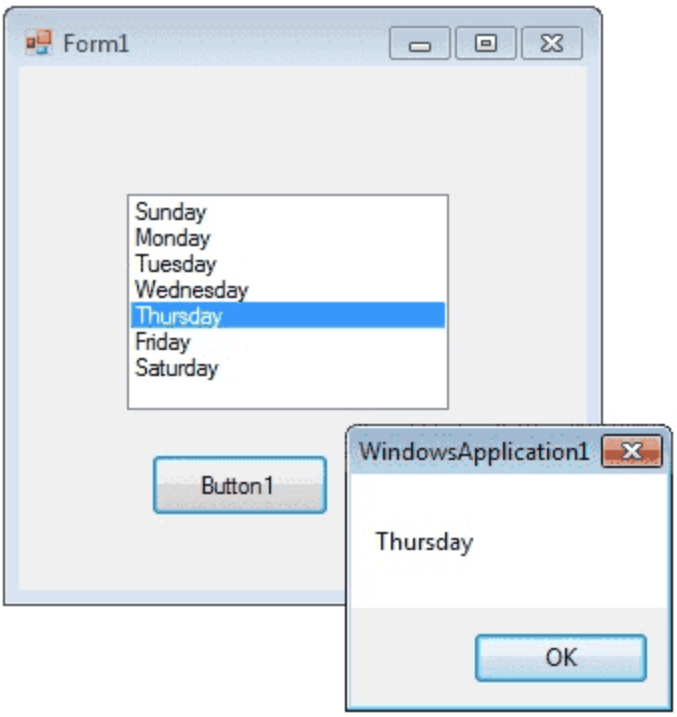
ComboBox Example

The following C# source code add seven days in a week to a combo box while load event of a Windows Form and int Button click event it displays the selected text in the Combo Box.

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            comboBox1.Items.Add("Sunday");
            comboBox1.Items.Add("Monday");
            comboBox1.Items.Add("Tuesday");
            comboBox1.Items.Add("wednesday");
            comboBox1.Items.Add("Thursday");
            comboBox1.Items.Add("Friday");
            comboBox1.Items.Add("Saturday");
                      comboBox1.SelectedIndex =
comboBox1.FindStringExact("Sunday");

        }

        private void button1_Click(object sender, EventArgs e)
        {
            string var;
            var = comboBox1.Text;
            MessageBox.Show(var);
        }
    }
}
```

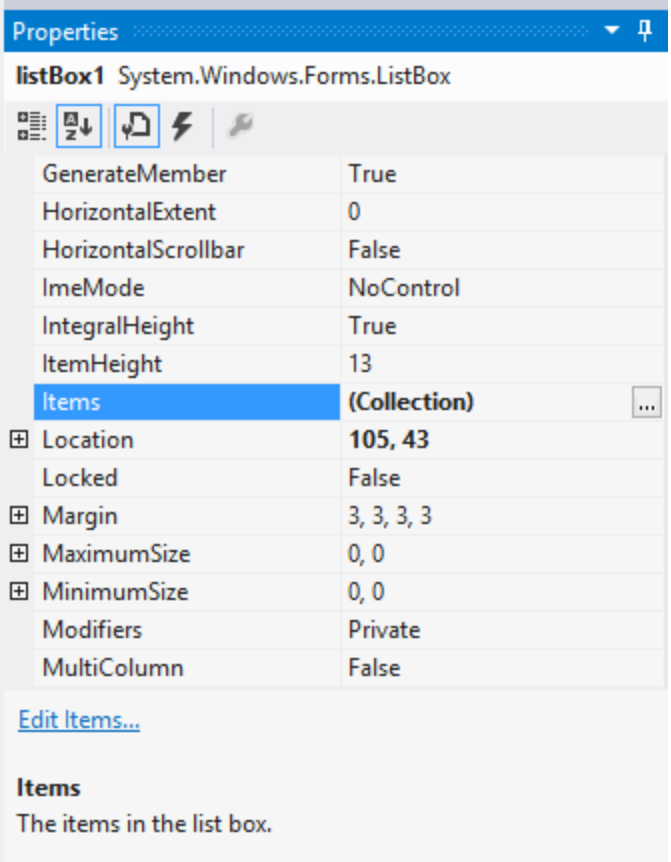
ListBox control

The **ListBox control** enables you to display a list of items to the user that the user can select by clicking.



Setting ListBox Properties

You can set **ListBox properties** by using Properties Window. In order to get Properties window you can Press F4 or by right-clicking on a control to get the "Properties" menu item.



You can set property values in the right side column of the property window.

C# Listbox example

Add Items in a Listbox

Syntax

```
public int Add (object item);
```

In addition to display and selection functionality, the **ListBox** also provides features that enable you to efficiently add items to the ListBox and to find text within the items of the list. You can use the Add or Insert method to add items to a list box. The **Add method** adds new items at the end of an unsorted list box.

```
listBox1.Items.Add("Sunday");
```

If the **Sorted property** of the C# ListBox is set to true, the item is inserted into the list alphabetically. Otherwise, the item is inserted at the end of the ListBox.

Insert Items in a Listbox

Syntax

```
public void Insert (int index, object item);
```

You can **inserts** an item into the list box at the specified index.

```
listBox1.Items.Insert(0, "First");

listBox1.Items.Insert(1, "Second");

listBox1.Items.Insert(2, "Third");

listBox1.Items.Insert(3, "Forth");
```

Listbox Selected Item

If you want to retrieve a **single selected** item to a variable , use the following code.

```
string item = listBox1.GetItemText(listBox1.SelectedItem);
```

Or you can use

```
string item = listBox1.SelectedItem.ToString();
```

Or

```
string item= listBox1.Text;
```

Selecting Multiple Items from Listbox

The **SelectionMode property** determines how many items in the list can be selected at a time. A ListBox control can provide single or **multiple selections** using the SelectionMode property . If you change the selection mode property to multiple select , then you will retrieve a collection of items from ListBox1.SelectedItems property.

```
listBox1.SelectionMode = SelectionMode.MultiSimple;
```

The ListBox class has two SelectionMode. **Multiple** or **Extended** .

In **Multiple mode** , you can select or deselect any item in a ListBox by clicking it. In Extended mode, you need to hold down the Ctrl key to select additional items or the Shift key to select a range of items.

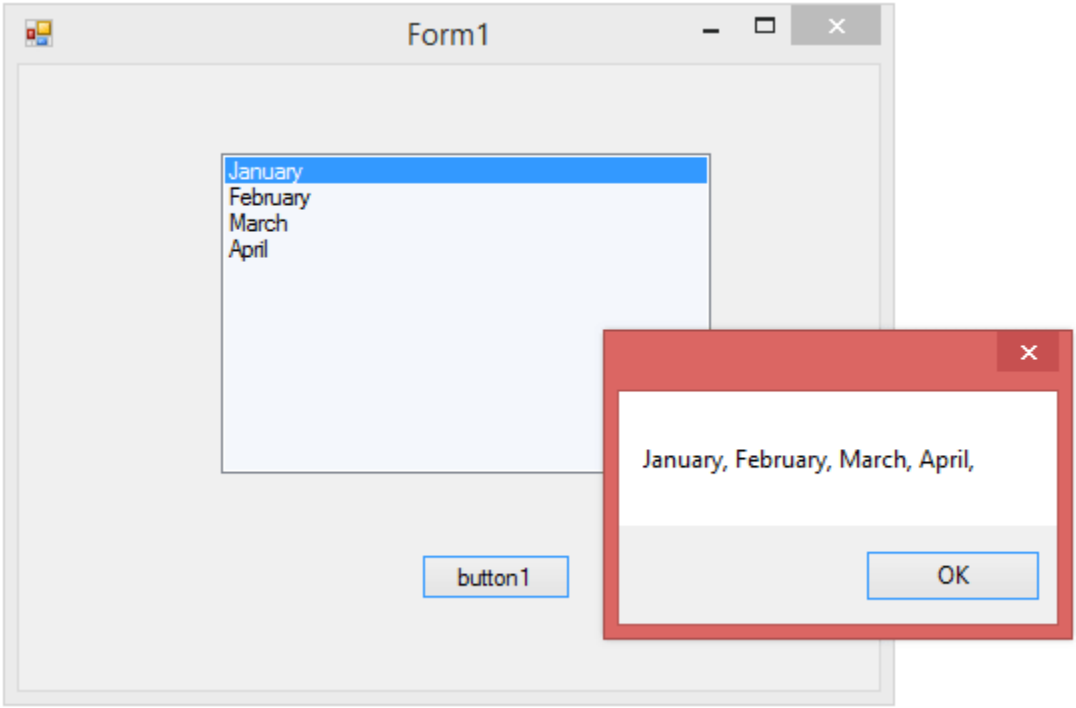
The following C# program initially fill seven days in a week while in the form load event and set the selection mode property to **MultiSimple** . At the Button click event it will display the selected items.

```
using System;

using System.Drawing;

using System.Windows.Forms;

namespace WindowsFormsApplication1

{

 public partial class Form1 : Form

 {

  public Form1()

  {

   InitializeComponent();

  }

  private void Form1_Load(object sender, EventArgs e)
```

```
  {
    listBox1.Items.Add("Sunday");
    listBox1.Items.Add("Monday");
    listBox1.Items.Add("Tuesday");
    listBox1.Items.Add("Wednesday");
    listBox1.Items.Add("Thursday");
    listBox1.Items.Add("Friday");
    listBox1.Items.Add("Saturday");
    listBox1.SelectionMode = SelectionMode.MultiSimple;
  }
  private void button1_Click(object sender, EventArgs e)
  {
    foreach (Object obj in listBox1.SelectedItems )
    {
      MessageBox.Show(obj.ToString ());
    }
  }
 }
}
```

Getting All Items from List box

The Items collection of C# Winforms Listbox returns a **Collection type** of Object so you can use ToString() on each item to display its text value as below:




```
private void button1_Click_1(object sender, EventArgs e)
{
 string items = "";
 foreach (var item in listBox1.Items)
 {
  items += item.ToString() + ", ";
 }
 MessageBox.Show(items);
}
```

How to bind a ListBox to a List ?

You can bind a List to a ListBox control by create a **fresh List Object** and add items to the List.

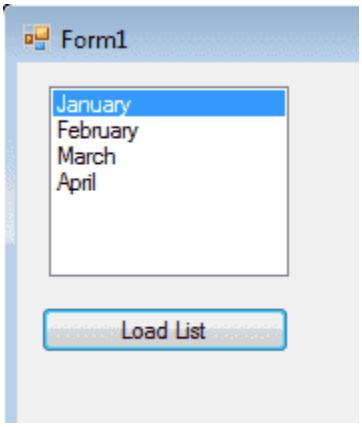
Creating a List

```
List<string> nList = new List<string>();

nList.Add("January");

nList.Add("February");

nList.Add("March");

nList.Add("April");
```

Binding to List

The next step is to bind this List to the Listbox. In order to do that you should **set datasource** of the Listbox.

```
listBox1.DataSource = nList;
```




```
private void button1_Click(object sender, EventArgs e)

{

  List<string> nList = new List<string>();

  nList.Add("January");

  nList.Add("February");

  nList.Add("March");

  nList.Add("April");
```

```
 listBox1.DataSource = nList;

}
```

How to bind a listbox to database values ?

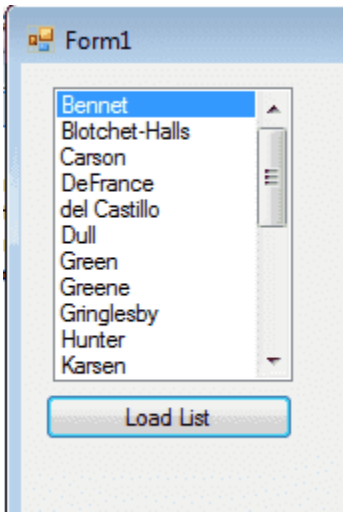
First of all, you could create a **connection string** and fetch data from database to a Dataset.

```
connetionString = "Data Source=ServerName;Initial Catalog=databasename;User
ID=userid;Password=yourpassword";

sql = "select au_id,au_lname from authors";
```

Set Datasource for ListBox

Next step is that you have set Listbox **datasoure** as Dataset.

```
listBox1.DataSource = ds.Tables[0];

listBox1.ValueMember = "au_id";

listBox1.DisplayMember = "au_lname";
```




```
using System;
```

```
using System.Data;
using System.Data.SqlClient;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{
  public partial class Form1 : Form
  {
    public Form1()
    {
      InitializeComponent();
    }
    private void button1_Click(object sender, EventArgs e)
    {
      string connetionString = null;
      SqlConnection connection;
      SqlCommand command;
      SqlDataAdapter adapter = new SqlDataAdapter();
      DataSet ds = new DataSet();
      int i = 0;
      string sql = null;
      //connetionString = "Data Source=ServerName;Initial Catalog=databasename;User
ID=userid;Password=yourpassword";
      //sql = "select au_id,au_lname from authors";
      connection = new SqlConnection(connetionString);
      try
      {
```

```
        connection.Open();
        command = new SqlCommand(sql, connection);
        adapter.SelectCommand = command;
        adapter.Fill(ds);
        adapter.Dispose();
        command.Dispose();
        connection.Close();
        listBox1.DataSource = ds.Tables[0];
        listBox1.ValueMember = "au_id";
        listBox1.DisplayMember = "au_lname";
      }
      catch (Exception ex)
      {
        MessageBox.Show("Cannot open connection ! ");
      }
    }
  }
}
```

How to refresh DataSource of a ListBox ?

How to clear the Listbox if its already binded with datasource ?

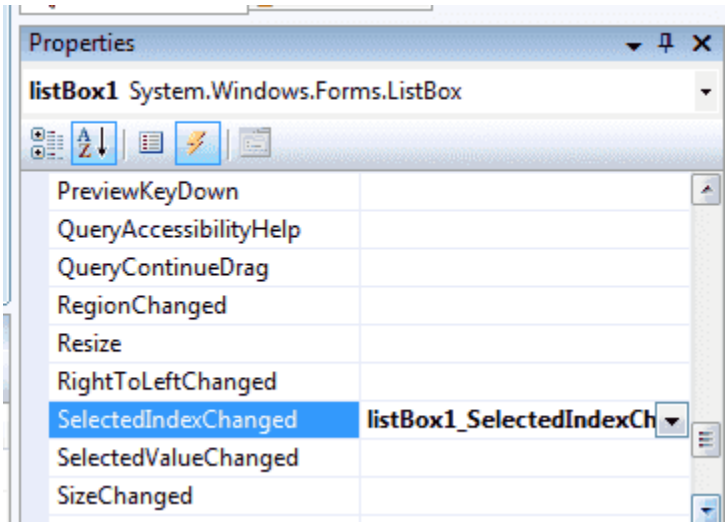
When you want to clear the Listbox, if the ListBox already **binded with Datasource** , you have to set the Datasource of Listbox as null.

```
listBox1.DataSource = null;
```

How to SelectedIndexChanged event in ListBox ?

This event is fired when the item selection is changed in a **ListBox** . You can use this event in a situation that you want select an item from your listbox and accodring to this selection you can perform other programming needs.

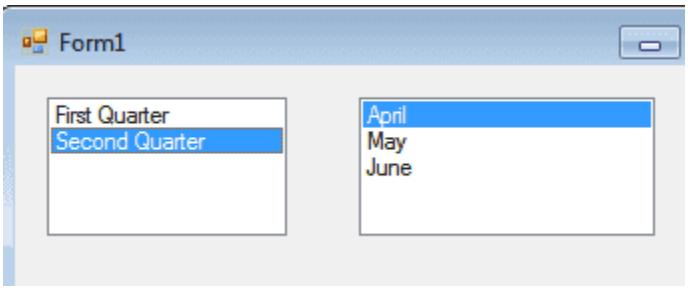
You can add the event handler using the **Properties Window** and selecting the Event icon and double-clicking on SelectedIndexChanged as you can see in following image.



The event will fire again when you select a new item. You can write your code within **SelectedIndexChanged** event . When you double click on ListBox the code will automatically come in you code editor like the following image.

```
private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    //your code here
}
```

From the following example you can understand how to fire the SelectedIndexChanged event.



First you should drag two listboxes on your Form. First listbox you should set the List as **Datasource** , the List contents follows:

```
List<string> nList = new List<string>();

nList.Add("First Quarter");

nList.Add("Second Quarter");
```

When you load this form you can see the listbox is **populated** with List and displayed first quarter and second quarter. When you click the "Fist Quarter" the next listbox is populated with first quarter months and when you click "Second Quarter" you can see the second listbox is changed to second quarter months. From the following program you can understand how this happened.

```
using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

using System.Collections.Generic;

namespace WindowsFormsApplication1

{

  public partial class Form1 : Form
```

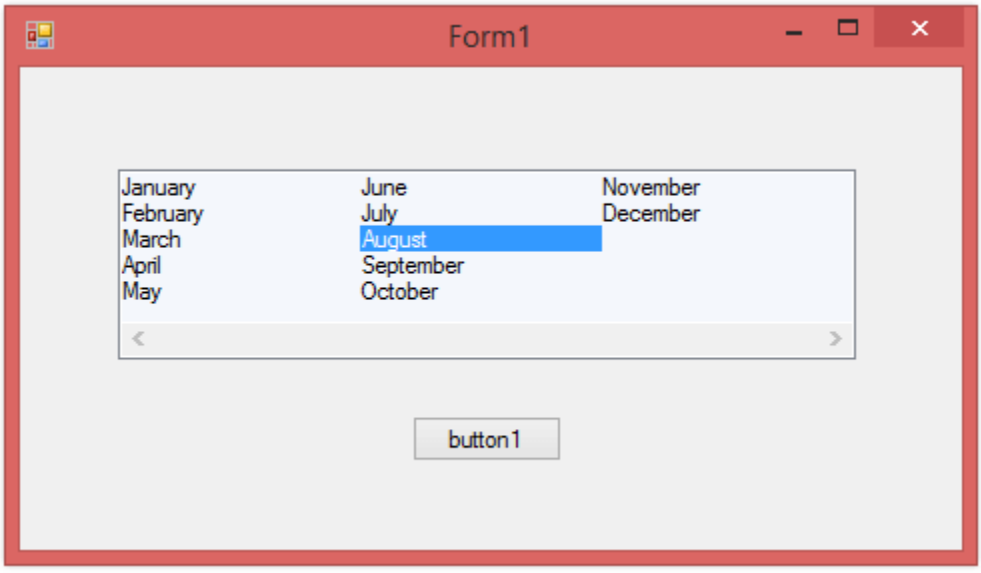
```
{
  public Form1()
  {
     InitializeComponent();
  }
  List  < string > fQ = new List  < string > ();
  List  < string > sQ = new List  < string > ();
  private void Form1_Load(object sender, EventArgs e)
  {
    fQ.Add("January");
    fQ.Add("February");
    fQ.Add("March");
    sQ.Add("April");
    sQ.Add("May");
    sQ.Add("June");
    List < string > nList = new List  < string > ();
    nList.Add("First Quarter");
    nList.Add("Second Quarter");
    listBox1.DataSource = nList;
  }
  private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
  {
    if (listBox1.SelectedIndex == 0)
    {
       listBox2.DataSource = null;
```

```
        listBox2.DataSource = fQ;
    }
    else if (listBox1.SelectedIndex == 1)
    {
        listBox2.DataSource = null;
        listBox2.DataSource = sQ;
    }
  }
 }
}
```

C# Listbox Column

A **multicolumn ListBox** places items into as many columns as are needed to make vertical scrolling unnecessary. The user can use the keyboard to navigate to columns that are not currently visible. First of all, you have Gets or sets a value indicating whether the ListBox supports **multiple columns** .

```
public bool MultiColumn { get; set; }
```

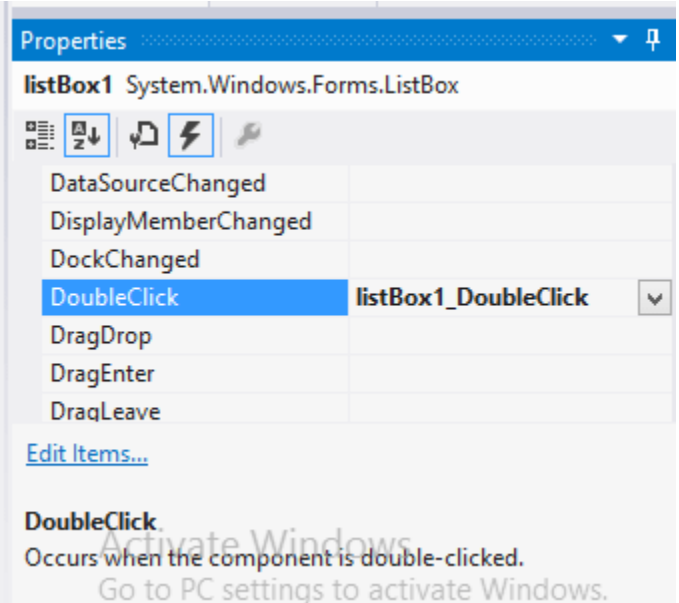


The following C# code example demonstrates a simple muliple column ListBox.

```
private void button1_Click_1(object sender, EventArgs e)
{
 listBox1.HorizontalScrollbar = true;
 listBox1.MultiColumn = true;
 listBox1.ScrollAlwaysVisible = true;
 listBox1.Items.AddRange(new object[] {
 "January",
 "February",
 "March",
 "April",
 "May",
 "June",
 "July",
 "August",
 "September",
 "October",
 "November",
 "December"});
}
```

Listbox Item Double Click Event

Add an event handler for the **Control.DoubleClick** event for your ListBox.



The following program shows when you double click the Listbox Item that event handler open up a MessageBox displaying the selected item.

```
private void listBox1_DoubleClick(object sender, EventArgs e)

{

 if (listBox1.SelectedItem != null)

 {

  MessageBox.Show(listBox1.SelectedItem.ToString());

 }

}
```

Listbox vertical scrollbar

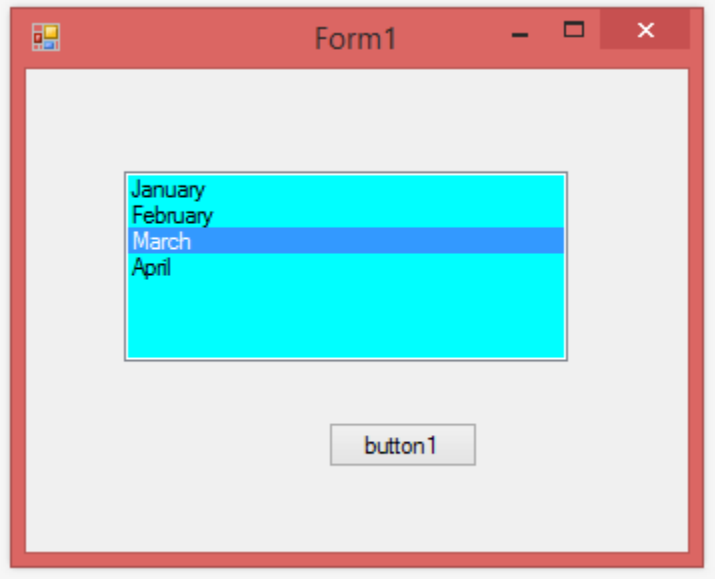
ListBox only shows a **vertical scroll bar** when there are more items then the displaying area. You can fix with ListBox.ScrollAlwaysVisible Property if you want the bar to be visible all the time. If **ListBox.ScrollAlwaysVisible** Property is true if the vertical scroll bar should always be displayed; otherwise, false. The default is false.

```
// Turn off the scrollbar.

ListBox1.ScrollAlwaysVisible = false;
```

Background and Foreground

The Listbox **BackColor** and **ForeColor properties** are used to set the background and foreground colors respectively. If you click on these properties in the Properties window, then the Color Dialog pops up and you can select the color you want.

Alternatively, you can set **background** and foreground colors from source code. The following code sets the BackColor and ForeColor properties:




```
listBox1.BackColor = System.Drawing.Color.Aqua;

listBox1.ForeColor = System.Drawing.Color.Black;
```

How to clear all data in a listBox?

Listbox **Items.Clear()** method clear all the items from ListBox.

```
private void button1_Click_1(object sender, EventArgs e)
```

```
{
   listBox1.Items.Clear();
}
```

ListBox.ClearSelected Method Unselects all items in the ListBox.

```
// Clear all selections in the ListBox.
listBox1.ClearSelected();
```

Remove item from Listbox

```
public void RemoveAt (int index);
```

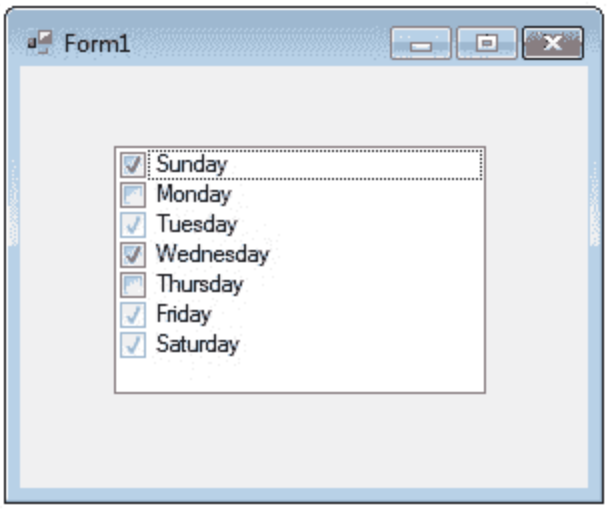
RemoveAt method removes the item at the **specified index** within the collection. When you remove an item from the list, the indexes change for subsequent items in the list. All information about the **removed item** is deleted.

Listbox Vs ListView Vs GridView

C# ListBox has many similarities with **ListView** or **GridView** (they share the parent class ItemsControl), but each control is oriented towards different situations. ListBox is best for **general UI** composition, particularly when the elements are always intended to be selectable, whereas ListView or GridView are best for **data binding** scenarios, particularly if virtualization or large data sets are involved. One most important difference is listview uses the extended selection mode by default .

C# Checked ListBox Control

The **CheckedListBox control** gives you all the capability of a list box and also allows you to display a **check mark** next to the items in the list box.



The user can place a check mark by one or more items and the checked items can be navigated with the **CheckedListBox.CheckedItemCollection** and **CheckedListBox.CheckedIndexCollection** .

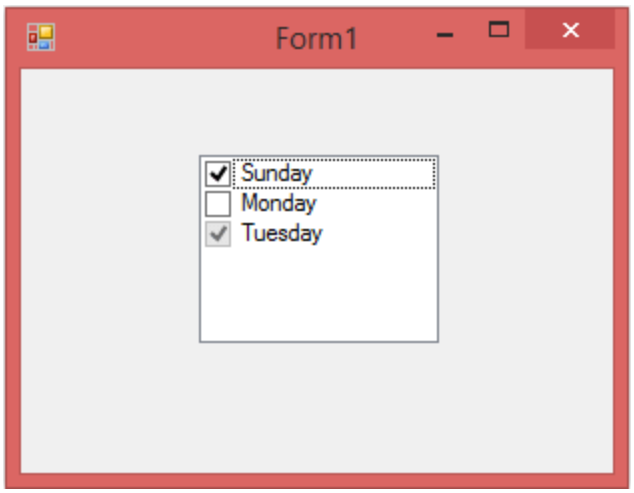
Checkedlistbox add items

Syntax

```
public int Add (object item, bool isChecked);
```

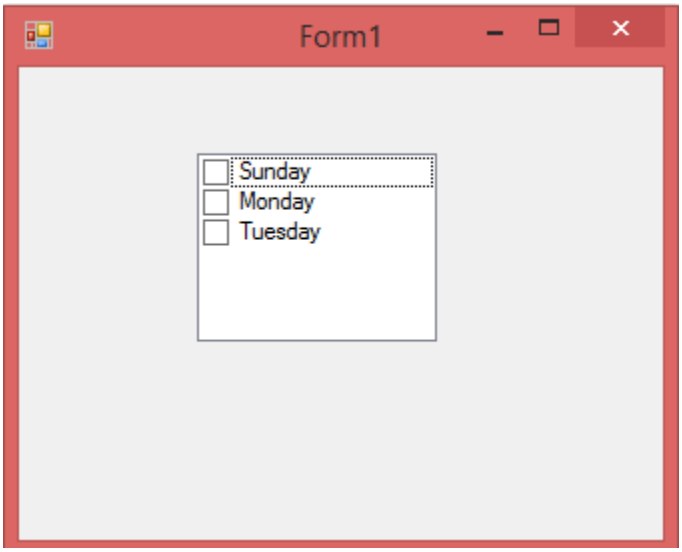
You can add individual items to the list with the **Add method** . The CheckedListBox object supports three states through the CheckState enumeration: Checked, Indeterminate, and Unchecked.

```
checkedListBox1.Items.Add("Sunday", CheckState.Checked);

checkedListBox1.Items.Add("Monday", CheckState.Unchecked);

checkedListBox1.Items.Add("Tuesday", CheckState.Indeterminate);
```



If you want to add objects to the list at run time, assign an array of object references with the **AddRange method** . The list then displays the default string value for each object.

```
string[] days = new[] { "Sunday", "Monday", "Tuesday" };

 checkedListBox1.Items.AddRange(days);
```



By default checkedlistbox items are **unchecked** .

Check all items in a Checkedlistbox

If you want to check an item in a Checkedlistbox, you need to call **SetItemChecked** with the relevant item.

```
public void SetItemChecked (int index, bool value);
```

Parameters

- index(Int32) - The index of the item to set the check state for.
- value(Boolean) - true to set the item as checked; otherwise, false.

If you want to set all items in a CheckedListBox to checked, change the value of **SetItemChecked** method to true.

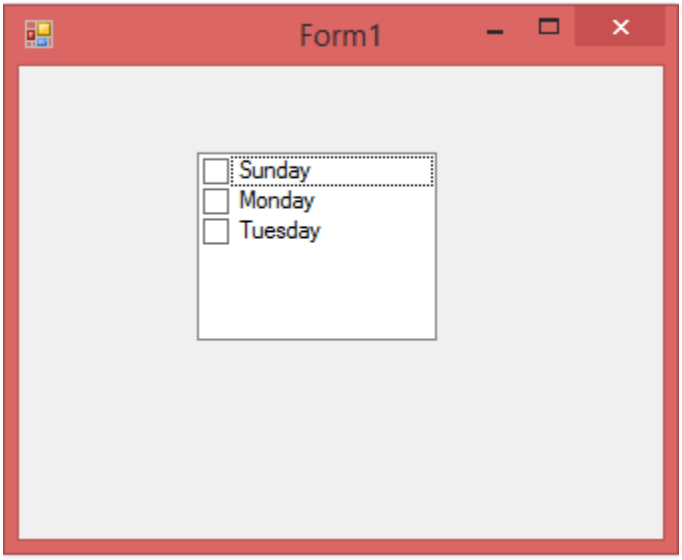
```
string[] days = new[] { "Sunday", "Monday", "Tuesday" };

checkedListBox1.Items.AddRange(days);

for (int i = 0; i < checkedListBox1.Items.Count; i++)

{

   checkedListBox1.SetItemChecked(i, true);

}
```

Uncheck all items in a Checkedlistbox

If you want to set all items in a CheckedListBox to unchecked, change the value of **SetItemChecked** method to false.

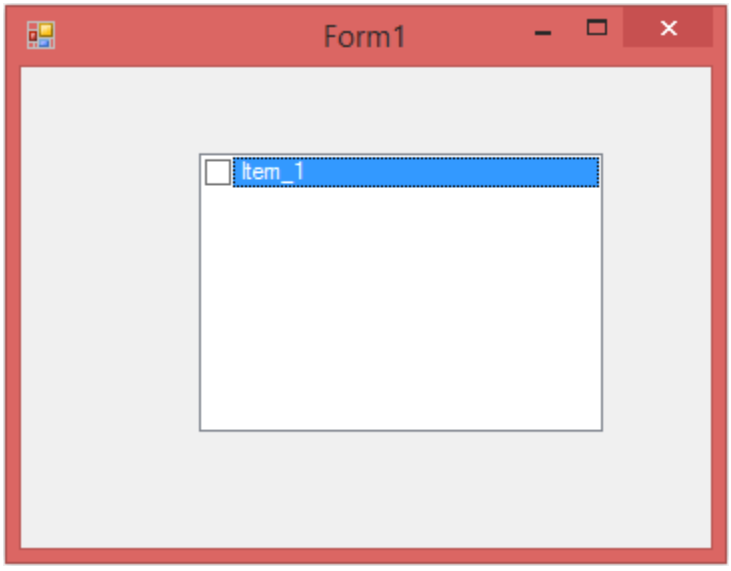
```
checkedListBox1.Items.Add("Sunday", CheckState.Checked);

checkedListBox1.Items.Add("Monday", CheckState.Checked);

checkedListBox1.Items.Add("Tuesday", CheckState.Checked);

for (int i = 0; i < checkedListBox1.Items.Count; i++)

{

   checkedListBox1.SetItemChecked(i, false);
```

```
}
```



CheckedListBox DataSource

Following example shows how to bind a **DataSource** to CheckedListBox

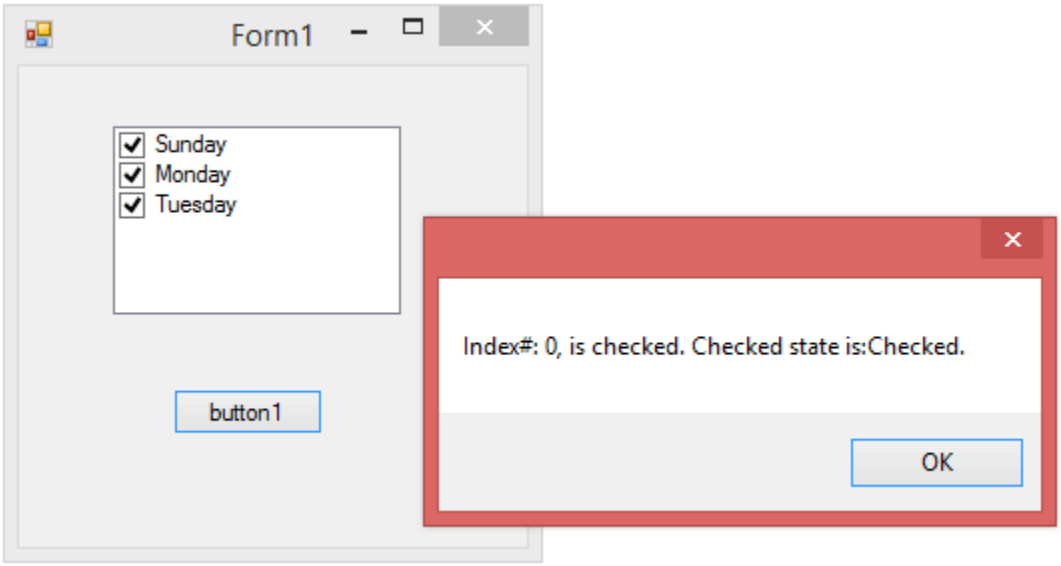



```
DataTable dt = new DataTable();

DataColumn dc = new DataColumn("StringType", typeof(String));

dt.Columns.Add(dc);
```

```
DataRow dr = dt.NewRow();

dr[0] = "Item_1";

dt.Rows.Add(dr);

this.checkedListBox1.DataSource = dt;

this.checkedListBox1.DisplayMember = "StringType";
```

How to get value of checked item from CheckedListBox?

```
public System.Windows.Forms.CheckedListBox.ObjectCollection Items { get; }
```



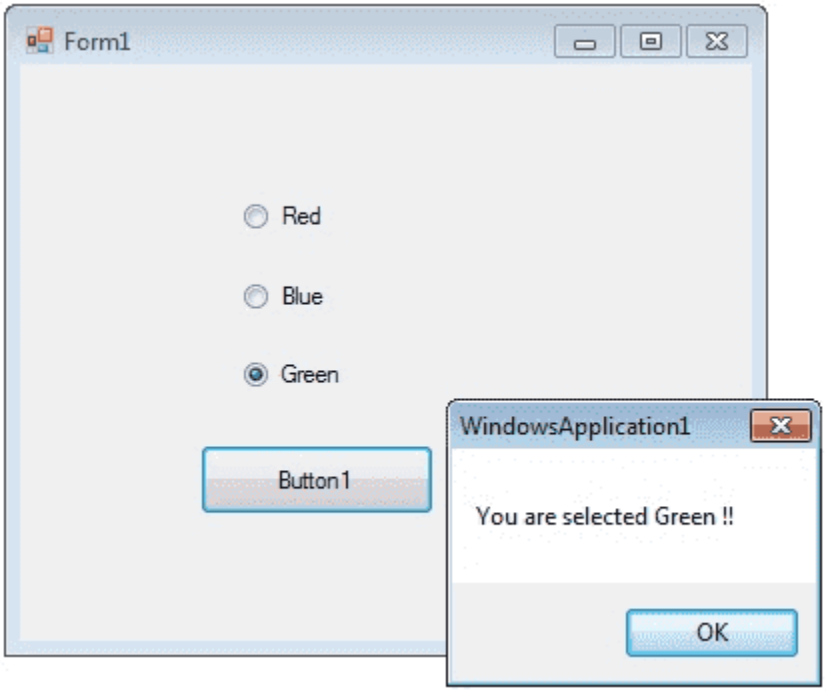
The following example uses the Items property to get the **CheckedListBox.ObjectCollection** to retrieve the index of an item using the ListBox.ObjectCollection.IndexOf method.

```
private void button1_Click_1(object sender, EventArgs e)

{

   foreach (int indexChecked in checkedListBox1.CheckedIndices)

   {
```

```
        // The indexChecked variable contains the index of the item.

        MessageBox.Show("Index: " + indexChecked.ToString() + ", is checked. Checked state is:"
+  checkedListBox1.GetItemCheckState(indexChecked).ToString() + ".");

    }

}
```

C# RadioButton Control

A radio button or option button enables the user to select a single option from a group of choices when paired with other RadioButton controls. When a user clicks on a radio button, it becomes checked, and all other radio buttons with same group become unchecked.



The RadioButton control can display text, an Image, or both. Use the Checked property to get or set the state of a RadioButton.

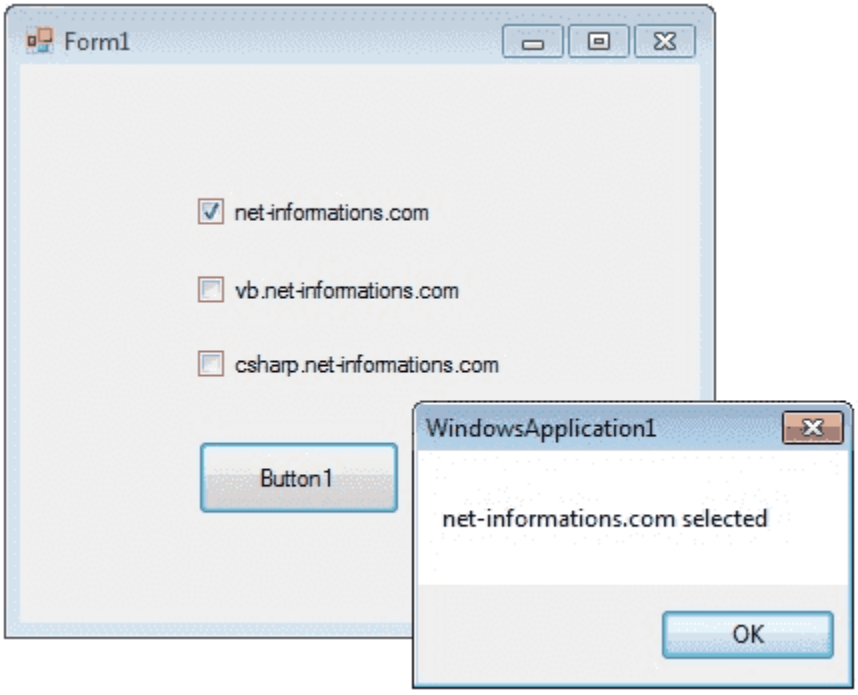
radioButton1.Checked = true;

The radio button and the check box are used for different functions. Use a radio button when you want the user to choose only one option. When you want the user to choose all appropriate options, use a check box. Like check boxes, radio buttons support a Checked property that indicates whether the radio button is selected.

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            radioButton1.Checked = true;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (radioButton1.Checked == true)
            {
                MessageBox.Show ("You are selected Red !! ");
                return;
            }
            else if (radioButton2.Checked == true)
            {
                MessageBox.Show("You are selected Blue !! ");
                return;
            }
            else
            {
                MessageBox.Show("You are selected Green !! ");
                return;
            }
        }
    }
}
```

C# CheckBox Control

CheckBoxes allow the user to make multiple selections from a number of options. CheckBox to give the user an option, such as true/false or yes/no. You can click a check box to select it and click it again to deselect it.



The CheckBox control can display an image or text or both. Usually CheckBox comes with a caption, which you can set in the Text property.

```
checkBox1.Text = "Net-informations.com";
```

You can use the CheckBox control ThreeState property to direct the control to return the Checked, Unchecked, and Indeterminate values. You need to set the check boxs ThreeState property to True to indicate that you want it to support three states.

```
checkBox1.ThreeState = true;
```

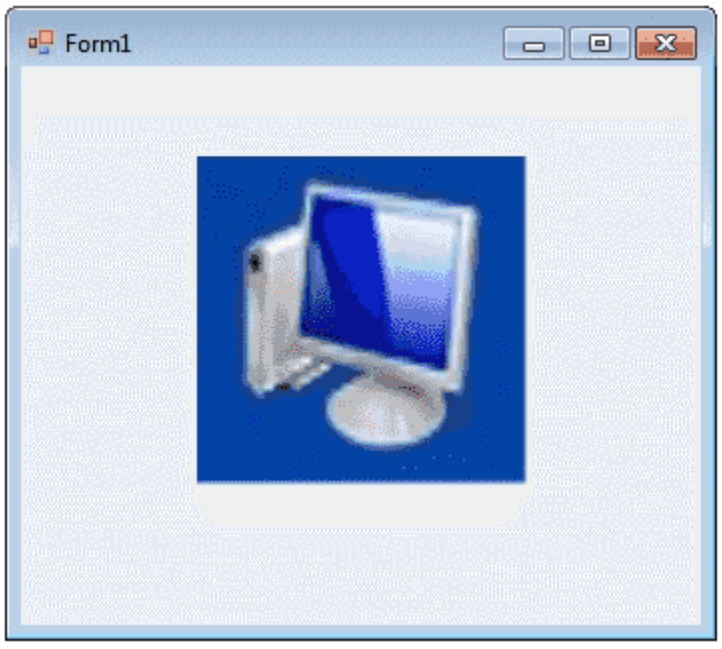
The radio button and the check box are used for different functions. Use a radio button when you want the user to choose only one option.When you want the user to choose all appropriate options, use a check box. The following C# program shows how to find a checkbox is selected or not.

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            string msg = "";

            if (checkBox1.Checked == true)
            {
                msg = "net-informations.com";
            }

            if (checkBox2.Checked == true)
            {
                msg = msg + "   vb.net-informations.com";
            }

            if (checkBox3.Checked == true)
            {
                msg = msg + "   csharp.net-informations.com";
            }


            if (msg.Length > 0)
            {
                MessageBox.Show (msg + " selected ");
            }
            else
            {
                MessageBox.Show ("No checkbox selected");
            }
```

```
            checkBox1.ThreeState = true;
        }
    }
}
```

C# PictureBox Control

The Windows Forms PictureBox control is used to display images in bitmap, GIF , icon , or JPEG formats.



You can set the Image property to the Image you want to display, either at design time or at run time. You can programmatically change the image displayed in a picture box, which is particularly useful when you use a single form to display different pieces of information.

```
pictureBox1.Image = Image.FromFile("c:\\testImage.jpg");
```

The SizeMode property, which is set to values in the PictureBoxSizeMode enumeration, controls the clipping and positioning of the image in the display area.

```
pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;
```

There are five different PictureBoxSizeMode is available to PictureBox control.

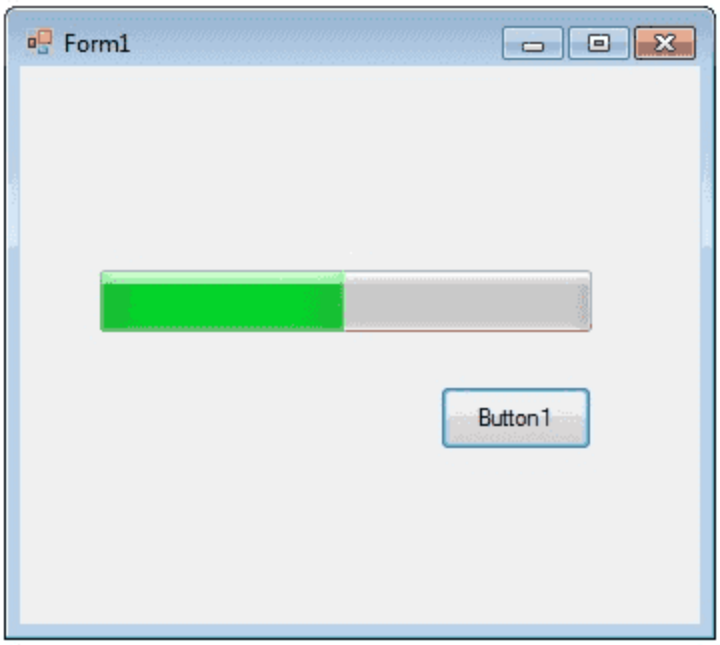
- AutoSize - Sizes the picture box to the image.
- CenterImage - Centers the image in the picture box.
- Normal - Places the upper-left corner of the image at upper left in the picture box
- StretchImage - Allows you to stretch the image in code

The PictureBox is not a selectable control, which means that it cannot receive input focus. The following C# program shows how to load a picture from a file and display it in streach mode.

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            pictureBox1.Image = Image.FromFile("c:\\testImage.jpg");
            pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;
        }

    }
}
```

C# ProgressBar Control

A progress bar is a control that an application can use to indicate the progress of a lengthy operation such as calculating a complex result, downloading a large file from the Web etc.



ProgressBar controls are used whenever an operation takes more than a short period of time. The Maximum and Minimum properties define the range of values to represent the progress of a task.

- Minimum : Sets the lower value for the range of valid values for progress.
- Maximum : Sets the upper value for the range of valid values for progress.
- Value : This property obtains or sets the current level of progress.

By default, Minimum and Maximum are set to 0 and 100. As the task proceeds, the ProgressBar fills in from the left to the right. To delay the program briefly so that you can view changes in the progress bar clearly.

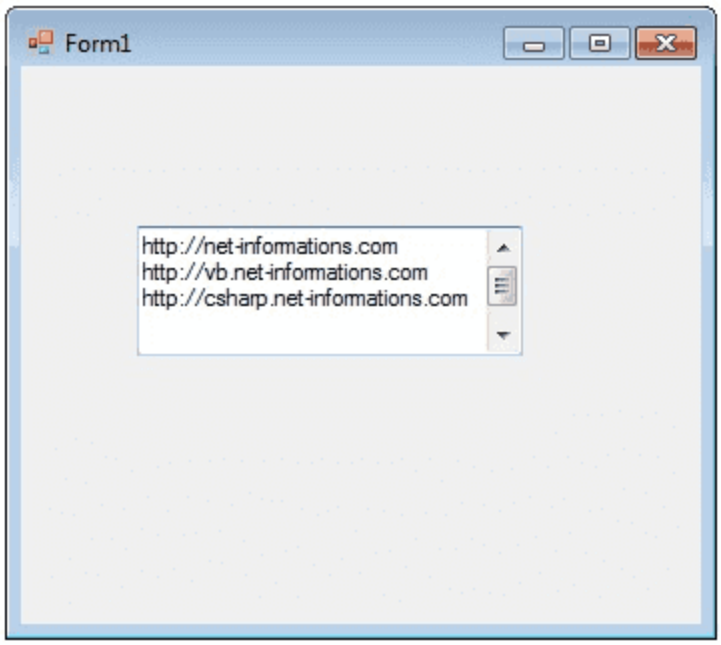
The following C# program shows a simple operation in a progressbar .

```
using System;
using System.Drawing;
using System.Windows.Forms;
```

```
namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            int i;

            progressBar1.Minimum = 0;
            progressBar1.Maximum = 200;

            for (i = 0; i <= 200; i++)
            {
                progressBar1.Value = i;
            }

        }

    }
}
```

C# ScrollBars Control

A ScrollBar allows you to view content that is outside of the current viewing area by sliding the Thumb to make the content visible.



The ScrollBar control contains a Track control. The Track control consists of a Thumb control and two RepeatButton controls. You can increase and decrease the Value property of the ScrollBar control by pressing the RepeatButton controls or by moving the Thumb. You can set the Value property yourself in code, which moves the scroll box to match. The Minimum and Maximum properties determine the range of values that the control can display. The default range of values for the Value property is from 0 to 1.

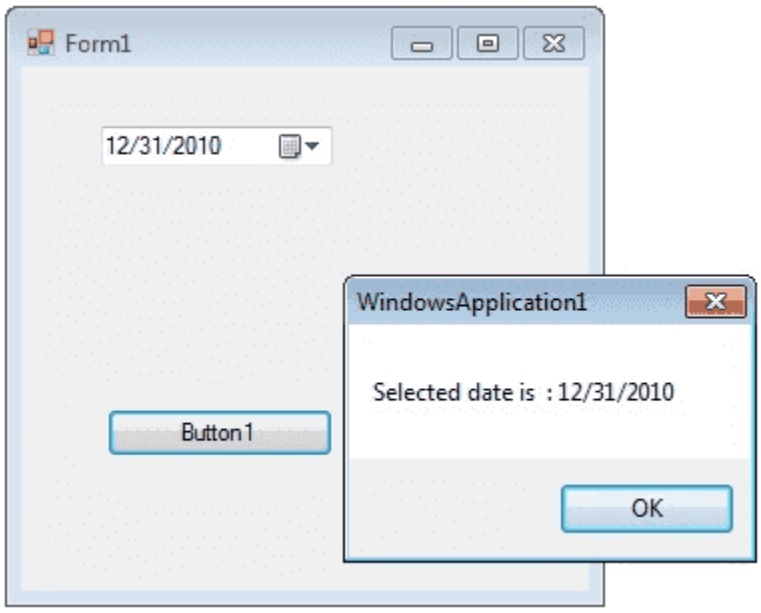
The following C# program shows a TextBox control with scrollbars.

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            textBox1.Multiline = true;
```

```
            textBox1.ScrollBars = ScrollBars.Both;
        }
    }
}
```

C# DateTimePicker Control

The DateTimePicker control allows you to display and collect date and time from the user with a specified format.



The DateTimePicker control has two parts, a label that displays the selected date and a popup calendar that allows users to select a new date. The most important property of the DateTimePicker is the Value property, which holds the selected date and time.

```
dateTimePicker1.Value = DateTime.Today;
```

The Value property contains the current date and time the control is set to. You can use the Text property or the appropriate member of Value to get the date and time value.

```
DateTime iDate;

iDate = dateTimePicker1.Value;
```

The control can display one of several styles, depending on its property values. The values can be displayed in four formats, which are set by the Format property: Long, Short, Time, or Custom.

```
dateTimePicker1.Format = DateTimePickerFormat.Short;
```

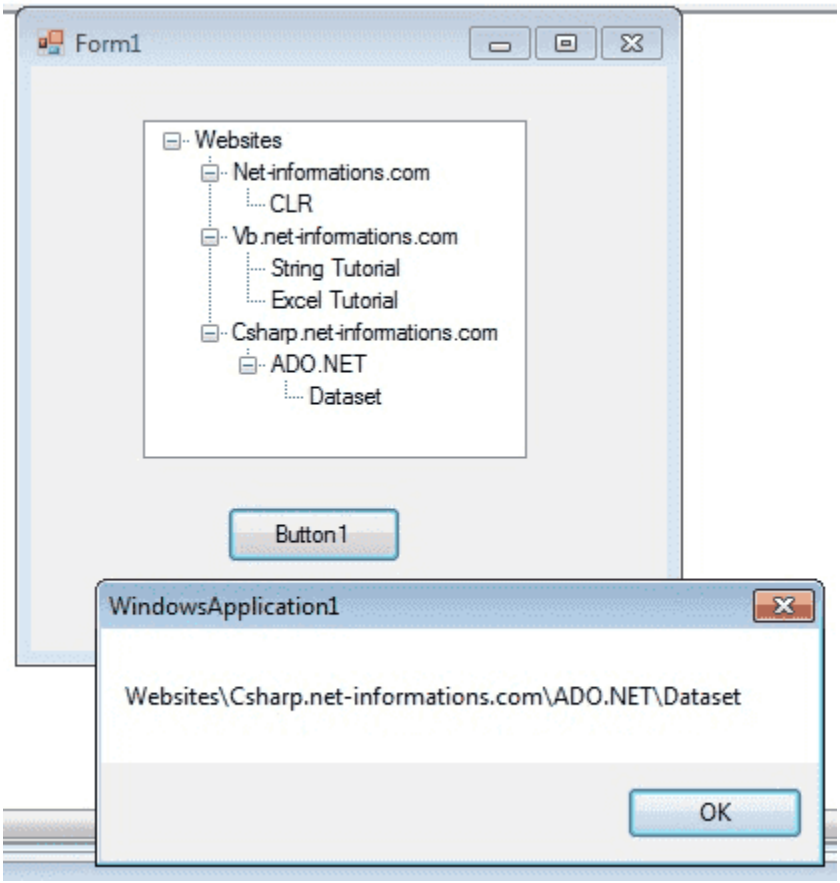
The following C# program shows how to set and get the value of a DateTimePicker1 control.

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            dateTimePicker1.Format = DateTimePickerFormat.Short;
            dateTimePicker1.Value = DateTime.Today;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            DateTime iDate;
            iDate = dateTimePicker1.Value;
            MessageBox.Show("Selected date is " + iDate);
        }
    }
}
```

C# Treeview Control

The TreeView control contains a hierarchy of TreeViewItem controls. It provides a way to display information in a hierarchical structure by

using collapsible nodes . The top level in a tree view are root nodes that can be expanded or collapsed if the nodes have child nodes.



You can explicitly define the TreeView content or a data source can provide the content. The user can expand the TreeNode by clicking the plus sign (+) button, if one is displayed next to the TreeNode, or you can expand the TreeNode by calling the TreeNode.Expand method. You can also navigate through tree views with various properties: FirstNode, LastNode, NextNode, PrevNode, NextVisibleNode, PrevVisibleNode.

The fullpath method of treeview control provides the path from root node to the selected node.

```
treeView1.SelectedNode.FullPath.ToString ();
```

Tree nodes can optionally display check boxes. To display the check boxes, set the CheckBoxes property of the TreeView to true.

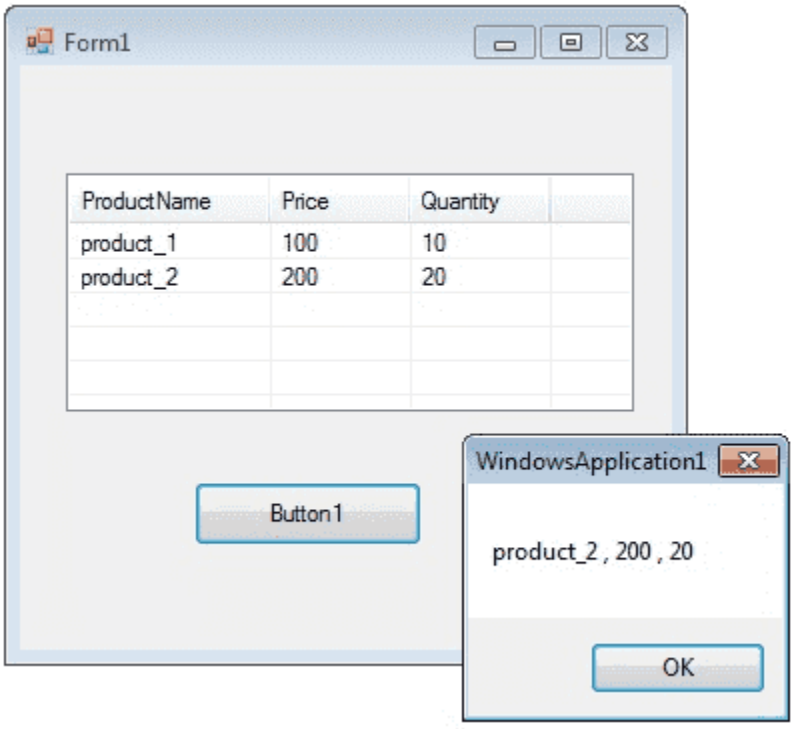
```
treeView1.CheckBoxes = true;
```

The following C# program shows a simple demonstration of treeview control

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            TreeNode tNode ;
            tNode = treeView1.Nodes.Add("Websites");

            treeView1.Nodes[0].Nodes.Add("Net-informations.com");
            treeView1.Nodes[0].Nodes[0].Nodes.Add("CLR");

            treeView1.Nodes[0].Nodes.Add("Vb.net-informations.com");
            treeView1.Nodes[0].Nodes[1].Nodes.Add("String Tutorial");
            treeView1.Nodes[0].Nodes[1].Nodes.Add("Excel Tutorial");

            treeView1.Nodes[0].Nodes.Add("Csharp.net-informations.com");
            treeView1.Nodes[0].Nodes[2].Nodes.Add("ADO.NET");
            treeView1.Nodes[0].Nodes[2].Nodes[0].Nodes.Add("Dataset");
        }

        private void button1_Click(object sender, EventArgs e)
        {
            MessageBox.Show(treeView1.SelectedNode.FullPath.ToString ());
        }
    }
}
```

C# ListView Control

The ListView control is an ItemsControl that is derived from ListBox.



Add Columns in ListView

You can add columns in Listview by using Columns.Add() method. This method takes two arguments, first one is the Column heading and second one the column width.

```
listView1.Columns.Add("ProductName", 100);
```

In the above code, "ProductName" is column heading and 100 is column width.

Add Item in Listview

You can add items in listbox using ListViewItem which represents an item in a ListView control.

```
string[] arr = new string[4];

ListViewItem itm;
```

```
//add items to ListView

arr[0] = "product_1";

arr[1] = "100";

arr[2] = "10";

itm = new ListViewItem(arr);

listView1.Items.Add(itm);
```

Get selected item from ListView

```
productName = listView1.SelectedItems[0].SubItems[0].Text;
```

Above code will return the itme from first column of first row.

Sorting Listview Items

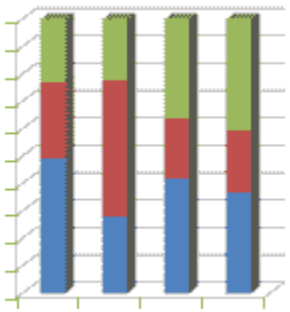
If the Sorted property of Listview is set to true, then the ListView items are sorted. The following code sorts the ListView items:

```
ListView1.Sorted = true;
```

Add Checkbox in Listview

You can add checkbox in Listview columns.

```
myListView.CheckBoxes = true;

myListView.Columns.Add(text, width, alignment);
```



ListView provides a large number of properties that provide flexibility in appearance and behavior. The View property allows you to change the way in which items are displayed. The SelectionMode property for a ListView determines how many items a user can select at one time.

The following C# program first set its view property as Details and GridLines property as true and FullRowSelect as true.

```
listView1.View = View.Details;

listView1.GridLines = true;

listView1.FullRowSelect = true;
```

Finally at the button click event, it will display the selected row values in a message box.

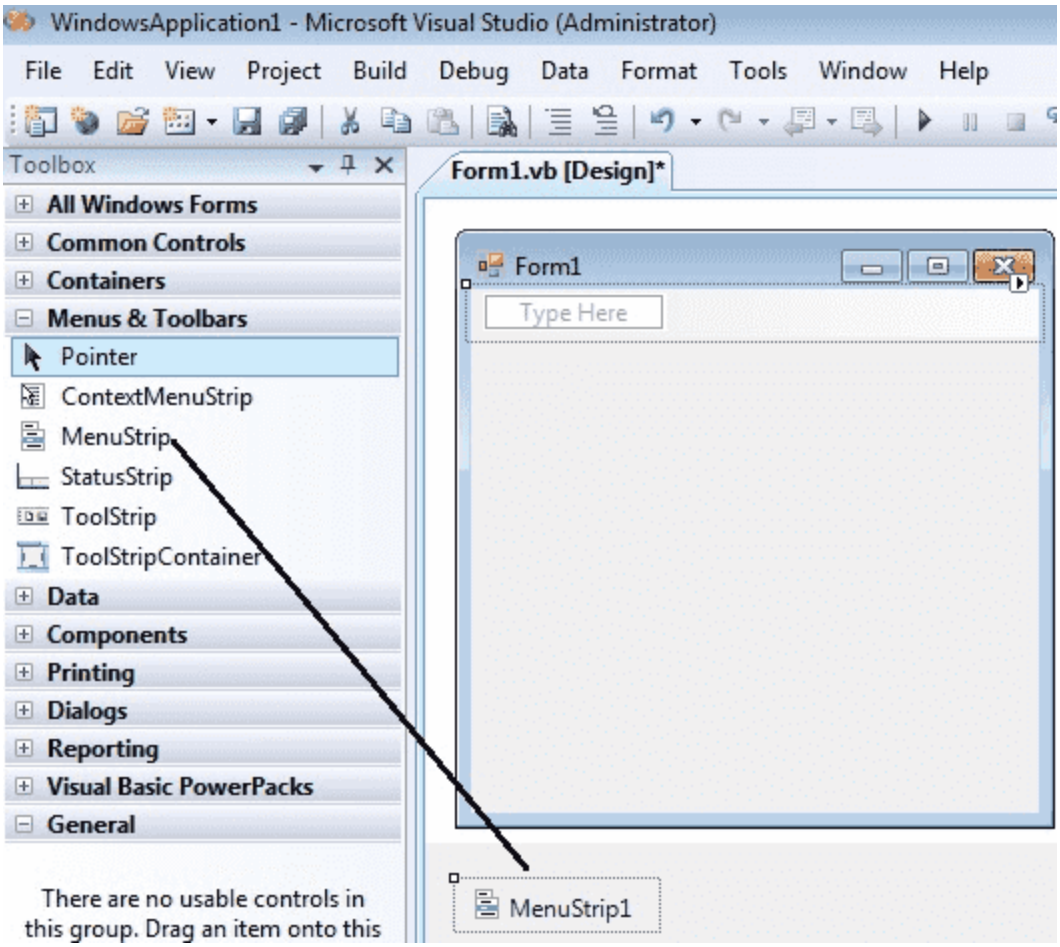
```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            listView1.View = View.Details;
            listView1.GridLines = true;
            listView1.FullRowSelect = true;

            //Add column header
            listView1.Columns.Add("ProductName", 100);
            listView1.Columns.Add("Price", 70);
            listView1.Columns.Add("Quantity", 70);

            //Add items in the listview
            string[] arr = new string[4];
            ListViewItem itm ;

            //Add first item
            arr[0] = "product_1";
            arr[1] = "100";
```

```
            arr[2] = "10";
            itm = new ListViewItem(arr);
            listView1.Items.Add(itm);

            //Add second item
            arr[0] = "product_2";
            arr[1] = "200";
            arr[2] = "20";
            itm = new ListViewItem(arr);
            listView1.Items.Add(itm);
        }

        private void button1_Click(object sender, EventArgs e)
        {
            string productName = null;
            string price = null;
            string quantity = null;

            productName = listView1.SelectedItems[0].SubItems[0].Text;
            price = listView1.SelectedItems[0].SubItems[1].Text;
            quantity = listView1.SelectedItems[0].SubItems[2].Text;

            MessageBox.Show (productName + " , " + price + " , " +
quantity);
        }
    }
}
```

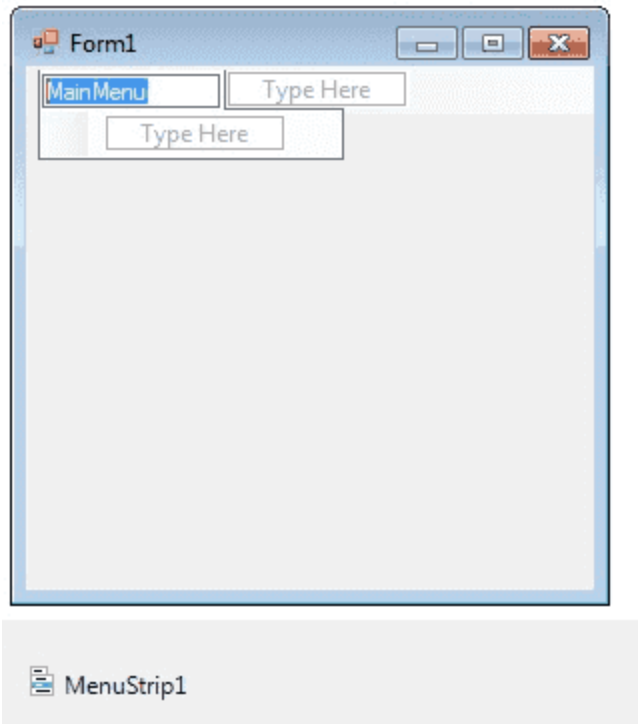
C# Menu Control

A Menu on a Windows Form is created with a MainMenu object, which is a collection of MenuItem objects. MainMenu is the container for the Menu structure of the form and menus are made of MenuItem objects that represent individual parts of a menu.

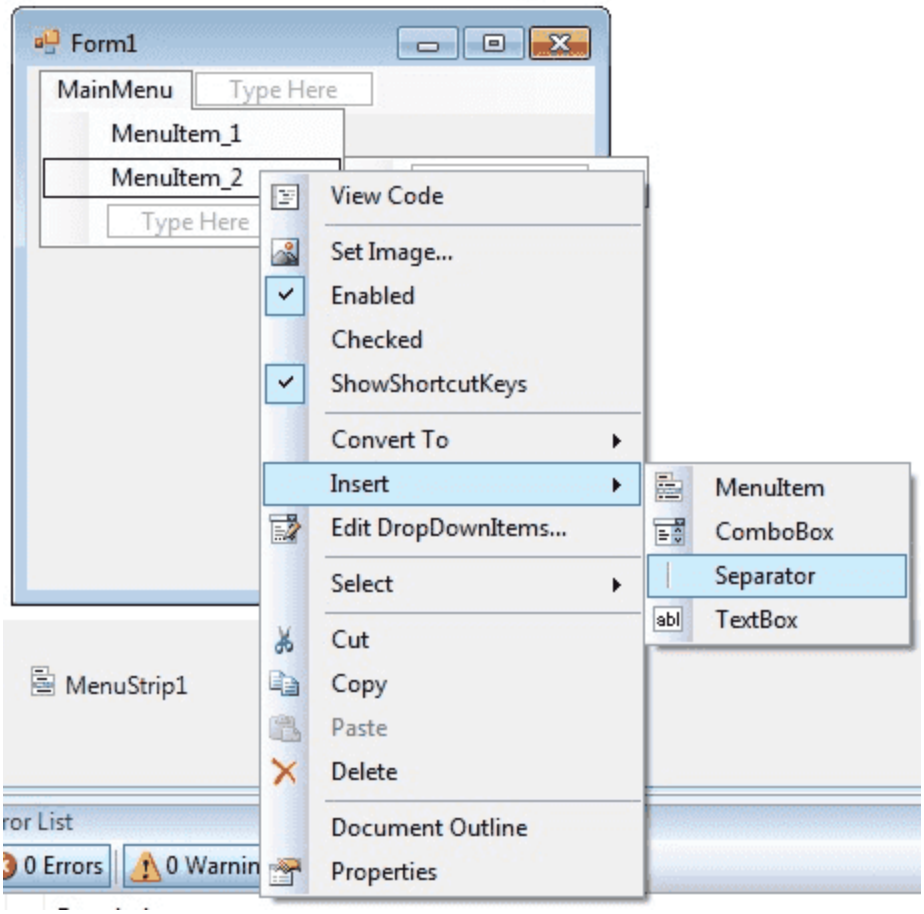
You can add menus to Windows Forms at design time by adding the MainMenu component and then appending menu items to it using the Menu Designer.



After drag the Menustrip on your form you can directly create the menu items by type a value into the "Type Here" box on the menubar part of your form. From the following picture you can understand how to create each menu items on mainmenu Object.



If you need a seperator bar , right click on your menu then go to insert->Seperator.



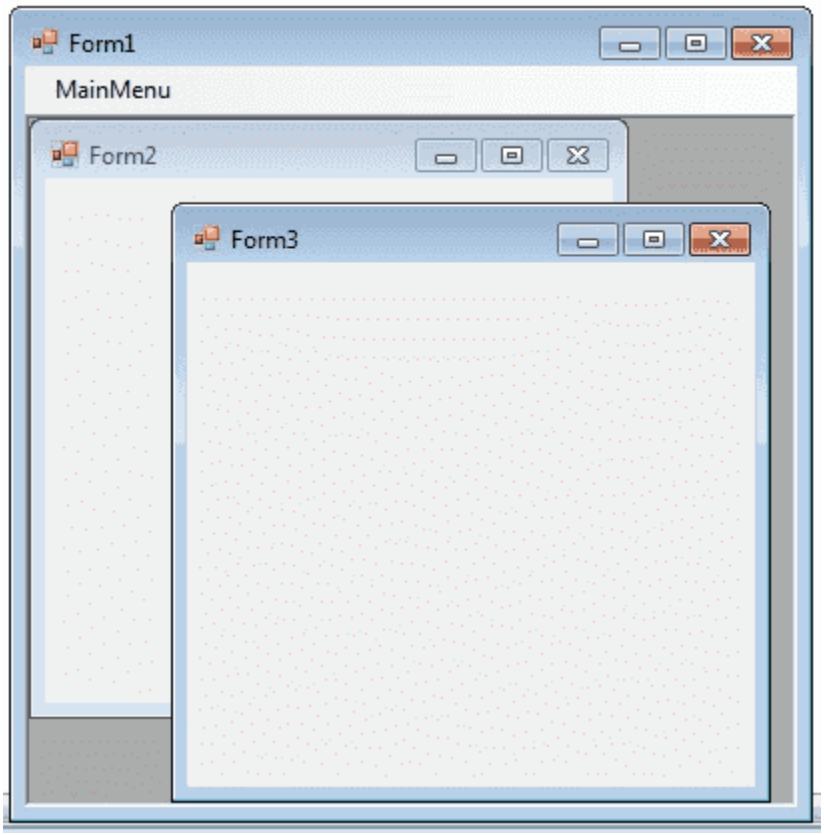
After creating the Menu on the form , you have to double click on each menu item and write the programs there depends on your requirements. The following C# program shows how to show a messagebox when clicking a Menu item.

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void menu1ToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            MessageBox.Show("You are selected MenuItem_1");
```

```
            }
        }
}
```

C# MDI Form

A Multiple Document Interface (MDI) programs can display multiple child windows inside them. This is in contrast to single document interface (SDI) applications, which can manipulate only one document at a time. Visual Studio Environment is an example of Multiple Document Interface (MDI) and notepad is an example of an SDI application. MDI applications often have a Window menu item with submenus for switching between windows or documents.



Any windows can become an MDI parent, if you set the IsMdiContainer property to True.

IsMdiContainer = true;

The following C# program shows a MDI form with two child forms. Create a new C# project, then you will get a default form Form1 . Then add two mnore forms in the project (Form2 , Form 3) . Create a Menu on your form and call these two forms on menu click event. Click here to see how to create a Menu on your form How to Menu Control C#.

NOTE: If you want the MDI parent to auto-size the child form you can code like this.

```
form.MdiParent = this;

form.Dock=DockStyle.Fill;

form.Show();
```

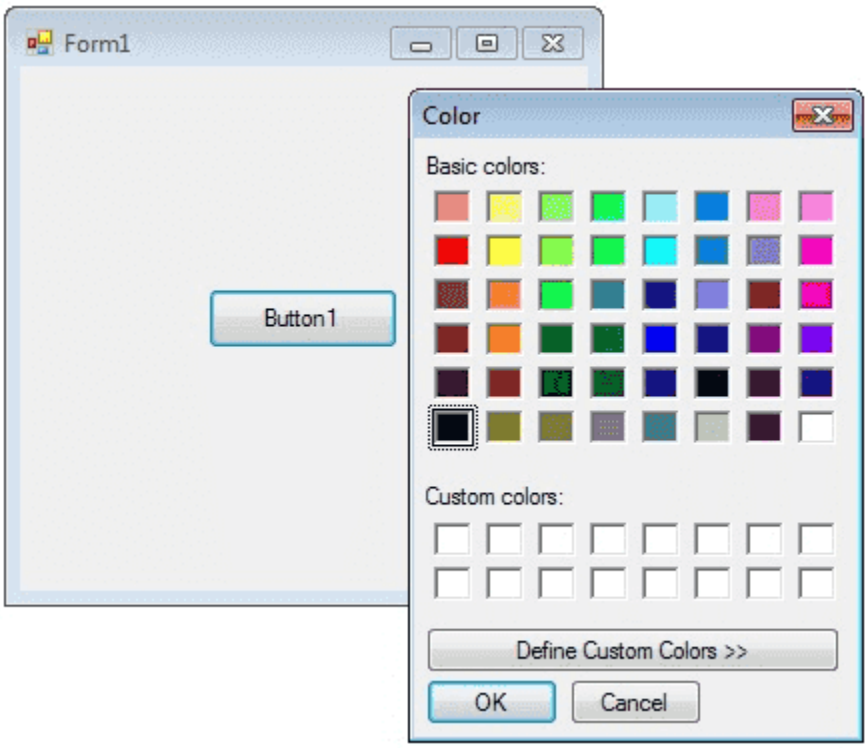
code:

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            IsMdiContainer = true;
        }
```

```
        private void menu1ToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            Form2 frm2 = new Form2();
            frm2.Show();
            frm2.MdiParent = this;
        }

        private void menu2ToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            Form3 frm3 = new Form3();
            frm3.Show();
            frm3.MdiParent = this;
        }
    }
}
```

Color dialogBox

There are several classes that implement common dialog boxes, such as color selection , print setup etc.



A ColorDialog object is a dialog box with a list of colors that are defined for the display system. The user can select or create a particular color from the list, which is then reported back to the application when the dialog box exits. You can invite a color dialog box by calling ShowDialog() method.

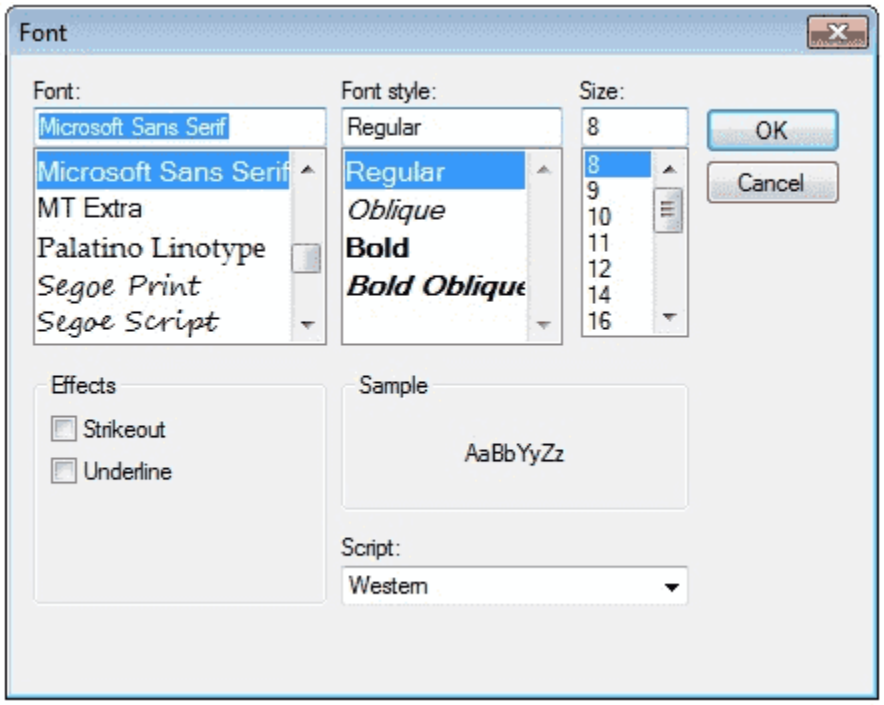
```
ColorDialog dlg = new ColorDialog();

dlg.ShowDialog();
```

The following C# program invites a color dialog box and retrieve the selected color to a string.

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            ColorDialog dlg = new ColorDialog();
            dlg.ShowDialog();

            if (dlg.ShowDialog() == DialogResult.OK)
            {
                string str = null;
                str = dlg.Color.Name;
                MessageBox.Show (str);
            }
        }
    }
}
```

C# Font Dialog Box

Font dialog box represents a common dialog box that displays a list of fonts that are currently installed on the system. The Font dialog box lets the user choose attributes for a logical font, such as font family and associated font style, point size, effects , and a script .



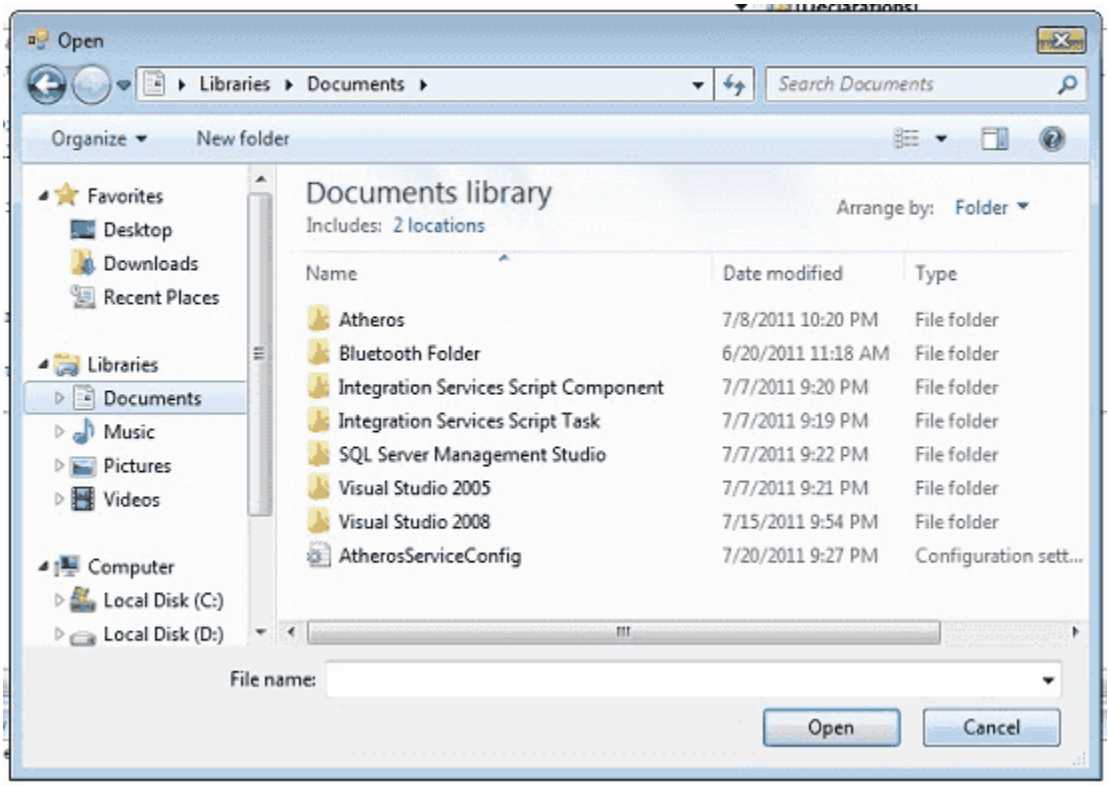
The following C# program invites a Font Dialog Box and retrieve the selected Font Name and Font Size.

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
```

```
            FontDialog dlg = new FontDialog();
            dlg.ShowDialog();

            if (dlg.ShowDialog() == DialogResult.OK)
            {
                string fontName;
                float  fontSize;
                fontName = dlg.Font.Name;
                fontSize = dlg.Font.Size;
                MessageBox.Show(fontName + "    " + fontSize );
            }
        }
    }
}
```

C# OpenFile Dialog Box

The OpenFileDialog component allows users to browse the folders of their computer or any computer on the network and select one or more files to open. The dialog box returns the path and name of the file the user selected in the dialog box.



The FileName property can be set prior to showing the dialog box. This causes the dialog box to initially display the given filename. In most cases, your applications should set the InitialDirectory, Filter, and FilterIndex properties prior to calling ShowDialog.

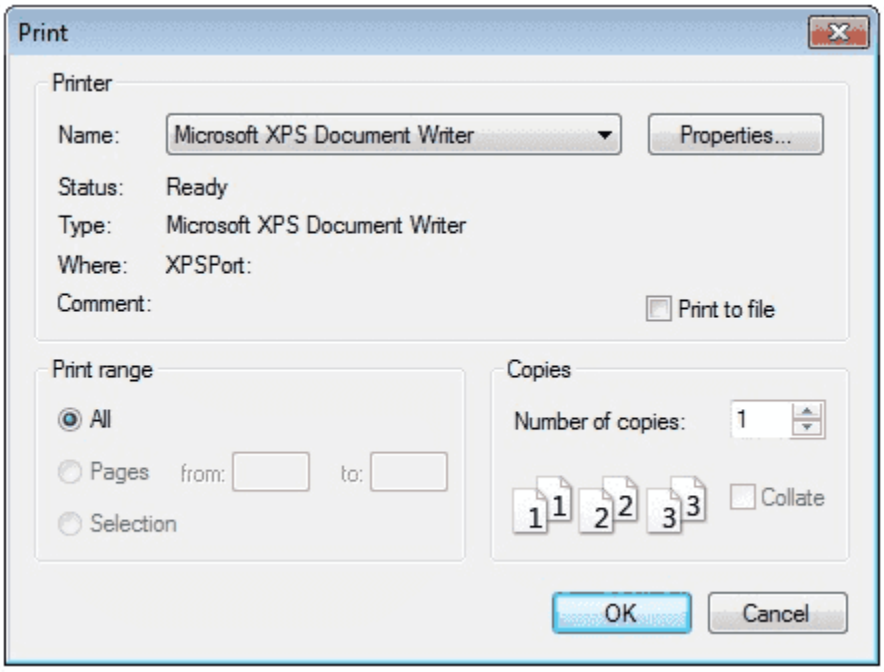
The following C# program invites an OpenFile Dialog Box and retrieve the selected filename to a string.

```
using System;
using System.Drawing;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            OpenFileDialog dlg = new OpenFileDialog();
            dlg.ShowDialog();

            if (dlg.ShowDialog() == DialogResult.OK)
            {
                string fileName;
                fileName = dlg.FileName;
                MessageBox.Show(fileName);
            }
        }
    }
}
```

C# Print Dialog Box

```
public sealed class PrintDialog : System.Windows.Forms.CommonDialog
```

A user can use the **Print dialog box** to select a printer, configure it, and perform a print job. Print dialog boxes provide an easy way to implement Print and **Print Setup dialog** boxes in a manner consistent with Windows standards.



The **Print dialog box** includes a Print Range group of radio buttons that indicate whether the user wants to print **all pages** , a range of pages, or only the selected text. The dialog box includes an edit control in which the user can type the number of **copies to print** . By default, the Print dialog box initially displays information about the current default printer.

```
using System;

using System.Drawing;

using System.Windows.Forms;

namespace Win

dowsFormsApplication1

{
```

```
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            PrintDialog dlg = new PrintDialog();
            dlg.ShowDialog();
        }
    }
}
```

How to print a Document in C#?

```
public class PrintDocument : System.ComponentModel.Component
```

PrintDocument object represents a **document** to be printed. It encapsulates all the information needed to print a page. They associate with the control which content can be print. They handle the events and operations of printing. Once a **PrintDocument** is created, we can set the Document property of **PrintDialog** as this document. After that we can also set other properties.

```
privatevoid PrintButton_Click(object sender, EventArgs e) {
   PrintDialog pDlg = newPrintDialog();
   PrintDocument pDoc = newPrintDocument();
   pDoc.DocumentName = "Print Document";
   pDlg.Document = pDoc;
   pDlg.AllowSelection = true;
   pDlg.AllowSomePages = true;
   if (pDlg.ShowDialog() == DialogResult.OK)
   {
      pDoc.Print();
   }
   else
   {
     MessageBox.Show("Print Cancelled");
   }
}
```

How to Format PrintDocument?

If you want to set the size of the paper:

```
printDialog.Document = printDocument;
printDocument.DefaultPageSettings.PaperSize = new PaperSize("my Document", width, height);
```

```
printDocument.DefaultPageSettings.Landscape = true;
```

How to Preview PrintDocument

With the help of **PrintPreviewDialog** You can preview a document also.

```
printPreviewDialog1.Document = printDocument; //Associate PrintPreviewDialog with PrintDocument.

printPreviewDialog1.ShowDialog(); // Show PrintPreview Dialog
```

keyPress event in C#

Handle Keyboard Input at the Form Level in C#

Windows Forms processes keyboard input by raising keyboard events in response to Windows messages. Most Windows Forms applications process keyboard input exclusively by handling the keyboard events.

How do I detect keys pressed in C#

You can detect most physical key presses by handling the KeyDown or KeyUp events. Key events occur in the following order:

- KeyDown
- KeyPress
- KeyUp

How to detect when the Enter Key Pressed in C#

The following C# code behind creates the KeyDown event handler. If the key that is pressed is the Enter key, a MessegeBox will displayed .

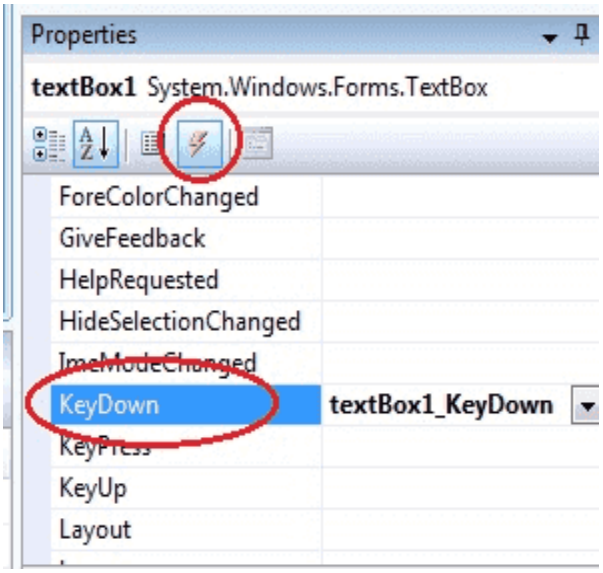
```
if (e.KeyCode == Keys.Enter)

{

MessageBox.Show("Enter Key Pressed ");
```

```
}
```

How to get TextBox1_KeyDown event in your C# source file ?

Select your TextBox control on your Form and go to Properties window. Select Event icon on the properties window and scroll down and find the KeyDown event from the list and double click the Keydown Event. The you will get the KeyDown event in your source code editor.

```
private void textBox1_KeyDown(.....) { }
```



Difference between the KeyDown Event, KeyPress Event and KeyUp Event

KeyDown Event : This event raised as soon as the user presses a key on the keyboard, it repeats while the user keeps the key depressed.

KeyPress Event : This event is raised for character keys while the key is pressed and then released. This event is not raised by noncharacter keys, unlike KeyDown and KeyUp, which are also raised for noncharacter keys

KeyUp Event : This event is raised after the user releases a key on the keyboard.

KeyPress Event :

```
using System;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      private void textBox1_KeyPress(object sender, KeyPressEventArgs e)
      {
         if (e.KeyChar == (char)Keys.Enter)
         {
            MessageBox.Show("Enter key pressed");
         }
         if (e.KeyChar == 13)
         {
            MessageBox.Show("Enter key pressed");
         }
      }
   }
}
```

KeyDown Event :

```
using System;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      private void textBox1_KeyDown(object sender, KeyEventArgs e)
      {
         if (e.KeyCode == Keys.Enter)
         {
            MessageBox.Show("Enter key pressed");
         }
      }
   }
}
```

How to Detecting arrow keys in C#

In order to capture keystrokes in a Forms control, you must derive a new class that is based on the class of the control that you want, and you override the ProcessCmdKey().

```
protected override bool ProcessCmdKey(ref Message msg, Keys keyData)

{

        //handle your keys here

}
```

More about http://net-informations.com/q/faq/arrowkeys.html

KeyUp Event :

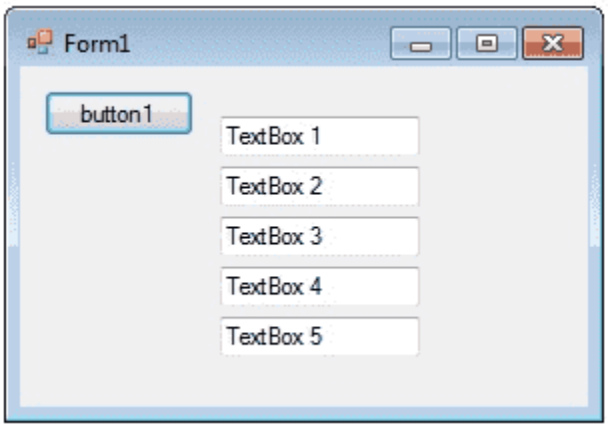
The following C# source code shows how to capture Enter KeyDown event from a TextBox Control.

```
using System;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void textBox1_KeyUp(object sender, KeyEventArgs e)
        {
            if (e.KeyCode == Keys.Enter)
            {
                MessageBox.Show("Enter key pressed");
            }
        }
    }
}
```

How to create Dynamic Controls in C# ?

How to create Control Arrays in C# ?

Visual Studio .NET does not have control arrays like Visual Basic 6.0 does. The good news is that you can still set things up to do similar things. The advantages of C# dynamic controls is that they can be created in response to how the user interacts with the application. Common controls that are added during run-time are the Button and TextBox controls. But of course, nearly every C# control can be created dynamically.



How to create Dynamic Controls in C# ?

The following program shows how to create a dynamic TextBox control in C# and setting the properties dynamically for each TextBox control. Drag a Button control in the form and copy and paste the following source code . Here each Button click the program create a new TextBox control dyanmically.

```
using System;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        int cLeft = 1;

        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
```

```
        {
            AddNewTextBox();
        }

        public System.Windows.Forms.TextBox AddNewTextBox()
        {
            System.Windows.Forms.TextBox txt = new
System.Windows.Forms.TextBox();
            this.Controls.Add(txt);
            txt.Top = cLeft * 25;
            txt.Left = 100;
            txt.Text = "TextBox " + this.cLeft.ToString();
            cLeft = cLeft + 1;
            return txt;
        }
    }
}
```

Keep Form on Top of All Other Windows

The System.Windows.Forms namespace contains classes for creating Windows-based applications that take full advantage of the rich user interface features available in the Microsoft Windows operating system. You can bring a Form on top of application by simply setting the Form.topmost form property to true will force the form to the top layer of the screen, while leaving the user able to enter data in the forms below.

```
Form2 frm = new Form2();

frm.TopMost = true;

frm.Show();
```

Topmost forms are always displayed at the highest point in the z-order of the windows on the desktop. You can use this property to create a form that is always displayed in your application, such as a MessageBox window.

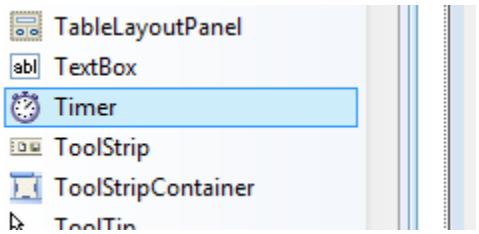
```
using System;
using System.Windows.Forms;
```

```
namespace WindowsFormsApplication1
{
        public partial class Form1 : Form
        {
                public Form1()
                {
                        InitializeComponent();
                }

                private void button1_Click(object sender, EventArgs e)
                {
                        Form2 frm = new Form2();
                        frm.TopMost = true;
                        frm.Show();
                }
        }
}
```

C# Timer Control

What is C# Timer Control ?

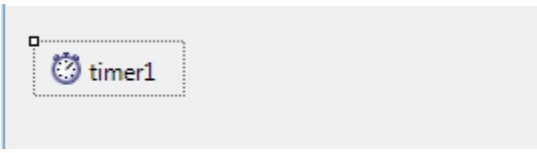
The **Timer Control** plays an important role in the development of programs both Client side and Server side development as well as in Windows Services. With the **Timer Control** we can raise events at a specific interval of time without the interaction of another thread.



Use of Timer Control

We require **Timer Object** in many situations on our development environment. We have to use Timer Object when we want to set an interval between events, periodic checking, to start a process at a fixed time schedule, to increase or decrease the speed in an animation graphics with **time schedule** etc.

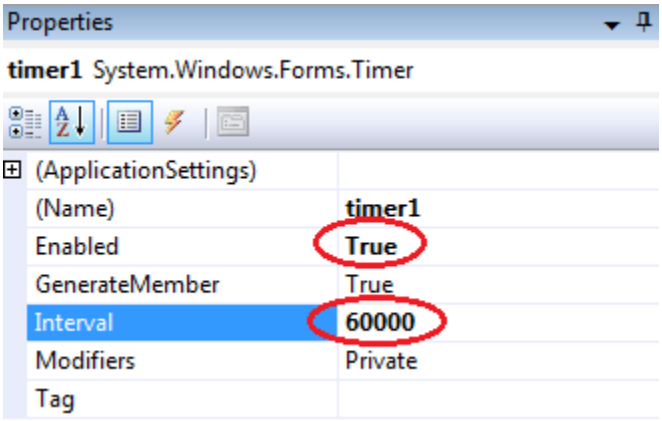
A **Timer control** does not have a visual representation and works as a component in the background.



How to use C# Timer Control ?

We can control programs with Timer Control in **millisecond** , seconds, minutes and even in hours. The Timer Control allows us to set **Interval property** in milliseconds. That is, one second is equal to 1000 milliseconds. For example, if we want to set an interval of 1 minute we set the value at Interval property as 60000, means 60x1000 .

By default the **Enabled property** of Timer Control is False. So before running the program we have to set the Enabled property is True , then only the Timer Control starts its function.



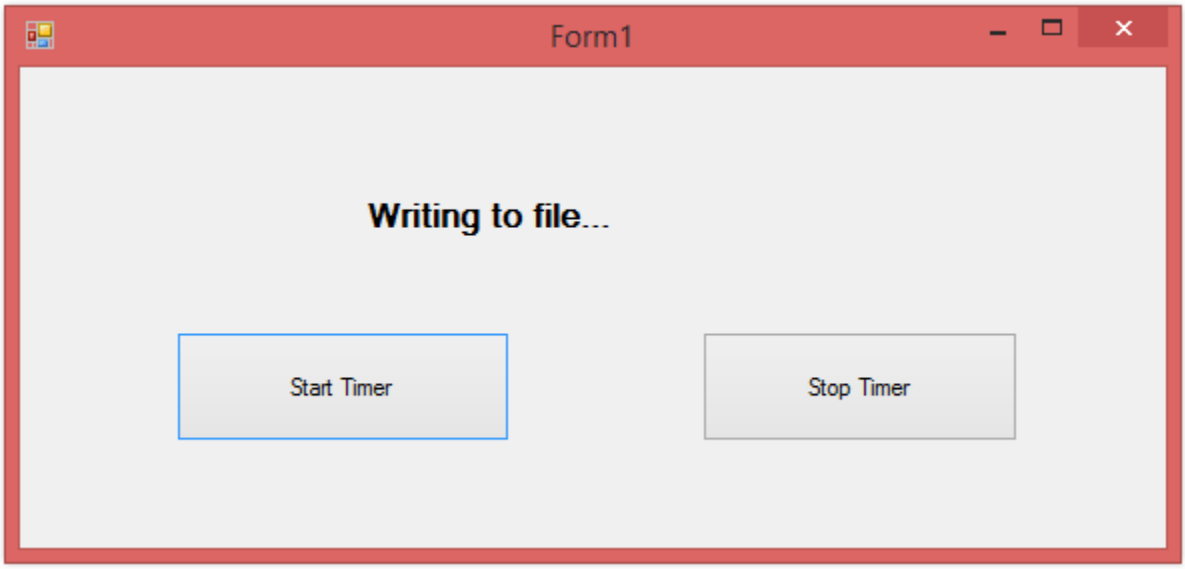
C# Timer example

In the following program we display the **current time** in a Label Control. In order to develop this program, we need a **Timer Control** and a Label Control. Here we set the timer interval as 1000 milliseconds, that means one second, for displaying current system time in Label control for the interval of one second.

```
using System;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      private void timer1_Tick(object sender, EventArgs e)
      {
         label1.Text = DateTime.Now.ToString();
      }
   }
}
```

Start and Stop Timer Control

The Timer control have included the Start and Stop methods for start and stop the Timer control functions. The following **C# program** shows how to use a timer to write some text to a text file each seconds. The program has two buttons, Start and Stop. The application will write a line to a text file every 1 second once the **Start button** is clicked. The application stops writing to the text file once the **Stop button** is clicked.




```
using System;
using System.Windows.Forms;
using System.IO;
namespace WindowsFormsApplication1
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      TextWriter tsw;
      private void Form1_Load(object sender, EventArgs e)
      {
         tsw = new StreamWriter(@"D:\timer.txt", true);
```

```
    }
    private void timer1_Tick(object sender, EventArgs e)
    {
      label1.Text = DateTime.Now.ToString();
      tsw.WriteLine(DateTime.Now.ToString());
    }
    private void button1_Click(object sender, EventArgs e)
    {
      timer1.Interval = 1000;
      timer1.Start();
    }
    private void button2_Click(object sender, EventArgs e)
    {
      timer1.Stop();
      tsw.Close();
    }
  }
}
```

Timer Tick Event

Timer **Tick event** occurs when the specified timer interval has elapsed and the timer is enabled.

```
myTimer.Tick += new EventHandler(TimerEventProcessor);
```

Timer Elapsed Event

Timer **Elapsed event** occurs when the interval elapses. The Elapsed event is raised if the Enabled property is true and the **time interval** (in milliseconds) defined by the Interval property elapses.

```
MyTimer.Elapsed += OnTimedEvent;
```

Timer Interval Property

Timer **Interval property** gets or sets the time, in milliseconds, before the Tick event is raised relative to the last occurrence of the **Tick event** .

```
// Sets the timer interval to 1 seconds.

myTimer.Interval = 1000;
```

Timer Reset Property

Timer **AutoReset property** gets or sets a Boolean indicating whether the Timer should raise the **Elapsed event** only once (false) or repeatedly (true).

```
MyTimer.AutoReset = false;
```

TimerCallback Delegate

Callback represents the method that handles calls from a Timer. This method does not execute in the thread that created the timer; it executes in a separate **thread pool** thread that is provided by the system.

Timer Class

System.Timers.Timer fires an event at **regular intervals** . This is a somewhat more powerful timer. Instead of a Tick event, it has the Elapsed event. The Start and Stop methods of **System.Timers.Timer** which are similar to changing the Enabled property. Unlike the System.Windows.Forms.Timer, the events are

effectively queued - the timer doesn't wait for one event to have completed before starting to wait again and then firing off the next event. The class is intended for use as a **server-based** or service component in a multithreaded environment and it has no user interface and is not visible at runtime.

The following example instantiates a **System.Timers.Timer** object that fires its Timer.Elapsed event every two seconds sets up an event handler for the event, and starts the timer.

- Create a timer object for one seconds interval.

```
myTimer = new System.Timers.Timer(1000);
```

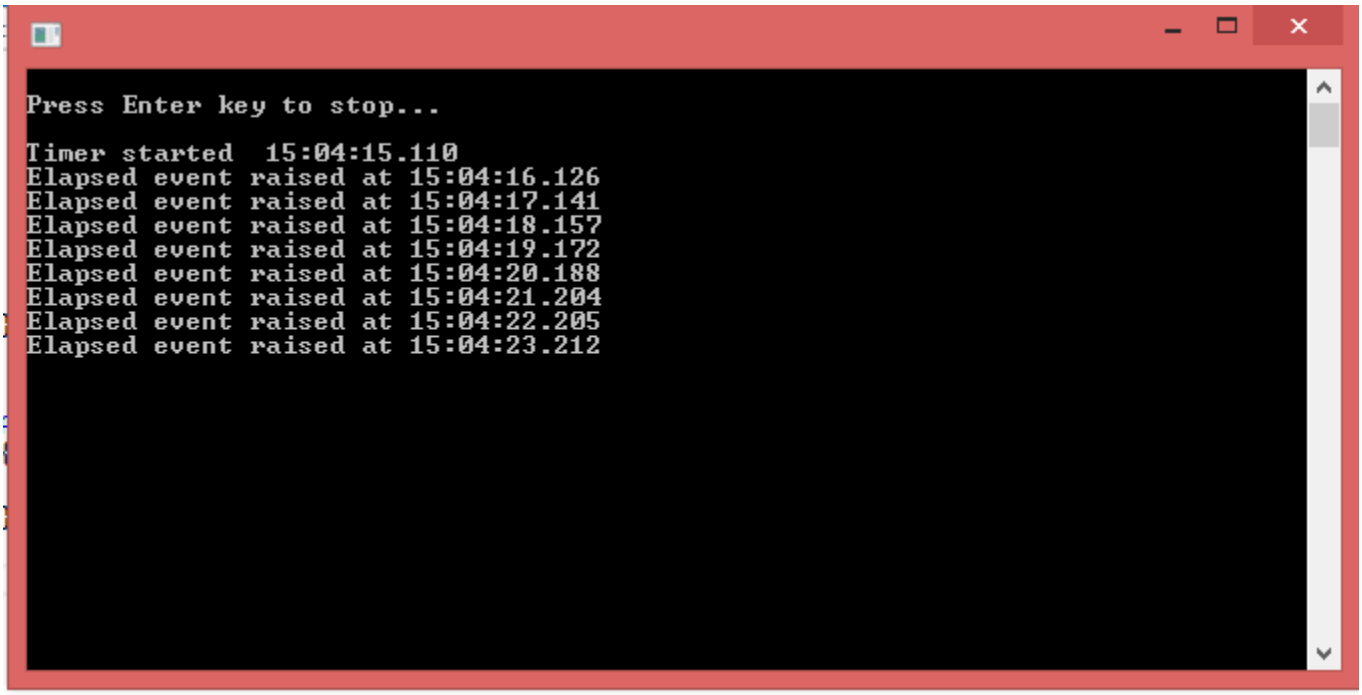
- Set elapsed event for the timer. This occurs when the interval elapses.

```
myTimer.Elapsed += OnTimedEvent;
```

- Finally, start the timer.

```
myTimer.Enabled = true;
```

C# Timer (Console Application)

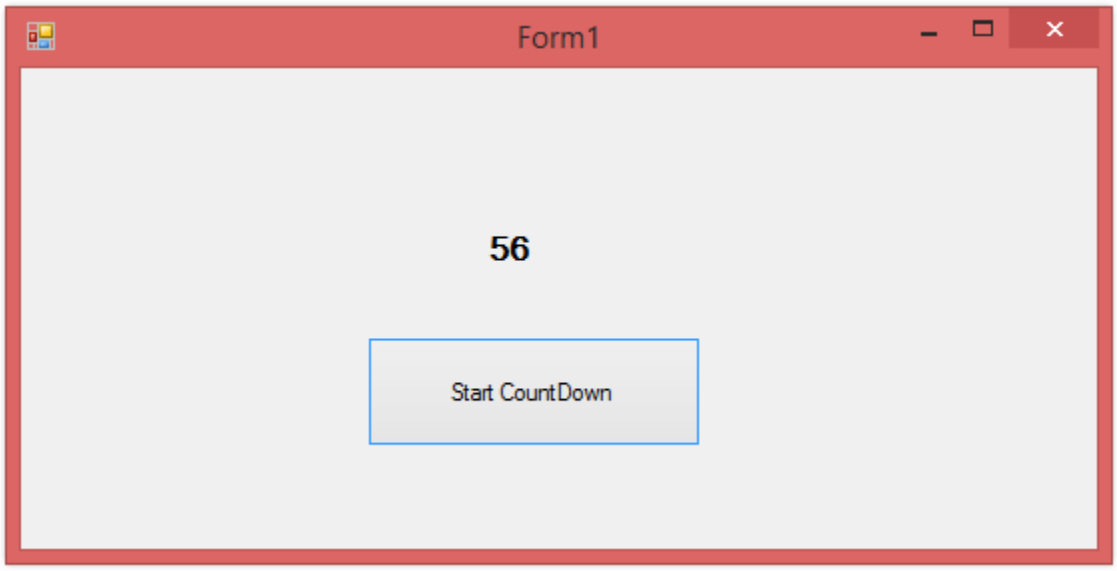



```
using System;
using System.Timers;
namespace ConsoleApplication1
{
   class Program
   {
      private static System.Timers.Timer myTimer;
      static void Main(string[] args)
      {
         myTimer = new System.Timers.Timer(1000);
         myTimer.Elapsed += OnTimedEvent;
         myTimer.AutoReset = true;
```

```
        myTimer.Enabled = true;
        Console.WriteLine("\nPress Enter key to stop...\n");
        Console.WriteLine("Timer started  {0:HH:mm:ss.fff}  ", DateTime.Now);
        Console.ReadLine();
        myTimer.Stop();
        myTimer.Dispose();
        Console.WriteLine("Finished...");
    }
    private static void OnTimedEvent(Object source, ElapsedEventArgs e)
    {
        Console.WriteLine("Elapsed event raised at {0:HH:mm:ss.fff}",e.SignalTime);
    }
  }
}
```

Count down using C# Timer

The following C# program show how to cerate a **seconds countdown** using C# Timer.




```
using System;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      private System.Windows.Forms.Timer aTimer;
      private int counter = 60;
      private void button1_Click(object sender, EventArgs e)
      {
         aTimer = new System.Windows.Forms.Timer();
         aTimer.Tick += new EventHandler(aTimer_Tick);
```

```
        aTimer.Interval = 1000; // 1 second
        aTimer.Start();
        label1.Text = counter.ToString();
    }
    private void aTimer_Tick(object sender, EventArgs e)
    {
        counter--;
        if (counter == 0)
            aTimer.Stop();
        label1.Text = counter.ToString();
    }
  }
}
```

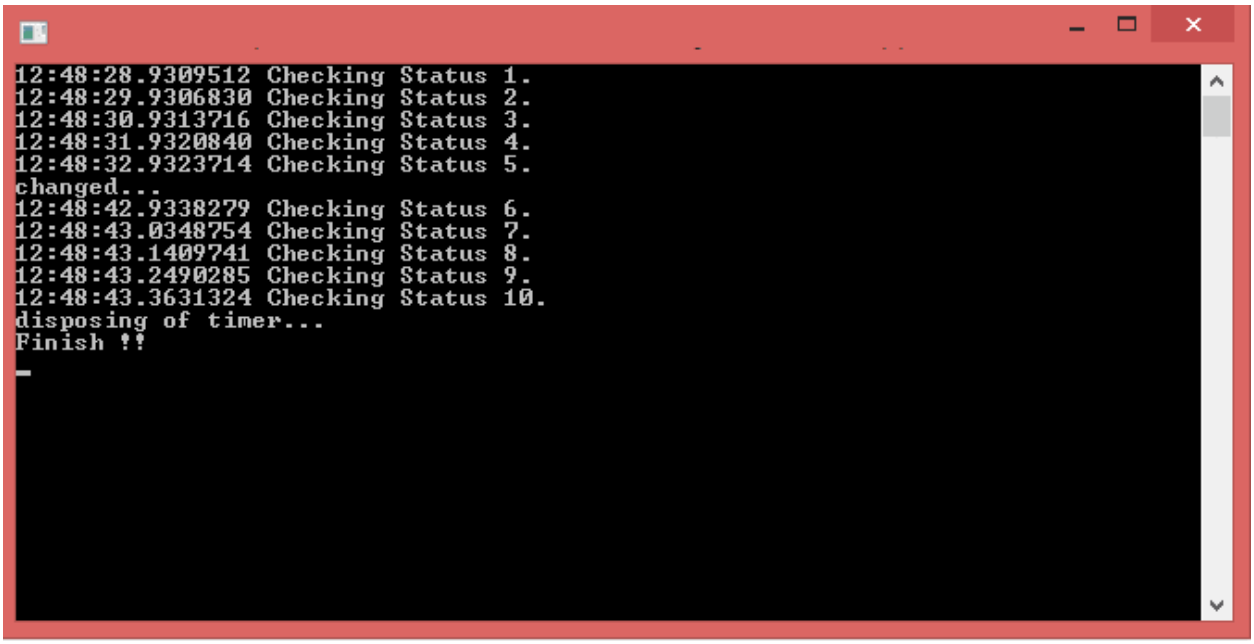
System.Threading.Timer Class

System.Threading.Timer is a simple, lightweight timer that uses **callback methods** and is served by thread pool threads. It is not recommended for use with Windows Forms, because its callbacks do not occur on the user interface thread. Use a **TimerCallback delegate** to specify the method you want the Timer to execute. The timer delegate is specified when the timer is constructed, and cannot be changed. The method does not execute on the thread that created the timer; it executes on a **ThreadPool thread** supplied by the system.

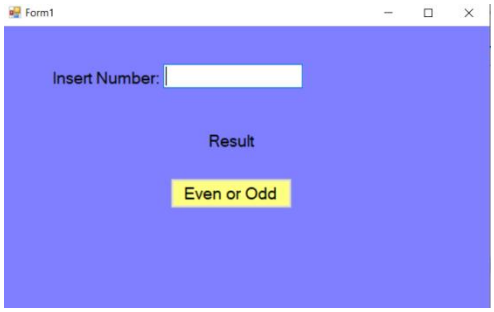
When creating a timer, the application specifies an amount of time to wait before the first invocation of the **delegate methods** , and an amount of time to wait between subsequent invocations. A timer invokes its methods when its due time elapses, and invokes its methods once per period thereafter. You can change these values, or you can disable the timer, by using the Timer.Change method.

```
using System;
using System.Threading;
namespace ConsoleApplication1
{
   class Program
   {
      static void Main(string[] args)
      {
         TimerState TS = new TimerState();
        TimerCallback tDelegate = new TimerCallback(CheckStatus);
        Timer thisTimer = new Timer(tDelegate, TS, 1000, 1000);
        TS.tmr = thisTimer;
        while (TS.tmr != null)
            Thread.Sleep(0);
        Console.WriteLine("Finish !!");
        Thread.Sleep(10000);
  }
   // The following method is called by the timer's delegate.
      static void CheckStatus(Object state)
      {
         TimerState TS = (TimerState)state;
         TS.counter++;
         Console.WriteLine("{0} Checking Status {1}.", DateTime.Now.TimeOfDay, TS.counter);
         if (TS.counter == 5)
         {
```

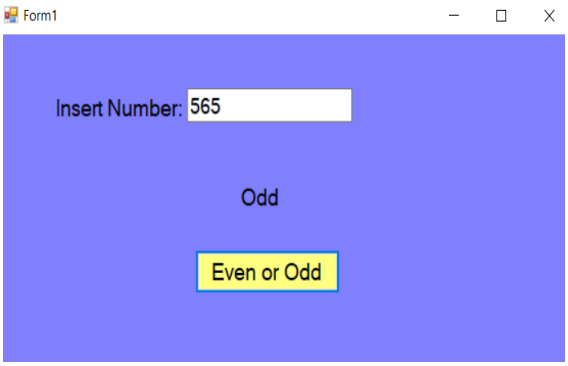
```
            (TS.tmr).Change(10000, 100);
            Console.WriteLine("changed...");
        }
        if (TS.counter == 10)
        {
            Console.WriteLine("disposing of timer...");
            TS.tmr.Dispose();
            TS.tmr = null;
        }
      }
    }
    class TimerState
    {
      public int counter = 0;
      public Timer tmr;
    }
}
```

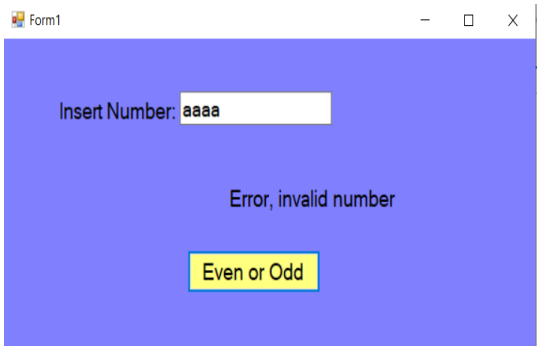

12:48:28.9309512 Checking Status 1.
12:48:29.9306830 Checking Status 2.
12:48:30.9313716 Checking Status 3.
12:48:31.9320840 Checking Status 4.
12:48:32.9323714 Checking Status 5.
changed...
12:48:42.9338279 Checking Status 6.
12:48:43.0348754 Checking Status 7.
12:48:43.1409741 Checking Status 8.
12:48:43.2490285 Checking Status 9.
12:48:43.3631324 Checking Status 10.
disposing of timer...
Finish !!

Using textbox, Label, Button application to find the number even or odd




```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApplication12
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btnChek_Click(object sender, EventArgs e)
        {
            int i;
            try
            {
                i = Convert.ToInt32(textNum.Text);
                if ((i % 2) == 0)
                {
                    lblResult.Text = "Even";
                }
                else
                {
                    lblResult.Text = "Odd";
                }

            }
            catch (Exception err)
            {
                lblResult.Text = "Error, invalid number";
            }
            }

    }
}
```






```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace carpting
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
```

```
{
    int i, j;
    for (i = 1; i <= 10; ++i)
    {
        combolength.Items.Add(i);
    }
    for (j = 1; j <= 10; ++j)
    {
        combowidth.Items.Add(j);

    }

}

private void btnarea_Click(object sender, EventArgs e)
{
    int area, width, length;
    length = Convert.ToInt32(combolength.Text);
    width = Convert.ToInt32(combowidth.Text);
    area = length * width;
    lblroomarea.Text = Convert.ToString(area) + "m";

}

private void btnroomcost_Click(object sender, EventArgs e)
{
    int width, length;
    int price = 0, roomcost;
    length = Convert.ToInt32(combolength.Text);
    width = Convert.ToInt32(combowidth.Text);
    price = Convert.ToInt32(txtprice.Text);
    roomcost = length * width * price;
    lblcarptingcost.Text = Convert.ToString(roomcost) + "$";

}

private void btnclear_Click(object sender, EventArgs e)
{
    combolength.Text = "";
    combowidth.Text = "";
    txtprice.Text = "";
    lblroomarea.Text = "..... ";
    lblcarptingcost.Text = ".....";

}
```

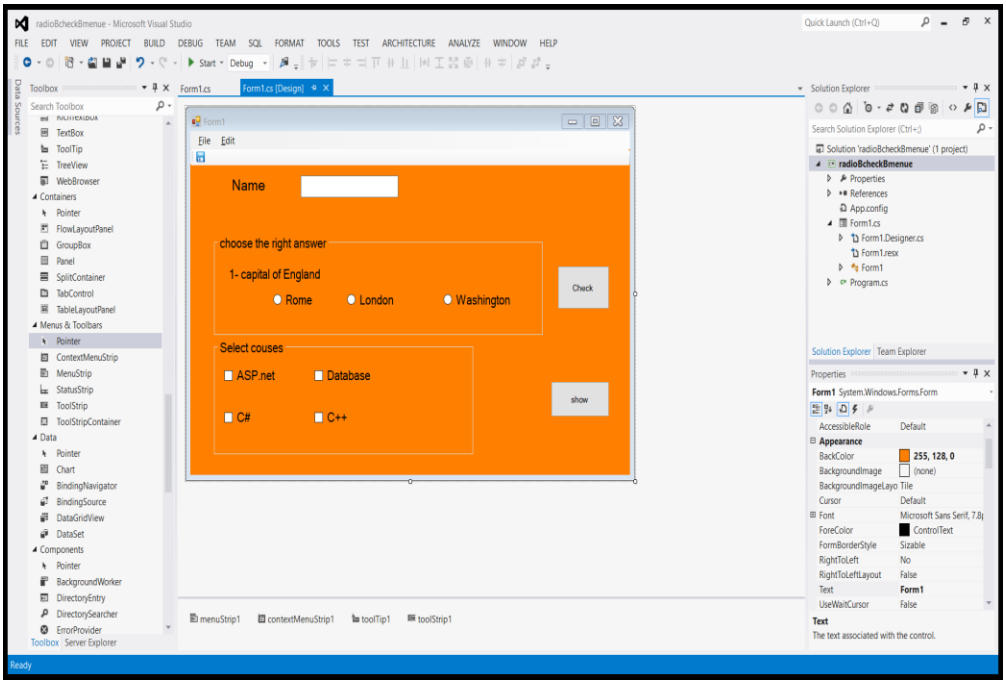
```
        private void button1_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }
    }
}
```

Radiobutton+checkbox+menustrip+contextmenu strip+toolstrip+tooltip

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace radioBcheckBmenue
{
    public partial class Form1 : Form
    {
        string tt, rr;
        public Form1()
        {
            InitializeComponent();


        }

        private void button1_Click(object sender, EventArgs e)
        {
            //if (radioButton1.Checked == true)
            //{
            //    MessageBox.Show("You selected Rome !! "+ "the right answer is London");

            //}
            //else if (radioButton2.Checked == true)
            //{
```

```
            //    MessageBox.Show("You selected London !! " + " Congrats it's the right
answer ");

            //}
            //else
            //{
            //    MessageBox.Show("You selected Washington !! " + "the right answer is
London");

            //}
            if (radioButton1.Checked == true)
            {
                rr = radioButton1.Text;

            }
            else if (radioButton2.Checked == true)
            {
                rr = radioButton2.Text;

            }
            else
            {
                rr = radioButton3.Text;


            }

            MessageBox.Show("You selected  " + rr);


        }

        private void button2_Click(object sender, EventArgs e){


             if (checkBox1.Checked == true)
            {
                tt = "  Asp.net \n ";
            }

            if (checkBox2.Checked == true)
            {
                tt += " C# \n";
            }
            if (checkBox3.Checked == true)
            {
                tt += "   Database\n";
            }
            if (checkBox4.Checked == true)
            {
                tt += "  C++\n";
            }
            MessageBox.Show(tt,"selected courses");
        }

        private void Form1_Load(object sender, EventArgs e)
        {
```

```
        }

        private void copyToolStripMenuItem_Click(object sender, EventArgs e)
        {

        }

        private void newToolStripMenuItem_Click(object sender, EventArgs e)
        {
            new Form1().Show();
        }

        private void exitToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void copyToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            textBox1.Copy();
        }

        private void cutToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            textBox1.Cut();
        }

        private void pasteToolStripMenuItem1_Click(object sender, EventArgs e)
        {
            textBox1.Paste();
        }

        private void toolTip1_Popup(object sender, PopupEventArgs e)
        {

        }
    }
}
```



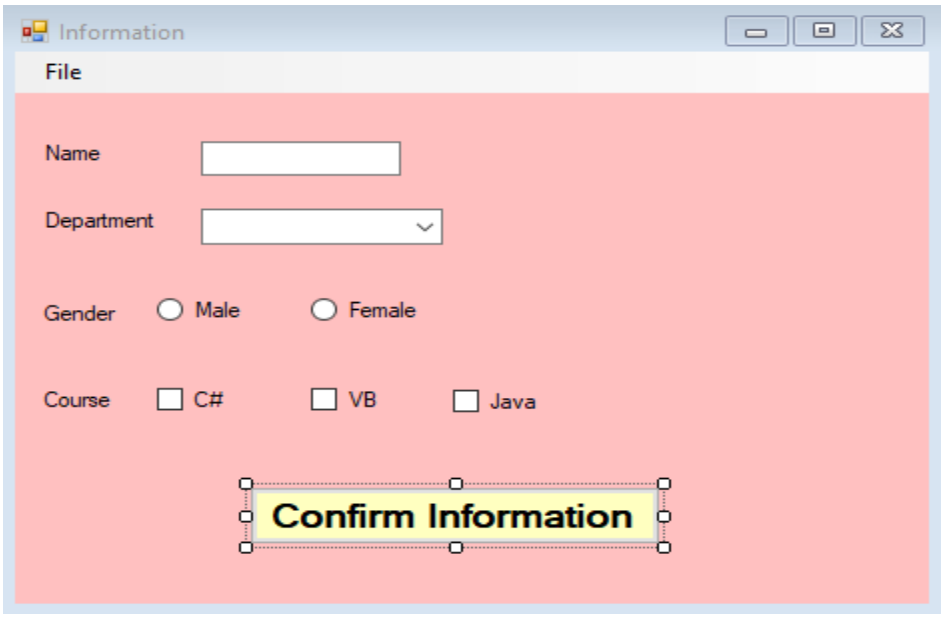
Link the textbox1 with the contextmenustrip

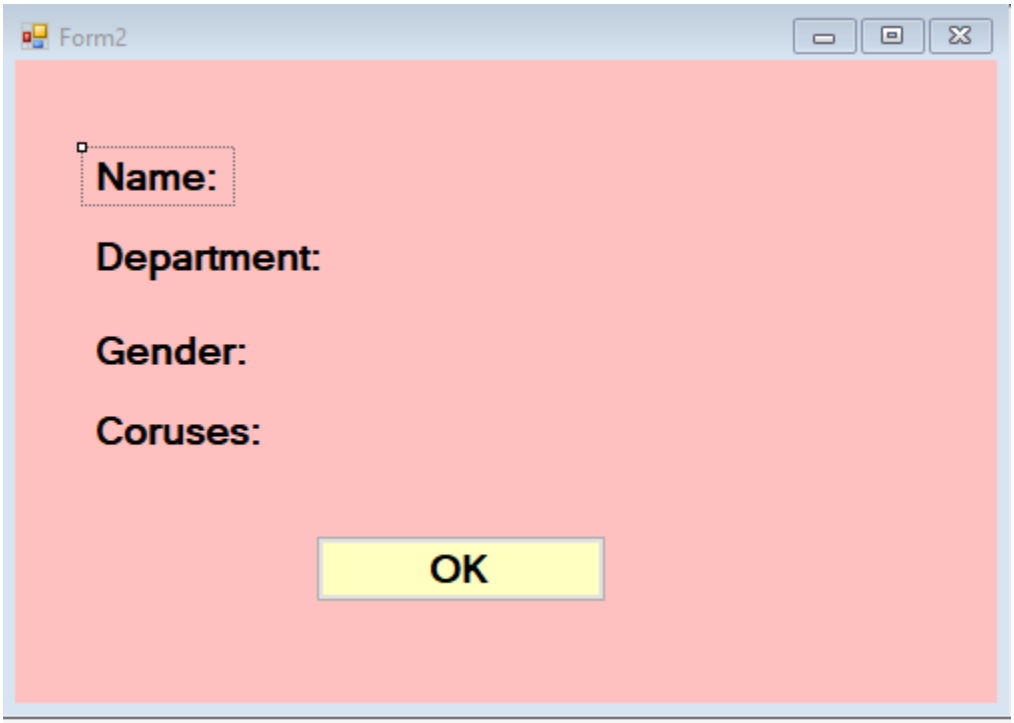
To underline the first letter add & before the word ex &File

Tooltip after adding can be combined to any tool from property the message text will be added

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApplication21
{
    public partial class Form1 : Form
    {

        public Form1()
        {
            InitializeComponent();
        }

        private void newToolStripMenuItem_Click(object sender, EventArgs e)
        {
            new Form1().Show();
        }

        private void exitToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void copyToolStripMenuItem_Click(object sender, EventArgs e)
        {
            textBox1.Copy();
        }

        private void cutToolStripMenuItem_Click(object sender, EventArgs e)
        {
            textBox1.Cut();
        }

        private void pasteToolStripMenuItem_Click(object sender, EventArgs e)
        {
            textBox1.Paste();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Form2 ff = new Form2();
            ff.Show();
            ff.lblname.Text += textBox1.Text;

            string dep;
            dep = comboBox1.Text;
            ff.lbldep.Text += dep;
```

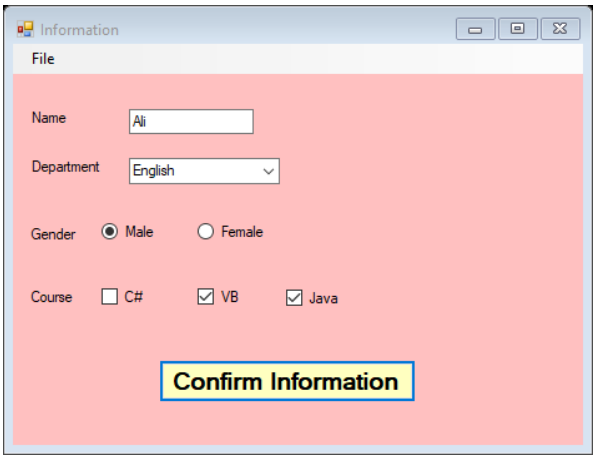
```
            if (radioButton1.Checked==true)
            {
                ff.lblgender.Text += "Male";
            }
            else

                ff.lblgender.Text += "Female";


            if (checkBox1.Checked == true)
            {
                ff.lblcourse.Text += "   C# ";
            }

            if (checkBox2.Checked == true)
            {
                ff.lblcourse.Text += "   VB";
            }
            if (checkBox3.Checked == true)
            {
                ff.lblcourse.Text += "   Java";
            }
        }


        private void Form1_Load(object sender, EventArgs e)
        {

        }
    }
}
```

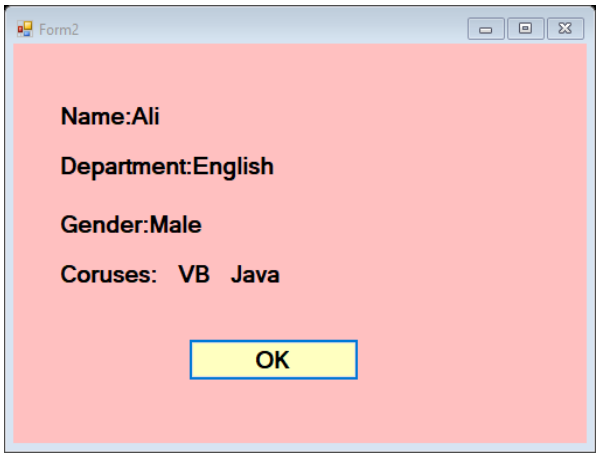




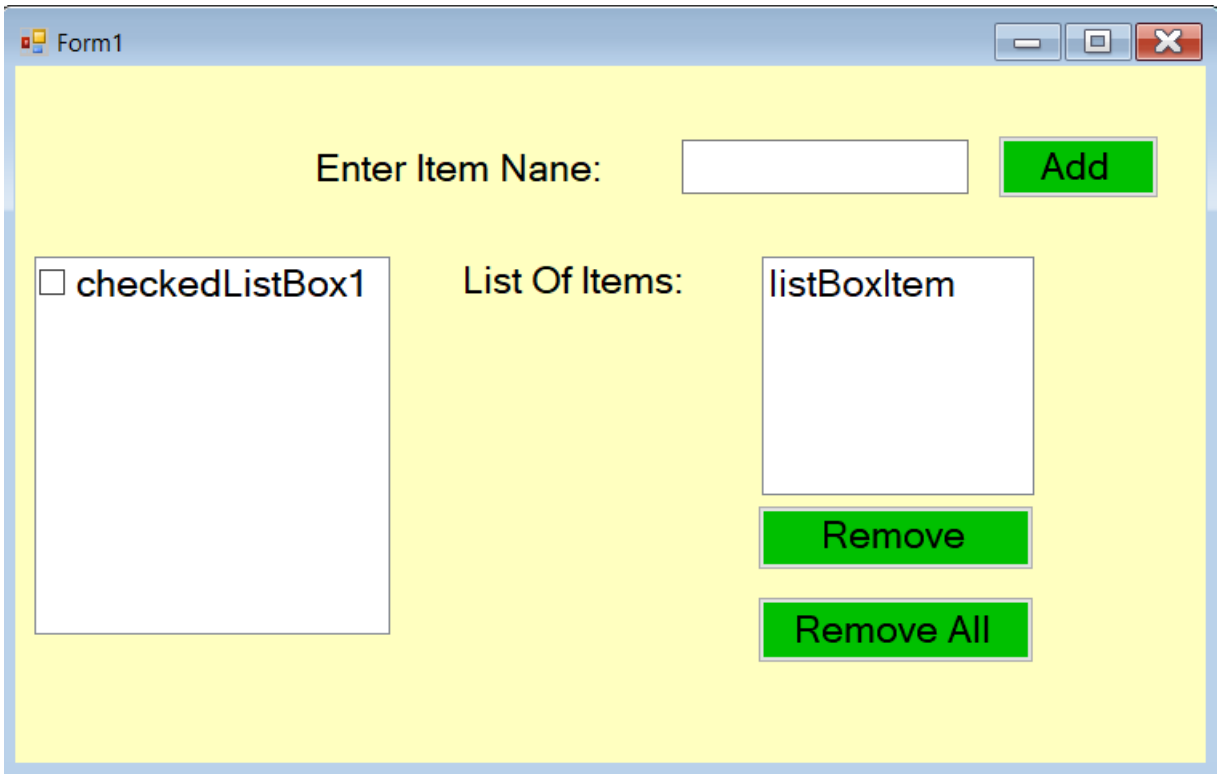
Note : the labels in the form 2 should be public (the modifiers in property = public)

When confirm information is clicked the information will be viewed in form2





ListBox and checked listBox




```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace listbox_and_checklistbox
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
```

```
        private void button1_Click(object sender, EventArgs e)
        {
           string item = txtitem.Text;
            if (item.Length <= 0)
            {
                MessageBox.Show("Please enter the Item", "ERROR", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
            else
            {
                checkedListBox1.Items.Add(item);
                txtitem.Clear();
                txtitem.Focus();
            }


        }

        private void btnrmoveselected_Click(object sender, EventArgs e)
        {
            if (listBoxItem.SelectedItems.Count== 0)
            {
                MessageBox.Show("select item first");
            }

            listBoxItem.Items.Remove(listBoxItem.SelectedItem);

        }

        private void btnremoveall_Click(object sender, EventArgs e)
        {
            listBoxItem.Items.Clear();
            txtitem.Focus();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            checkedListBox1.Items.Add("Shose");
            checkedListBox1.Items.Add("Shirts");
            checkedListBox1.Items.Add("Hats");
            checkedListBox1.Items.Add("Socks");
            checkedListBox1.Items.Add("Jackets");
        }

        private void checkedListBox1_ItemCheck(object sender, ItemCheckEventArgs e)
        {
            if (e.NewValue == CheckState.Checked)
                listBoxItem.Items.Add(checkedListBox1.SelectedItem.ToString());
            else
            {
                listBoxItem.Items.Remove(checkedListBox1.SelectedItem.ToString());

            }

        }

        private void listBoxItem_SelectedIndexChanged(object sender, EventArgs e)
        {
```

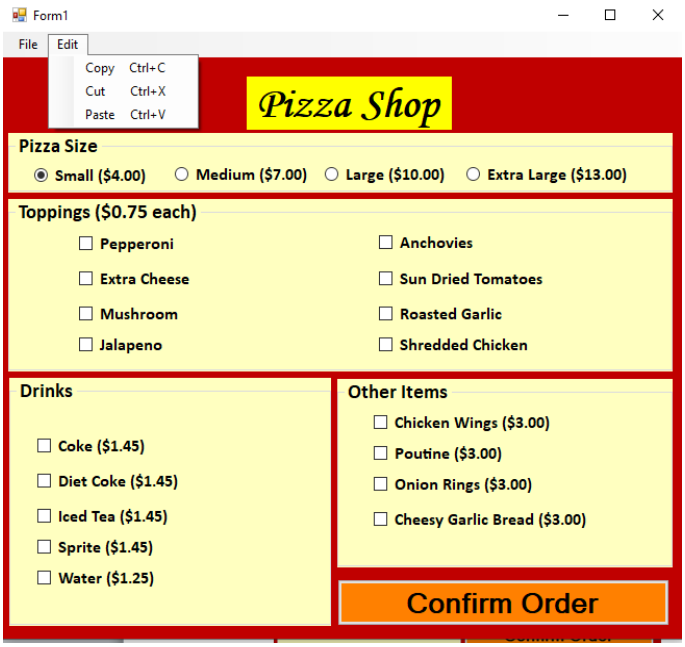
```
        }
    }

}
```

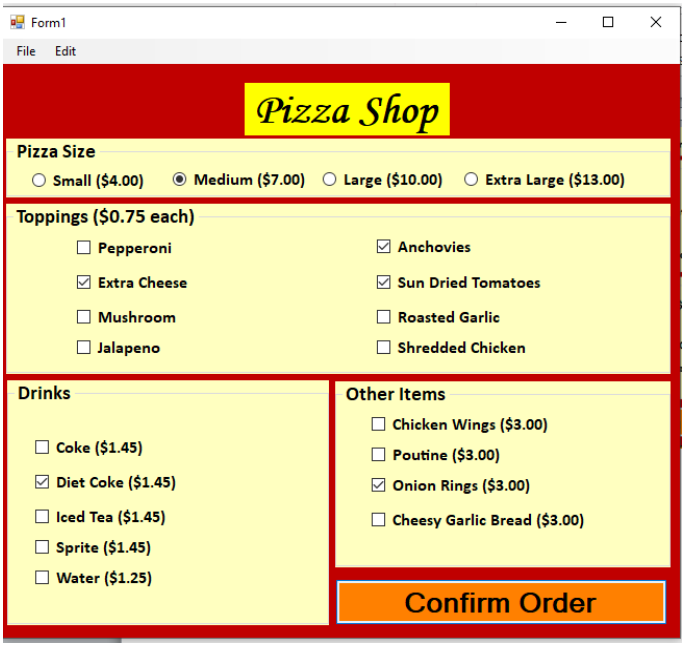
Or:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace listbox_and_checklistbox
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
           string item = txtitem.Text;
            if (item.Length <= 0)
            {
                MessageBox.Show("Please enter the Item", "ERROR", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
            else
            {
                checkedListBox1.Items.Add(item);
                txtitem.Clear();
                txtitem.Focus();
            }
```

```
        }

        private void btnrmoveselected_Click(object sender, EventArgs e)
        {
            if (listBoxItem.SelectedIndex >= 0)
            {

                listBoxItem.Items.RemoveAt(listBoxItem.SelectedIndex);
            }
            else
            {
                MessageBox.Show("Select an item first");
            }


        }

        private void btnremoveall_Click(object sender, EventArgs e)
        {
            listBoxItem.Items.Clear();
            txtitem.Focus();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            checkedListBox1.Items.Add("Shose");
            checkedListBox1.Items.Add("Shirts");
            checkedListBox1.Items.Add("Hats");
            checkedListBox1.Items.Add("Socks");
            checkedListBox1.Items.Add("Jackets");
        }

        private void checkedListBox1_ItemCheck(object sender, ItemCheckEventArgs e)
        {
            if (e.NewValue == CheckState.Checked)
                listBoxItem.Items.Add(checkedListBox1.SelectedItem.ToString());
            else
            {
                listBoxItem.Items.Remove(checkedListBox1.SelectedItem.ToString());

            }

        }

        private void listBoxItem_SelectedIndexChanged(object sender, EventArgs e)
        {

        }
    }

}
```

Q: Design and program the following windows form application using C#





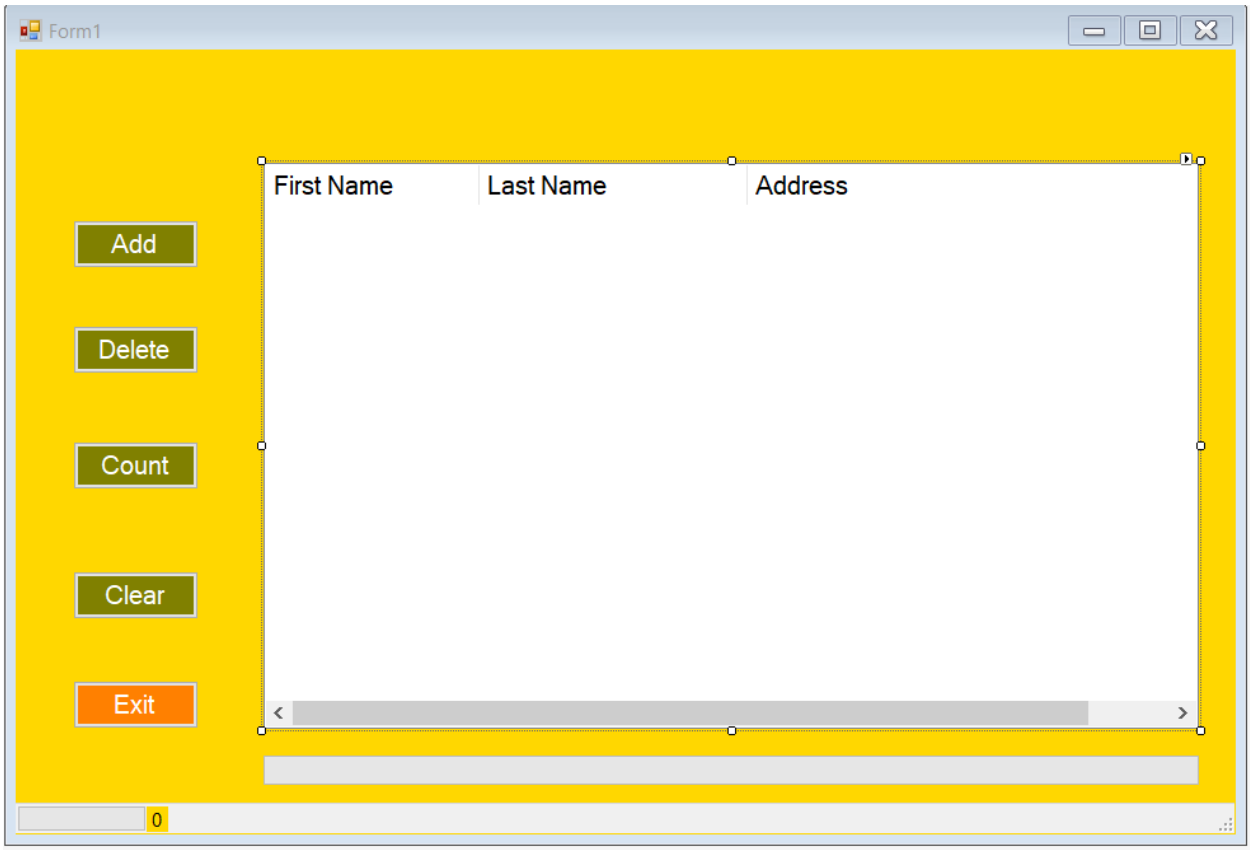
When **confirm order** button clicked **Form2** will be shown showing the order and the total cost

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApplication26
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            double cost = 0;
            Form2 ff = new Form2();
            ff.Show();

            if (radioButtonSmall.Checked==true)
            {
                ff.label1.Text += "Small Pizza";
                cost += 4.00;
            }
             if (radioButtonMedium.Checked==true)
            {
                ff.label1.Text += "medium Pizza";
                cost += 7.00;
            }
             if (radioButtonLarge.Checked == true)
             {
                 ff.label1.Text += "Large Pizza";
                 cost += 10.00;
             }
             if (radioButtonXlarge.Checked == true)
             {
                 ff.label1.Text +=  " Extra Large Pizza";
                 cost += 13.00;
             }


             if (checkpepperoni.Checked == true)
             {
```

```
    ff.label2.Text += "\t" + "   pepperoni "+"\t";
    cost += 0.75;
}

if (checkextracheese.Checked == true)
{
    ff.label2.Text += "\t" + "   Extra Cheese" + "\t";
    cost += 0.75;
}
if (checkmashroom.Checked == true)
{
    ff.label2.Text += "\t" + "   Mashroom" + "\t";
    cost += 0.75;
}
if (checkjalapeno.Checked == true)
{
    ff.label2.Text += "\t" + "   Jalapeno" + "\t";
    cost += 0.75;
}
if (checkanchovies.Checked == true)
{
    ff.label2.Text += "\t" + "   Anchovis" + "\t";
    cost += 0.75;
}
if (checkdriedtomato.Checked == true)
{
    ff.label2.Text += "\t" + "   Sun Dried Tomato" + "\t";
    cost += 0.75;

}
if (checkrostedgarlic.Checked == true)
{
    ff.label2.Text += "\t" + "   Rosted Garlic" + "\t";
    cost += 0.75;
}
if (checkshredchicken.Checked == true)
{
    ff.label2.Text += "\t" + "   Shred Chicken" + "\t";
    cost += 0.75;
}


if (checkBox15.Checked == true)
{
    ff.label3.Text +=  "   Coke" + "\t";
    cost += 1.45;
}
if (checkBox16.Checked == true)
{
    ff.label3.Text +=  "   Diet Coke" + "\t";
    cost += 1.45;
}
if (checkBox17.Checked == true)
{
    ff.label3.Text +=  "   Iced Tea" + "\t";
```

```
                cost += 1.45;

            }
            if (checkBox18.Checked == true)
            {
                ff.label3.Text +=  "   Sprite" + "\t";
                cost += 1.45;
            }
            if (checkBox19.Checked == true)
            {
                ff.label3.Text +=  "   Water" + "\n";
                cost += 1.00;
            }


            if (checkBox22.Checked == true)
            {
                ff.label4.Text +=  "   Chicken Wings" + "\t";
                cost += 3.00;
            }
            if (checkBox23.Checked == true)
            {
                ff.label4.Text +=  "   Pounti" + "\t";
                cost += 3.00;
            }
            if (checkBox24.Checked == true)
            {
                ff.label4.Text += "   Onion Rings" + "\t";
                cost += 3.00;
            }
            if (checkBox25.Checked == true)
            {
                ff.label4.Text +=  "   Cheesy Grlic Bread" + "\t";
                cost += 3.00;
            }
            ff.labelcost.Text = Convert.ToString(cost)+"$";
        }



        private void textBox5_TextChanged(object sender, EventArgs e)
        {
                    }

    }
}
```

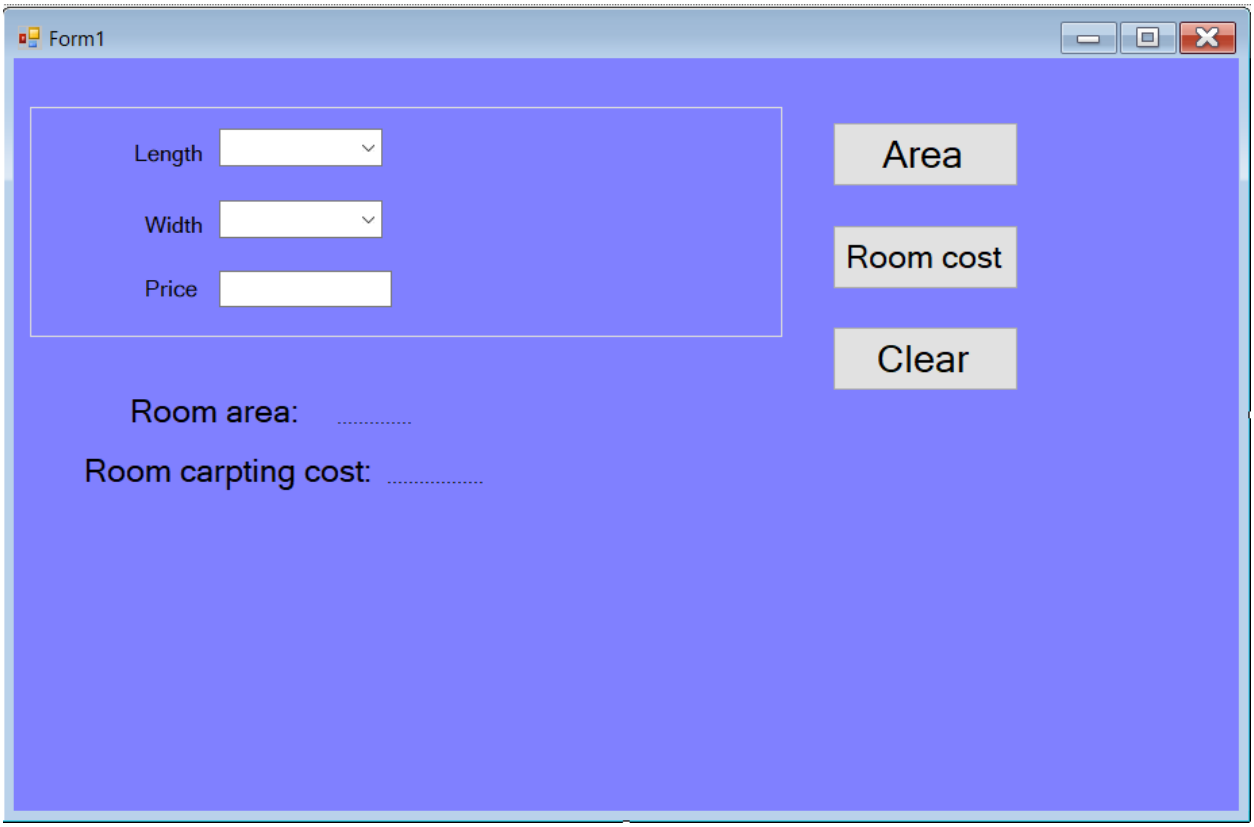
List view, stausbar ,statusstrip




```
using System;
using System.Collections.Generic;
using System.ComponentModel;
```

```
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btn_Add_Click(object sender, EventArgs e){

           ListViewItem  lvitem= new ListViewItem("David");
           lvitem.SubItems.Add(" Beckham");
           lvitem.SubItems.Add(" Manchester");
           listView1.Items.Add(lvitem);

           ListViewItem lvitem2 = new ListViewItem("Rayan");
           lvitem2.SubItems.Add(" Giggs");
           lvitem2.SubItems.Add(" leads");
           listView1.Items.Add(lvitem2);

           ListViewItem lvitem3 = new ListViewItem("poul");
           lvitem3.SubItems.Add(" scols");
           lvitem3.SubItems.Add("London ");
           listView1.Items.Add(lvitem3);

           int i = listView1.Items.Count;
           {
               progressBar1.Value = i;



             // toolStripProgressBar1.Value = i;

             //toolStripStatusLabel1.Text = Convert.ToString(i);
           }

        }

        private void btn_del_Click(object sender, EventArgs e)
        {
            if (listView1.SelectedItems.Count > 0)
            {
                listView1.Items.Remove(listView1.SelectedItems[0]);
                --progressBar1.Value;

               // --toolStripProgressBar1.Value;

            }
            else {
                MessageBox.Show("Please select a record", "alert");
```

```
    }
}

private void btn_count_Click(object sender, EventArgs e)
{
    int count = listView1.Items.Count;
    MessageBox.Show("Total records="+count,"Total");
}

private void btn_clear_Click(object sender, EventArgs e)
{
    listView1.Items.Clear();
    progressBar1.Value = 0;
    toolStripStatusLabel1.Text = "0";


    //toolStripProgressBar1.Value = 0;

}

private void btn_exit_Click(object sender, EventArgs e)
{
    Application.Exit();
}

private void Form1_Load(object sender, EventArgs e)
{
    listView1.GridLines = true;
}
```



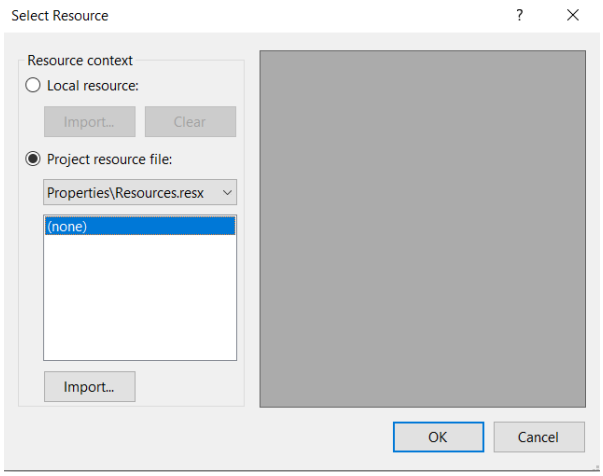

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace carpting
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
```

```
private void Form1_Load(object sender, EventArgs e)
{
    int i, j;
    for (i = 1; i <= 10; ++i)
    {
        combolength.Items.Add(i);
    }
    for (j = 1; j <= 10; ++j)
    {
        combowidth.Items.Add(j);

    }

}

private void btnarea_Click(object sender, EventArgs e)
{
    int area, width, length;
    length = Convert.ToInt32(combolength.Text);
    width = Convert.ToInt32(combowidth.Text);
    area = length * width;
    lblroomarea.Text = Convert.ToString(area) + "m";

}

private void btnroomcost_Click(object sender, EventArgs e)
{
    int width, length;
    int price = 0, roomcost;
    length = Convert.ToInt32(combolength.Text);
    width = Convert.ToInt32(combowidth.Text);
    price = Convert.ToInt32(txtprice.Text);
    roomcost = length * width * price;
    lblcarptingcost.Text = Convert.ToString(roomcost) + "$";

}

private void btnclear_Click(object sender, EventArgs e)
{
    combolength.Text = "";
    combowidth.Text = "";
    txtprice.Text = "";
    lblroomarea.Text = "..... ";
    lblcarptingcost.Text = ".....";

}
```

```
    }
}
```

You Can add image to picture box

1. From properties (image)
2. From the shortcut arrow on the right corner (choose image)

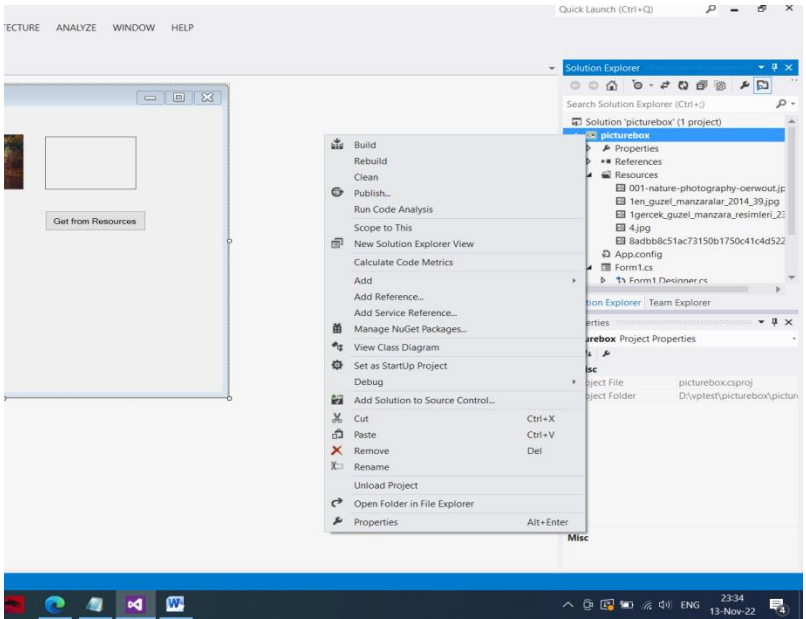


If you made import from local resources this picture you are going to use it just one time

If you choose project resources the picture will be added in to resources folder inside the project and you can use it whenever you want from the project resources

To add image to the picture box by button click we will type inside the button

```
pictur3.Image = Properties.Resources._8adbb8c51ac73150b1750c41c4d5223b;
```



- right click on your project and choose <u>open folder in file explorer</u>

You can see your resources folder there be careful to not delete any image because your project may crash

Go bin-> debug and copy you .exe file and put it on desktop for example run it you can see it will run and view the pictures normally.

Adding image from file

You can copy the image that you want to add inside bin folder in the project when you use it you need to write the name.extention of the file.

Or you can use it by writing the full path of the image

There are many ways to add the picture like:

```
picture4.Image = Image.FromFile("sunset.jpg");
```

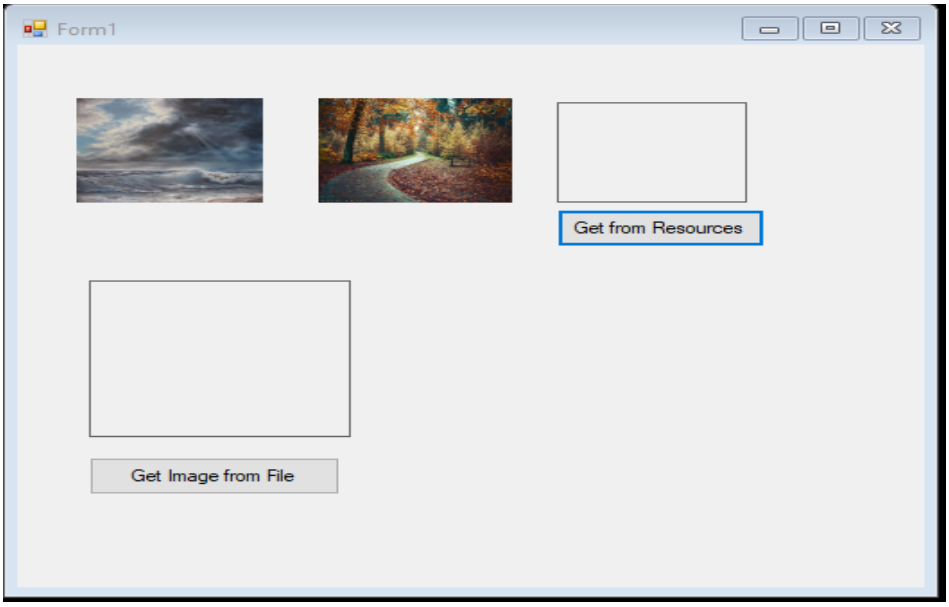
Or using bitmap

```
  Bitmap btp = new Bitmap("sunset.jpg");
            picture4.Image = btp;
```

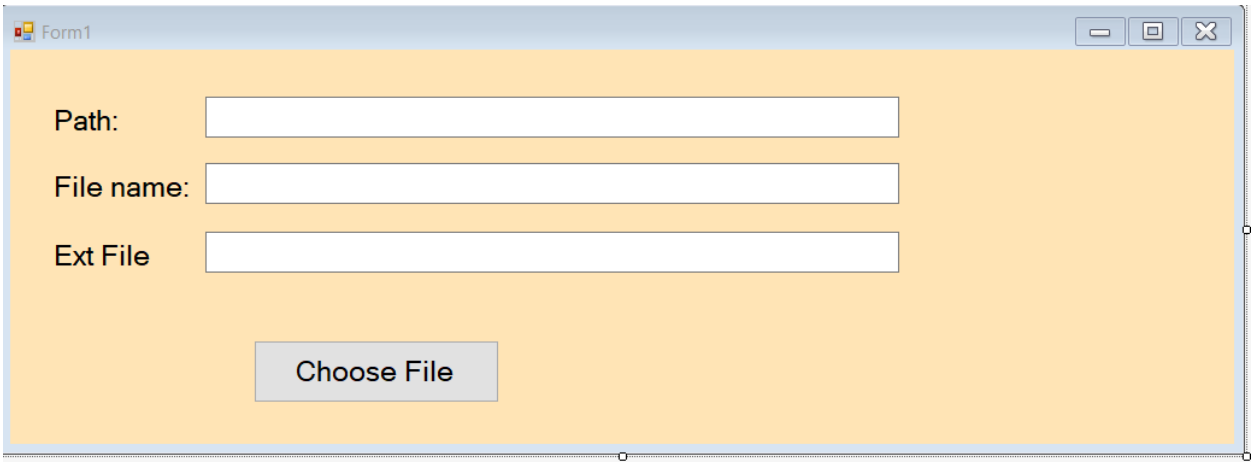
source code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace picturebox
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
```

```
        private void btnGetfromResources_Click(object sender, EventArgs e)
        {
            pictur3.Image = Properties.Resources._8adbb8c51ac73150b1750c41c4d5223b;

        }

        private void btnGetimagefromFile_Click(object sender, EventArgs e)
        {
            //picture4.Image = Image.FromFile("sunset.jpg");
            Bitmap btp = new Bitmap("sunset.jpg");
            picture4.Image = btp;
        }
    }
}
```



Open File Dialog Path File Name extension In C Sharp



Go to the tools from dialog section choose Open file dialog, you can change the name for example ofd

When the button clicked

```
ofd.Title = "Select the File";
this code will give title to the dialig box

ofd.InitialDirectory = "D:\\";

this piece of code makes your initial directory drive D

ofd.ShowDialog();

this will open a dialog box


after opening the dialogbox when we select a file

txtPath.Text = ofd.FileName;
```

this code will bring the path of the file and put it in the txtPath textbox

```
txtFileName.Text = Path.GetFileName(ofd.FileName);
```

this code will take the name of the file from the path and show it in the txtfilename textbox

```
in order to activate the Path

using System.IO; must be declared at the beginning of the program

txtExtFile.Text = Path.GetExtension(ofd.FileName);
```

this code will take the extention of the file from the path and show it in the txtExt textbox

code:

```
using System;
using System.Collections.Generic;
```

```
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;

namespace Open_File_Dialog_Path_File_Name_extension_In_C_Sharp
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            ofd.Title = "Select the File";
            ofd.InitialDirectory = "D:\\";
            ofd.ShowDialog();
            txtPath.Text = ofd.FileName;
            txtFileName.Text = Path.GetFileName(ofd.FileName);
            txtExtFile.Text = Path.GetExtension(ofd.FileName);

        }


    }
}
```

If you want to select more than one file when button clicked

```
        private void button1_Click(object sender, EventArgs e)
```

```
        {
            ofd.Multiselect = true;
            ofd.ShowDialog();
            foreach (string f in ofd.FileNames)
            {
                MessageBox.Show(f);
            }

        }
```

You can open file dialog instead of using the dialogbox control from the tools:

```
private void button1_Click(object sender, EventArgs e)
        {
           /* ofd.Title = "Select the File";
            ofd.InitialDirectory = "D:\\";
            ofd.ShowDialog();
            txtPath.Text = ofd.FileName;
            txtFileName.Text = Path.GetFileName(ofd.FileName);
            txtExtFile.Text = Path.GetExtension(ofd.FileName); */


            /*ofd.Multiselect = true;
            ofd.ShowDialog();
            foreach (string f in ofd.FileNames)
            {
                MessageBox.Show(f);
            }*/


            OpenFileDialog of = new OpenFileDialog();
            of.ShowDialog();

        }
```

If you want to be sure that you opened the file or not

```
private void button1_Click(object sender, EventArgs e)
```

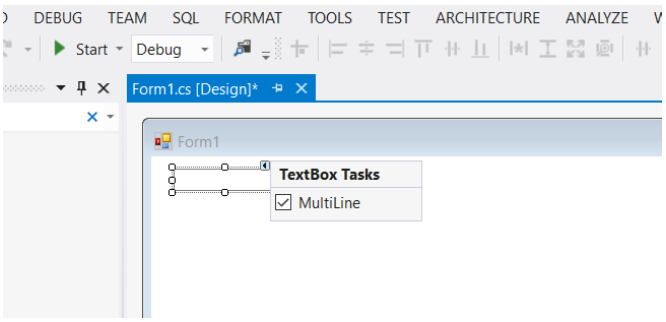
```
{
   /* ofd.Title = "Select the File";
    ofd.InitialDirectory = "D:\\";
    ofd.ShowDialog();
    txtPath.Text = ofd.FileName;
    txtFileName.Text = Path.GetFileName(ofd.FileName);
    txtExtFile.Text = Path.GetExtension(ofd.FileName); */


    /*ofd.Multiselect = true;
    ofd.ShowDialog();
    foreach (string f in ofd.FileNames)
    {
        MessageBox.Show(f);
    }*/


    OpenFileDialog of = new OpenFileDialog();

    if (of.ShowDialog() == DialogResult.OK)
    {
        MessageBox.Show("file Slected");
    }
    else
    {
        MessageBox.Show("!!!! you did not Select A file");
    }



}
```

Save File Dialog Path File Name extension In C Sharp

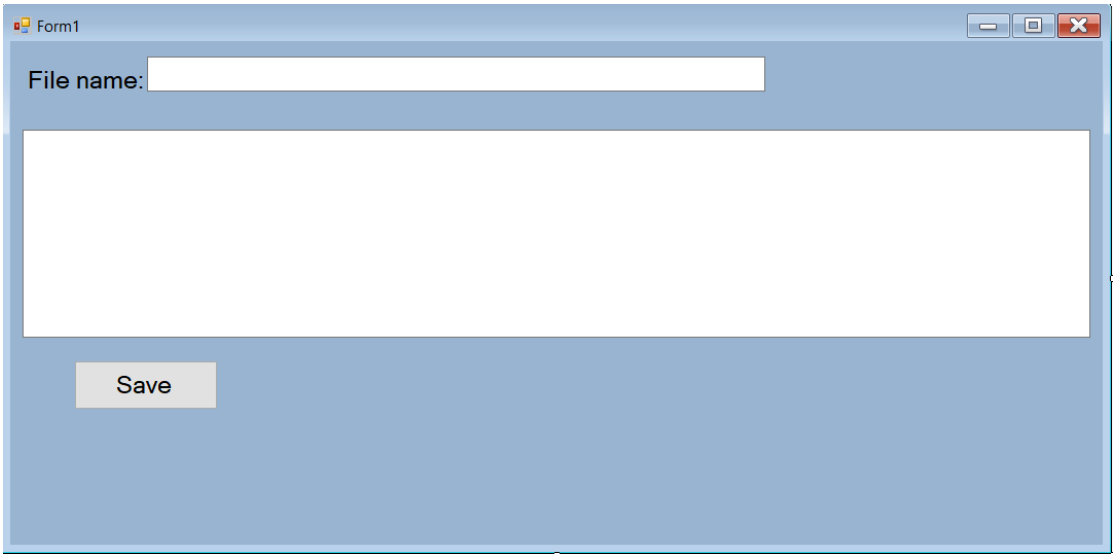
Open new project

Add a label(File name), add textbox name it txtFileName ,

Add textbox name it txtText from the arrow above make it multiline (you can do it from the property window or by codding)



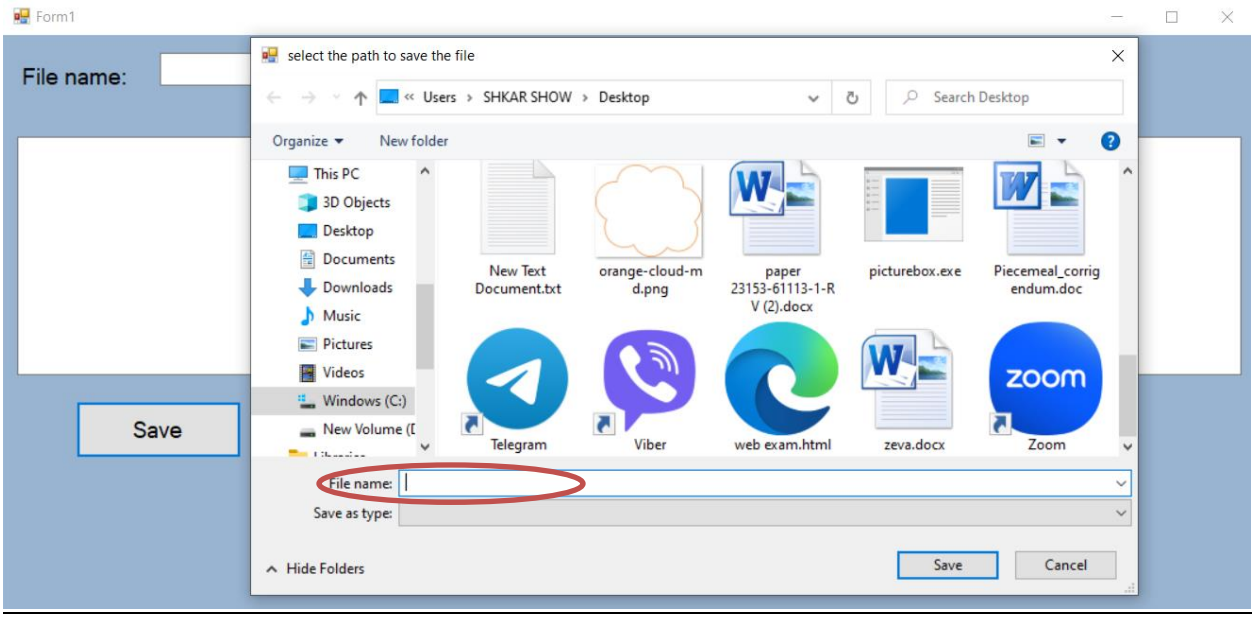
then add button (Save) name it btnSave



Click on save button and add the code:

```
  SaveFileDialog sfd = new SaveFileDialog();

  sfd.Title = " select the path to save the file";// title of the save window

  sfd.InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);//
initailly the path is desktop

  sfd.ShowDialog();// show the dialog
```

after running



Till now our program is just showing the dialog but still need some codes to activate save

The file name is empty as you can see , in order to add any default name before showing the dialog add the code: `sfd.FileName = "untitled.txt";`

```
private void btnSave_Click(object sender, EventArgs e)
        {

   SaveFileDialog sfd = new SaveFileDialog();

  sfd.Title = " select the path to save the file";// title of the save window


sfd.InitialDirectory=Environment.GetFolderPath(Environment.SpecialFolder.Desktop);//
initailly the path is desktop

 sfd.FileName = "untitled.txt";// default name

            sfd.ShowDialog();// show the dialog

        }
```

For more professional programming instead of untitled.txt we can use the txtFileName.text concatenated with .txt

```
  SaveFileDialog sfd = new SaveFileDialog();

  sfd.Title = " select the path to save the file";// title of the save window
```

```
  sfd.InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);//
initailly the path is desktop

 sfd.FileName = txtFileName.Text + ".txt"; // because we can write the file name in
textbox

 sfd.ShowDialog();// show the dialog
```

now let's program how to save a file

first add `using System.IO;` at the top because we will use `Path.`

from the property window we can add a default file name by adding untitled to the text fielf of the textbox (txtFileName)

then check if the savedialog result was Ok

```
if(sfd.ShowDialog()==DialogResult.OK) {

}
```

In order to not change the path or the extention

save the path in string

```
string strPath = sfd.FileName;
```

if the extention is not .txt,
ToLower() if the extention written in capital

```
if (Path.GetExtension(sfd.FileName).ToLower() != ".txt")
{
strPath += ".txt"; // if .txt was not exist it will be added
 }
```

In order to save what written in the textbox in a text file

```
StreamWriter sw = new StreamWriter(strPath); sw.WriteLine(txtText.Text);// any text
written in the textbox(txtText)
sw.Close();
```

Full source code :

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
```

```
using System.IO;

namespace savedialog
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btnSave_Click(object sender, EventArgs e)
        {
            SaveFileDialog sfd = new SaveFileDialog();

            sfd.Title = " select the path to save the file";// title of the save
window

    sfd.InitialDirectory=
Environment.GetFolderPath(Environment.SpecialFolder.Desktop);// initailly the path is
desktop

//sfd.FileName = "untitled.txt";// default name

sfd.FileName = txtFileName.Text + ".txt"; // because we write the file name in
textbox(txtText)


if (sfd.ShowDialog() == DialogResult.OK)
            {
             string strPath = sfd.FileName;// save the path in string

             if (Path.GetExtension(sfd.FileName).ToLower() != ".txt")// if the
extention is not .txt, ToLower()  if the extention written in capital accepted
                {
                    strPath += ".txt";// if .txt was not exist it will be added
                }

                StreamWriter sw = new StreamWriter(strPath);
                sw.WriteLine(txtText.Text);// any text written in the
textbox(txtText)
                sw.Close();
            }

        }
    }
}
```

Design the following form:

Add tooltip from the toolbox

And add a tooltip for each control after clicking the control such as textbox you can add the looltip for it from the property window.

Adding data when Add button clicked

```
private void btn_Add_Click(object sender, EventArgs e)
        {
            StreamWriter sw = new StreamWriter("Data.txt" , true);
            string strEmployee = txt_ID.Text + ";"
                                 + txt_Name.Text + ";"
                                 + txt_Adress.Text;
            sw.WriteLine(strEmployee);
            sw.Close();
            MessageBox.Show(" employee data added");
            foreach(Control c in this.Controls){
                if (c is TextBox)
                    c.Text="";
            }
            txt_ID.Focus();
        }
    }
```

Right Click on the project and choose open folder in file explorer → bin → debug you will see a text file where the data saved.

In this case if you write the same ID it will accept in order to prevent the duplication of the ID we have to use the stream reader before adding the Data.

So the code of the Add button will be

```
private void btn_Add_Click(object sender, EventArgs e)
        {

            StreamReader srCheck = new StreamReader("Data.txt");
            string strCheck = srCheck.ReadToEnd();
            srCheck.Close();
            if (strCheck.Contains(txt_ID.Text + ";"))
            {
                MessageBox.Show("This ID is already exist please insert invalid ID");
                txt_ID.Focus();
                txt_ID.SelectAll();
            }

            else
            {

                StreamWriter sw = new StreamWriter("Data.txt", true);
                string strEmployee = txt_ID.Text + ";"
                                   + txt_Name.Text + ";"
                                   + txt_Adress.Text;
                sw.WriteLine(strEmployee);
                sw.Close();
                MessageBox.Show(" employee data added");
                foreach (Control c in this.Controls)
                {
                    if (c is TextBox)
                        c.Text = "";
                }
                txt_ID.Focus();
            }
        }
    }
```

If you want the user to fill all the data there is a simple code can be added at the beginning in order to not let any textbox empty

```
private void btn_Add_Click(object sender, EventArgs e)
        {
            if (txt_ID.Text.Trim()=="" || txt_Name.Text.Trim()==""
||txt_Address.Text.Trim()=="")
            {
                MessageBox.Show("please fill All Data");
                return; // this will stop the code here will not go to the next code
            }

            StreamReader srCheck = new StreamReader("Data.txt");
            string strCheck = srCheck.ReadToEnd();
            srCheck.Close();
            if (strCheck.Contains(txt_ID.Text + ";"))
            {
                MessageBox.Show("This ID is already exist please insert invalid ID");
                txt_ID.Focus();
                txt_ID.SelectAll();
            }
```

```
            else
            {

                StreamWriter sw = new StreamWriter("Data.txt", true);
                string strEmployee = txt_ID.Text + ";"
                                   + txt_Name.Text + ";"
                                   + txt_Address.Text;
                sw.WriteLine(strEmployee);
                sw.Close();
                MessageBox.Show(" employee data added");
                foreach (Control c in this.Controls)
                {
                    if (c is TextBox)
                        c.Text = "";
                }
                txt_ID.Focus();
            }
        }
    }
```

Also you can add all the code in try catch to prevent any error my cause by opening file or changing data etc.

```
   try
            {
                if (txt_ID.Text.Trim() == "" || txt_Name.Text.Trim() == "" ||
txt_Address.Text.Trim() == "")
                {
                    MessageBox.Show("please fill All Data");
                    return;
                }

                StreamReader srCheck = new StreamReader("Data.txt");
                string strCheck = srCheck.ReadToEnd();
                srCheck.Close();
                if (strCheck.Contains(txt_ID.Text + ";"))
                {
                    MessageBox.Show("This ID is already exist please insert invalid ID");
                    txt_ID.Focus();
                    txt_ID.SelectAll();
                }

                else
                {

                    StreamWriter sw = new StreamWriter("Data.txt", true);
                    string strEmployee = txt_ID.Text + ";"
                                       + txt_Name.Text + ";"
                                       + txt_Address.Text;
                    sw.WriteLine(strEmployee);
                    sw.Close();
                    MessageBox.Show(" employee data added");
                    foreach (Control c in this.Controls)
                    {
```

```
                    if (c is TextBox)
                        c.Text = "";
                }
                txt_ID.Focus();
            }
        }

        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
}
```

Find button:

Find button is used to find the employee according to the ID

So the first thing we do we must check the ID textbox which is named txt_ID is not empty

```
if (txt_ID.Text.Trim() != "")
            {


            }

            else {
                MessageBox.Show(" Please enter Id and try again");
                txt_ID.Focus();
            }
```

If the txt_ID was not empty we need a streamreader

```
StreamReader sr = new StreamReader("Data.txt");
```

Then in this program we will declare a string line store the data for each loop

And a variable with the type bool named found initially set to false

```
string line = "";  // this will store the line in each loop
bool found = false;// initially is false
```

after that we will work with do while

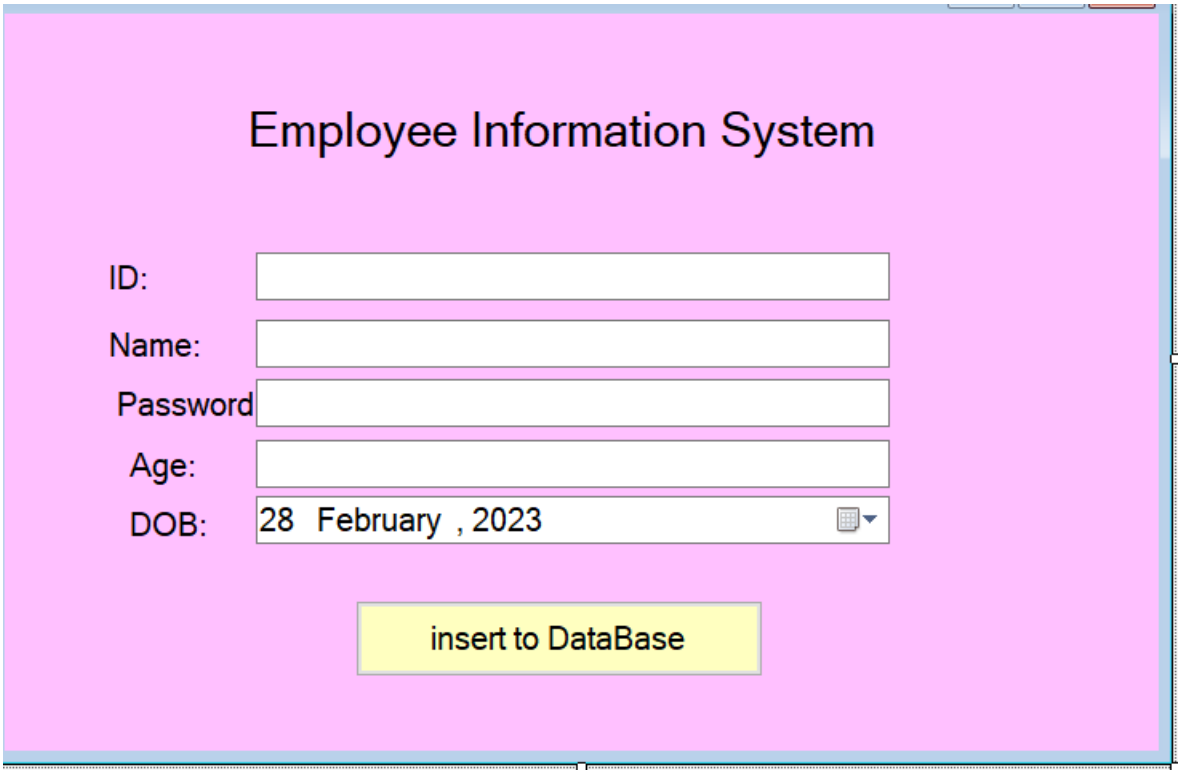
```
private void btn_find_Click(object sender, EventArgs e)
        {
            if (txt_ID.Text.Trim() != "")
            {
                StreamReader sr = new StreamReader("Data.txt");
                string line = "";  // this will store the line in each loop
                bool found = false;// initially is false
                do
                {
                    line = sr.ReadLine(); // for each loop

                    if (line != null)
                    {
                        string[] arrData = line.Split(';'); //because in our text file we
saparated the ID; Name; Address with ;

                        if(arrData[0]==txt_ID.Text){  // if the ID is found in the first
part [0] after split
                            txt_ID.Text = arrData[0];  // show the ID
                            txt_Name.Text = arrData[1]; // show the name
                            txt_Address.Text = arrData[2]; // show the address
                            found = true;

                        }
                    }
```

```
            }

            while (line != null);
            sr.Close();
            if (!found)
            {
                MessageBox.Show(" this ID not found!");
                txt_ID.Focus();
                txt_ID.SelectAll();
            }
        }

        else {
            MessageBox.Show(" Please enter Id and try again");
            txt_ID.Focus();
        }
    }
```

Show All button

```
    private void btn_Showall_Click(object sender, EventArgs e)
    {
        Form frmShow = new Form();
        TextBox txtShow = new TextBox();

        frmShow.StartPosition = FormStartPosition.CenterScreen;
        frmShow.Font = this.Font;
        frmShow.Icon = this.Icon;
        frmShow.Size = this.Size;
        frmShow.Text = "All Data";

        txtShow.Multiline = true;
        txtShow.Dock = DockStyle.Fill; // textbox fill the screen
        frmShow.Controls.Add(txtShow);

        try {
            StreamReader sr = new StreamReader("Data.txt");
            string strAll = sr.ReadToEnd();
            sr.Close();
            txtShow.Text = strAll;
        }
        catch(Exception ex) {
            MessageBox.Show(" ex.message");
        }

        frmShow.ShowDialog();
    }
```

Inserting data to database without duplicating the ID




```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.OleDb;

namespace dataconnection
{
    public partial class Form1 : Form
    {
        OleDbConnection connection = new OleDbConnection();
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
```

```
            connection.ConnectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=D:\cihan\cihan11\VP\2nd semester\2nd semester\lab 2022-2023\4
login\Database2.accdb";


        }

        private void button1_Click(object sender, EventArgs e)
        {

        }

        private void btn_insert_Click(object sender, EventArgs e)
        {
            connection.Open();
            OleDbCommand com = new OleDbCommand();
            com.Connection = connection;
            com.CommandText = " select * from employee where ID='" + txt_ID.Text + "' ";
            OleDbDataReader reader = com.ExecuteReader();
            if (reader.Read() == true)
            {

                MessageBox.Show("ID is deplicated");
                txt_ID.SelectAll();
              //  txt_ID.Text = "";
                txt_ID.Focus();
                connection.Close();
            }

            else
            {
                reader.Close();


                String cb = "insert into employee (ID,Employee_name,
Employee_password,age,DOB) VALUES (@p1, @p2,@p3,@p4,@5)";
                OleDbCommand command = new OleDbCommand(cb, connection);
                command.CommandType = CommandType.Text;
                command.Parameters.AddWithValue("@p1", txt_ID.Text);
                command.Parameters.AddWithValue("@p2", txt_name.Text);
                command.Parameters.AddWithValue("@p3", txt_Password.Text);
                command.Parameters.AddWithValue("@p4", txt_age.Text);
                command.Parameters.AddWithValue("@p5", dateTimePicker1.Text.ToString());


                command.ExecuteNonQuery();

                connection.Close();

            }
        }
    }
}
```