Chapter 5

The .NET framework

Chapter 5
The .NET framework

What Are Namespaces?

Namespaces are groups of classes, structures, interfaces, enumerations, and delegates, organized in a logical hierarchy by function. Namespaces are grouped by functionality. Namespaces can be nested.

Finding the Information You Need About Namespaces

common namespaces in the FCL and the functionality they offer:

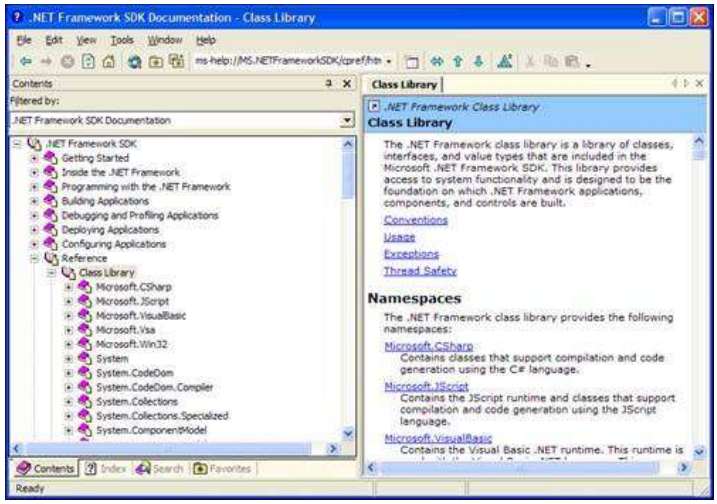
Namespace	**functionality**
System.Data	contains the core data access functionality for SQL Server and OLE DB data stores. Used in ADO.NET.
System.XML	The class namespace core XML processing
System. DirectoryServices	gives you access to all Active Directory objects.
System.Messaging	exposes Message Queue functionality
System. Globalization	ability to write internationally aware applications.
System.Net	sending and receiving data over the network wire.
System.Collections	support for arrays, lists, dictionaries, and hash tables.
System.IO	supports stream, file, and directory access.
System.Text	character encoding, and string manipulation
System.Text. RegularExpressions	contains full regular expression syntax support.
System.Threading	provides multithreading support.
System.Reflection	gives you access to assembly type information.

System.Drawing	gives you access to all the GDI+ drawing
System.Windows. Forms	enables you to write Windows Forms applications
System.Runtime. InteropServices	supports access to COM types.
System.Runtime. Serialization	supports serialization and deserialization of objects.
System.Web	functionality in ASP.NET, Web Forms…
System.Web.Services	creation of SOAP-based XML Web services

Searching the .NET Framework SDK

open the help file for Visual Studio .NET. You should see tabs across the bottom left of the help screen that say **Contents, Index, Search,** and **Favorites**.

You can go to **Reference** section of the .NET Framework class library in the SDK



Using Namespaces in Your Applications

Use the **Imports** statement in VB.NET and **using** statement in C# to avoid using fully qualified names.

➔ you can't use **Imports** or **using** statement on a namespace or class without it first being referenced in the project.

✍ If you created a C# application, you'll see the following items:

```
using System;                    using System.Drawing;
using System.Collections;        using System.ComponentModel;
using System.Data;               using System.Windows.Forms;
```

✍ In VB.NET application, you'll see no Imports statements at the top of your class. The defaults are kept on a per-project basis. You can right-click the project name and select Properties, select the **Imports** node under Common Properties node.

Example


```
Dim m as MessageQueue = New MessageQueue()
```

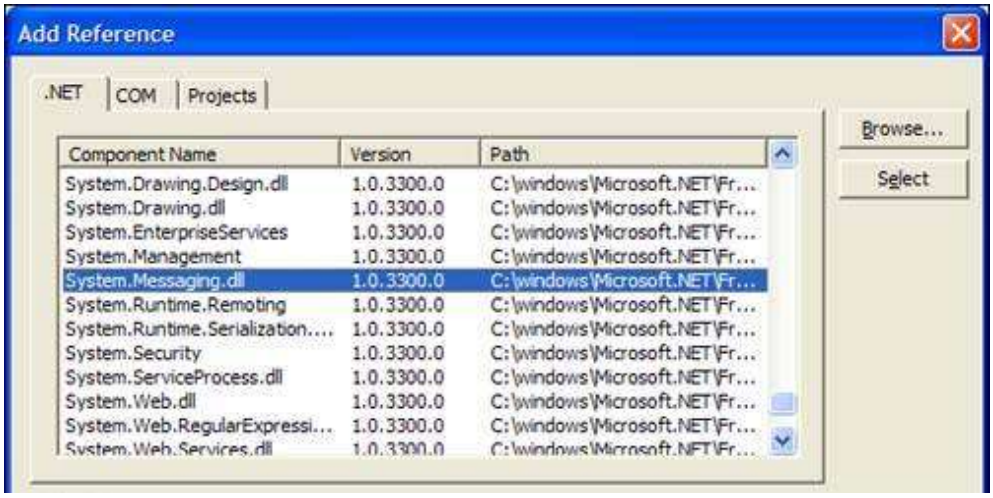

```
MessageQueue m = new MessageQueue();
```

Notice that the MessageQueue object didn't appear in the auto-list members options when you were typing . To make this work, you must do two things:

- Add a reference to the System.Messaging namespace to your application
- Reference the System.Messaging namespace using an **Imports** or **using** statement at the top of your class file

To add a reference to your application, do the following:

1. Right-click the References node of your project in the Solution Explorer.
2. Select Add Reference from.



3. Scroll down on the .NET tab until you see System.Messaging.Dll in the list. Select it and click the Select button. Click the OK button.
4. In your code, alias the System.Messaging namespace with the **Imports** or **using**.

Working with Your Environment

The properties of the Environment class

Property	***Description*:**
CommandLine	Gets the command line for this process.
CurrentDirectory	Gets and sets the fully qualified path of the current directory; that is, the directory from which this process starts.
ExitCode	Gets or sets the exit code of the process.
HasShutdownStarted	Indicates whether the common language runtime is shutting down.
MachineName	Gets the NetBIOS name of this local computer.
NewLine	Gets the newline string defined for this environment.

Property	***Description*:**
OSVersion	Gets an OperatingSystem object that contains the current platform identifier and version number.
StackTrace	Gets current stack trace information.
SystemDirectory	Gets the fully qualified path of the system directory.
TickCount	Gets the number of milliseconds elapsed since the system started.
UserDomainName	Gets the network domain name associated with the current user.
UserInteractive	Gets a value indicating whether the current process is running in user interactive mode.
UserName	Gets the username of the person who started the current thread.
Version	Gets a Version object that describes the major, minor version.

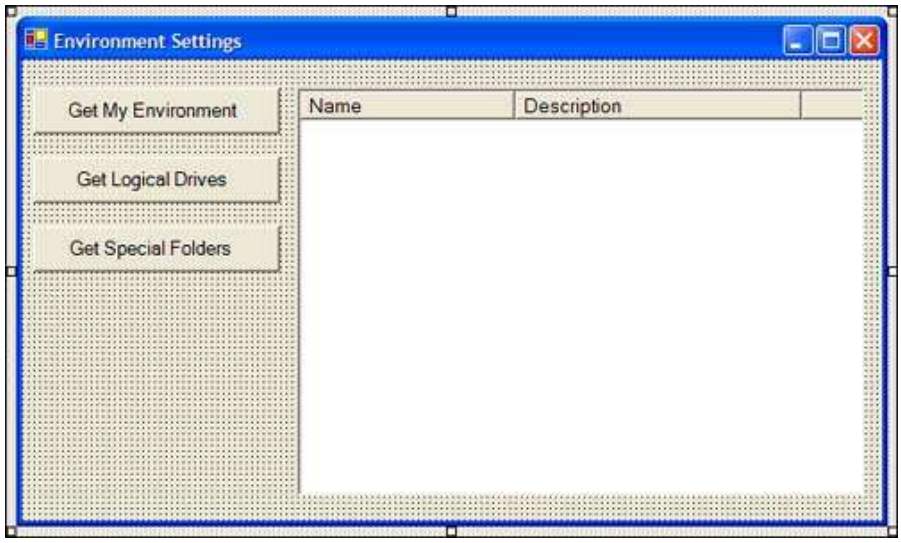
The Methods in the System.Environment Class

Method	***Description***
Exit	Terminates this process and gives the underlying operating system the specified exit code.
GetEnvironmentVariable	Returns the value of the specified environment variable.
GetFolderPath	Gets the path to the system special folder identified by the specified enumeration.

Method	***Description***
GetLogicalDrives	Returns an array of string containing the names of the logical drives on the current computer.

Creating the Environment Sample Application

1. Create a new Windows Forms application named Environment_Winforms.
2. On default form1, change the following properties:
    - Name = mainForm
    - Text = "Environment Settings"
    - StartupPosition = CenterScreen
3. Next, drag a button onto the form, and change these properties:
    - Name = GetEnvironment
    - Text = "Get My Environment"
4. Drag another button onto the form, and change these properties:
    - Name = GetDrives
    - Text = "Get Logical Drives"
5. Drag another button onto the form, and change these properties:
    - Name = GetSpecialFolders
    - Text = "Get Special Folders"
6. Drag a ListView object onto the form, and change these properties:
    - Name = EnvironmentListView
    - Anchor = Top, Left, Bottom, Right
    - Columns = Click the **ellipses** button. Click the **Add** button twice, and change the **Text** property for the first column to "Name" and the second column to "Description". Click **OK** to close the custom designer.
    - View = Details (else you won't see the column headers.)



GetEnvironment_Click Event Code

VB.NET

```
Private Sub GetEnvironment_Click (
   ByVal sender As System.Object, _
   ByVal e As System.EventArgs) _
   Handles GetEnvironment.Click

 ' Set the cursor to an hourglass
 Me.Cursor = Cursors.WaitCursor

 With EnvironmentListView
  ' Clear the ListView of any items
  .Items.Clear()

  ' Create a ListItem variable
  Dim lvi As ListViewItem

  ' Create a new instance of the ListItem
  're-use it for properties added to ListView
  lvi = New ListViewItem()
```

```
With lvi
   .Text = "Current Directory"
.SubItems.Add(Envronment.CurentDirectory)
  End With
  .Items.Add(lvi)
  lvi = New ListViewItem()
  With lvi
   .Text = "Machine Name"
 .SubItems.Add(Environment.MachineName)
  End With
  .Items.Add(lvi)

  lvi = New ListViewItem()
  With lvi
   .Text = "OS Version"
   'Call ToString method to convert
   ' the Number  to a string
.SubItems.Add( _
           Environment.OSVersion.ToString)
  End With
  .Items.Add(lvi)
  lvi = New ListViewItem()
  With lvi
   .Text = "System Directory"
.SubItems.Add(Envronmnt.SystemDirectory)
  End With
  .Items.Add(lvi)
  lvi = New ListViewItem()
  With lvi
   .Text = "User Name"
   .SubItems.Add(Environment.UserName)
  End With
  .Items.Add(lvi)
End With
 ' Set the cursor back to normal
 Me.Cursor = Cursors.Default
End Sub
```

C#

```
private void getEnvironment_Click(object sender,
System.EventArgs e)
 {

  // Set the cursor to an hourglass
  this.Cursor = Cursors.WaitCursor;


  // Clear the ListView of any items
  EnvironmentListView.Items.Clear();

  // Create a ListItem variable
  ListViewItem lvi;

  // Create a new instance of the ListItem
  // reuse it for properties added to ListView
  lvi = new ListViewItem();

  lvi.Text = "Current Directory";
  lvi.SubItems.Add( Environment.CurrentDirectory);
  EnvironmentListView.Items.Add(lvi);

  lvi = new ListViewItem();

  lvi.Text = "Machine Name";
  lvi.SubItems.Add( Environment.MachineName);
  EnvironmentListView.Items.Add(lvi);

  lvi = new ListViewItem();

  lvi.Text = "OS Version";
  //Call ToString method to convert
   // the Number  to a string
  lvi.SubItems.Add( Environment.OSVersion.ToString());
```

```
 EnvironmentListView.Items.Add(lvi);

  lvi = new ListViewItem();

  lvi.Text = "System Directory";
  lvi.SubItems.Add(
          Environment.SystemDirectory);
  EnvironmentListView.Items.Add(lvi);

  lvi = new ListViewItem();

  lvi.Text = "User Name";
  lvi.SubItems.Add(Environment.UserName);

  EnvironmentListView.Items.Add(lvi);


  // Set the cursor back to normal
  this.Cursor = Cursors.Default;
  }
 }
```

Getting the Logical Drives

VB.NET

```
Private Sub GetDrives_Click(  _
  ByVal sender As System.Object, _
  ByVal e As System.EventArgs) _
  Handles GetDrives.Click

     Me.Cursor = Cursors.WaitCursor
     EnvironmentListView.Items.Clear()
 Dim logicalDrives As String() =  _
        Environment.GetLogicalDrives()

Dim lvi As ListViewItem
Dim drive As String
```

```
For Each drive In logicalDrives
  With EnvironmentListView
   lvi = New ListViewItem()
   With lvi
    .Text = "Drive Letter"
    .SubItems.Add(drive)
   End With
   .Items.Add(lvi)
  End With
 Next drive

 Me.Cursor = Cursors.Default
End Sub
```

C#

```
private void getDrives_Click(
      object sender, System.EventArgs e)
 {


  this.Cursor = Cursors.WaitCursor;
  EnvironmentListView.Items.Clear();
  string[] logicalDrives =
        Environment.GetLogicalDrives();

  ListViewItem lvi;
  foreach(string drives in logicalDrives)
  {
     lvi = new ListViewItem();

     lvi.Text = "Drive Letter";
     lvi.SubItems.Add(drives);
     EnvironmentListView.Items.Add(lvi);
  }
  this.Cursor = Cursors.Default;
 }
```

Getting Special System Folders with the SpecialFolder Enumeration

The SpecialFolder enumeration contains the path to the following directories:

ApplicationData
Cookies
CommonApplicationData
ProgramFiles
DesktopDirectory
Favorites
CommonProgramFile
Programs
InternetCache
History
LocalApplicationData
StartMenu
Recent
System
Templates
SendTo
Startup

In .NET, figuring enumeration values is very easy, thanks to the auto list members feature,
Especially in VB.NET.

VB.NET

```
Private Sub GetSpecialFolders_Click(ByVal sender As
System.Object, _
ByVal e As System.EventArgs) Handles GetSpecialFolders.Click

Me.Cursor = Cursors.WaitCursor

With EnvironmentListView.Items
.Clear()
.Add("Program Files Folder").SubItems.Add
(Environment.GetFolderPath _
```

```
(Environment.SpecialFolder.ProgramFiles))

.Add("Application Data").SubItems.Add
(Environment.GetFolderPath _
(Environment.SpecialFolder.ApplicationData))

.Add("Personal Folder").SubItems.Add
(Environment.GetFolderPath _
(Environment.SpecialFolder.Personal))

.Add("Local Application Data Folder").SubItems.Add
(Environment.GetFolderPath _
(Environment.SpecialFolder.ApplicationData))

End With
Me.Cursor = Cursors.Default
End Sub
```

C#

```
private void getSpecialFolders_Click(object sender,
System.EventArgs e)
{
this.Cursor = Cursors.WaitCursor;

EnvironmentListView.Items.Clear();

EnvironmentListView.Items.Add("Program Files").SubItems.Add
(Environment.GetFolderPath
(Environment.SpecialFolder.ProgramFiles));

EnvironmentListView.Items.Add("AplicatonData").SubItems.Add
(Environment.GetFolderPath
(Environment.SpecialFolder.ApplicationData));

EnvironmentListView.Items.Add("MyDocuments").SubItems.Add
(Environment.GetFolderPath
(Environment.SpecialFolder.Personal));
```

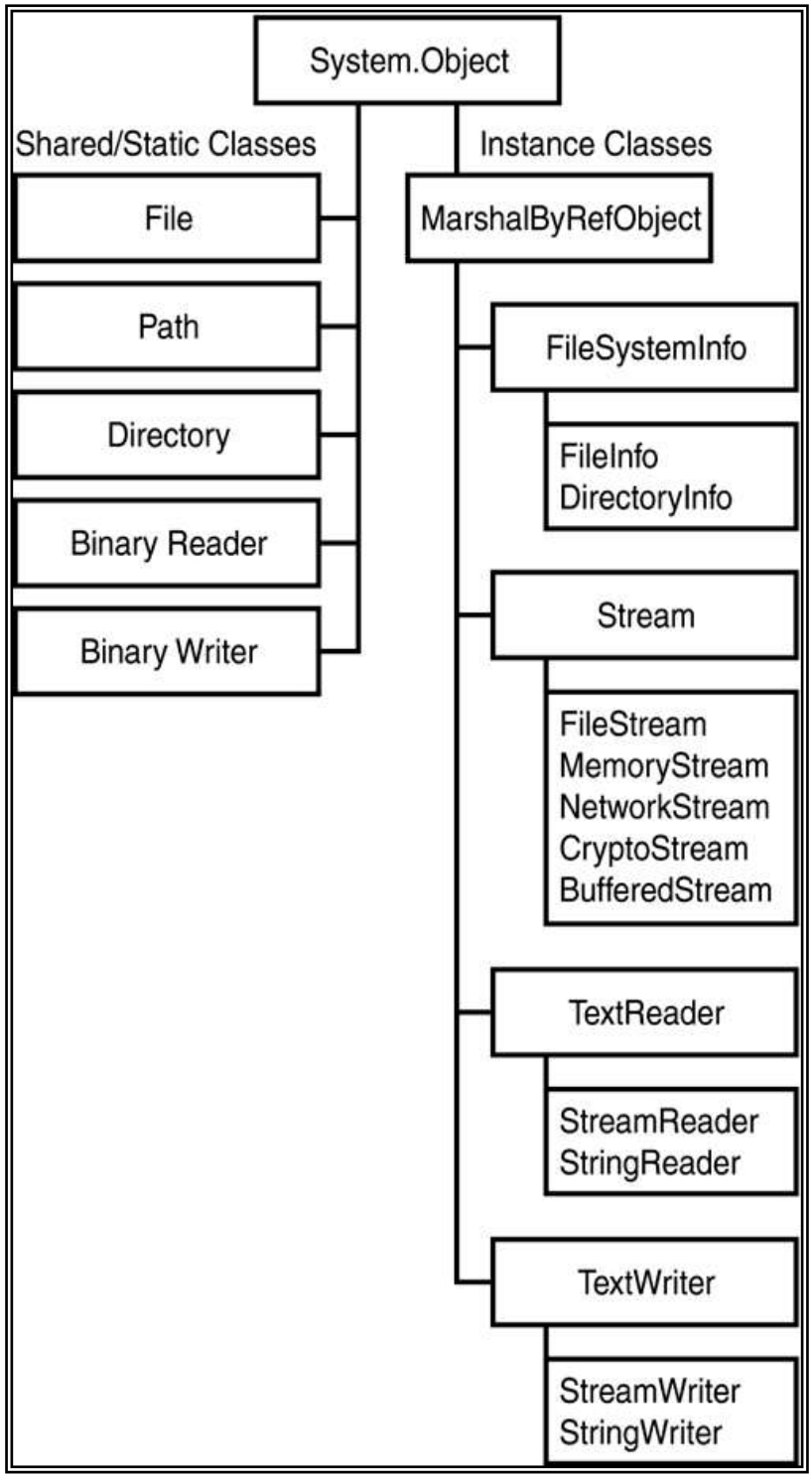
```
EnvironmentListView.Items.Add("Local Application
Data").SubItems.Add
(Environment.GetFolderPath
(Environment.SpecialFolder.LocalApplicationData));

this.Cursor = Cursors.Default;

}
```

f. Inside the System.IO Namespace

The System.IO namespace contains **static (shared)** classes and instance classes that enable you to interact with files, paths, directories, and streams. Streams can be any type of resource, such as a file on the file system, a network stream, or a URL from Internet.


System.Object
Shared/Static Classes
Instance Classes
File
Path
Directory
Binary Reader
Binary Writer
MarshalByRefObject
FileSystemInfo
FileInfo
DirectoryInfo
Stream
FileStream
MemoryStream
NetworkStream
CryptoStream
BufferedStream
TextReader
StreamReader
StringReader
TextWriter
StreamWriter
StringWriter

1. File Class

The File class provides static methods that give you access to files. The instance class FileInfo gives you functionality similar to the static File class. But because the File class is static, it performs security checks each time you access a file object. If you're going to access the same file objects more than once, you should use the FileInfo class to avoid the overhead of performing security checks on the same objects each time.

Methods of the File Class

Method	***Description***
AppendText	Creates StreamWriter that appends UTF-8–encoded text to an existing file.
Copy	Copies an existing file to a new file.
Create	Creates a file in the specified fully qualified path.
CreateText	Creates or opens a new file for writing UTF-8–encoded text.
Delete	Deletes the file specified by the fully qualified path. An exception isn't thrown if the specified file doesn't exist.
Exists	Determines whether the specified file exists.
GetAttributes	Gets the FileAttributes of the file on the fully qualified path.
GetCreationTime	Returns the creation date and time of the specified file or directory.
GetLastAccessTime	Returns the date and time the specified file or directory was last accessed.
GetLastWriteTime	Returns the date and time the specified file or directory was last written to.

Methods of the File Class

Method	*Description*
Move	Moves file to new location, allowing to specify a new file name.
Open	Overloaded. Opens a FileStream on the specified path.
OpenRead	Opens an existing file for reading.
OpenText	Opens an existing UTF-8–encoded text file for reading.
OpenWrite	Opens an existing file for writing.
SetAttributes	Sets the specified FileAttributes of the file on the specified path.
SetCreationTime	Sets the date and time the file was created.
SetLastAccessTime	Sets the date and time the specified file was last accessed.
SetLastWriteTime	Sets the date and time that the specified file was last written to.

VB.NET

```
Imports System.IO
Public Class Class1
Sub FileClassStuff()
 ' using Exists, Move, SetCreationTime,  SetLastAccessTime, SetLastWriteTime
  If File.Exists("C:\Junk.txt") Then
   File.Move("C:\Junk.txt", "D:\Junk.txt")
   File.SetCreationTime("D:\Junk.txt", Date.Now)
   File.SetLastAccessTime("D:\Junk.txt", Date.Now)
   File.SetLastWriteTime("D:\Junk.txt", Date.Now)
  End If
```

```
' using Exists and Delete methods
  If File.Exists("C:\Junk.txt") Then  File.Delete("C:\Junk.txt")
End Sub
End Class
```

C#

```
using System;
using System.IO;
namespace IO_FileClass_CS
{
 public class Class1
 {
  static void Main()
  {
 if (File.Exists(@"C:\Junk.txt"))
  {
   File.Move(@"C:\Junk.txt", @"D:\Junk.txt");
   File.SetCreationTime(@"D:\Junk.txt", Date.Now);
   File.SetLastAccessTime(@"D:\Junk.txt", Date.Now);
   File.SetLastWriteTime(@"D:\Junk.txt", Date.Now);
  }
  // using Exists and Delete methods
  if (File.Exists(@"C:\Junk.txt"))
File.Delete(@"C:\Junk.txt");
  }
 }
}
```

2.Path Class

The Path class is also a static class.

Method	***Description***
ChangeExtension	Changes the extension of a path string.
Combine	Combines two path strings.
GetDirectoryName	Returns the directory information for the specified path string.
GetExtension	Returns the extension of the specified path string.
GetFileName	Returns the filename and extension of the specified path string.
GetFileNameWithoutExtension	Returns the filename of the path string without the extension.
GetFullPath	Returns the absolute path for the specified path string.
GetPathRoot	Gets the root directory information of the specified path.
GetTempFileName	Returns a unique temporary filename and creates a zero-byte file by that name on disk.
GetTempPath	Returns the path of the current system's temporary folder.
HasExtension	Determines whether a path includes a filename extension.
IsPathRooted	Gets a value indicating whether the specified path string contains absolute or relative path information.

3. Directory Class

The Directory class exposes methods that enable you to create, delete, move, rename, and enumerate directories and subdirectories. The Directory class is also a static class.

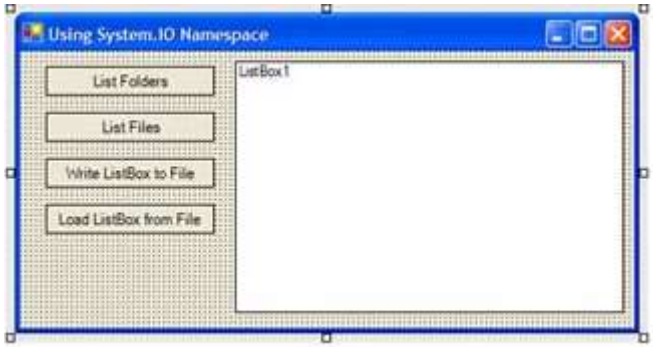
Method	***Description***
CreateDirectory	Creates all directories and subdirectories as specified by path.
Delete	Overloaded. Deletes a directory and its contents.
Exists	Determines whether the given path refers to an existing directory on disk.
GetCreationTime	Gets the creation date and time of a directory.
GetCurrentDirectory	Gets the current working directory of the application.
GetDirectories	Overloaded. Gets the names of subdirectories in the specified directory.
GetDirectoryRoot	Returns the volume information, root information, or both for the specified path.
GetFiles	Returns the names of files in the specified directory.
GetFileSystemEntries	Overloaded. Returns the names of all files and subdirectories in the specified directory.
GetLastAccessTime	Returns the date and time the specified file or directory was last accessed.
GetLastWriteTime	Returns date and time the specified file or directory was last written to.
GetLogicalDrives	Retrieves the names of the logical drives on this computer in the form "<*drive letter*>:\".

Method	***Description***
GetParent	Retrieves the parent directory of the specified path, including both absolute and relative paths.
Move	Moves a file or a directory and its contents to a new location.
SetCreationTime	Sets the creation date and time for the specified file or directory.
SetCurrentDirectory	Sets the application's current working directory to the specified directory.
SetLastAccessTime	Sets the date and time the specified file or directory was last accessed.
SetLastWriteTime	Sets the date and time a directory was last written to.

Creating an I/O Sample Application

1. Create a new Windows Forms application named IO_VB or IO_CS.
2. On the default form1, change the following properties:
    - Text = "Using the System.IO Namespace"
    - StartupPosition = CenterScreen
3. Next, drag a button onto the form, and change these properties:
    - Name = ListFolders
    - Text = "List Folders"
4. Drag another button onto the form, and change these properties:
    - Name = ListFiles
    - Text = "List Files"
5. Drag another button onto the form, and change these properties:
    - Name = WriteToFile
        - Text = "Write To ListBox File"

6. Drag another button onto the form, and change these properties:
    - Name = ReadFromFile
    - Text = "Read ListBox From File"
7. Drag a ListBox from the Toolbox onto the form, and change these properties:
    - Anchor = Top, Bottom, Left, Right



ListFolders_Click Event Code

VB.NET

```
Private Sub ListFolders_Click(ByVal sender As System.Object, _
   ByVal e As System.EventArgs) Handles ListFolders.Click

 ' Create a string variable array named  dirs to hold the directory entries for the
 ' special folder Program Files
 Dim dirs As String() = Directory.GetDirectories (Environment.GetFolderPath _
         (Environment.SpecialFolder.ProgramFiles))

Dim dir As String
For Each dir In dirs
  With ListBox1.Items
   .Add(dir)
  End With
 Next
End Sub
```

C#

```
private void listFolders_Click(object sender, System.EventArgs e)
 {
// Create a string variable array named  dirs to hold the directory entries for the
// special folder Program Files
string[] dirs = Directory.GetDirectories (Environment.GetFolderPath
        (Environment.SpecialFolder.ProgramFiles));

  foreach (string dir in dirs)
        listBox1.Items.Add(dir);
 }
```

Using the GetFileSystemEntries Method

VB.NET

```
Private Sub ListFiles_Click(ByVal sender As System.Object, _
   ByVal e As System.EventArgs) Handles ListFiles.Click

 ListBox1.Items.Clear()

 ' Create a string variable array named  files to hold the directory and file entries for the
 ' special folder My Documents
 Dim files As String() = Directory.GetFileSystemEntries (Environment.GetFolderPath _
         (Environment.SpecialFolder.Personal))

Dim file As String
For Each file In files
  With ListBox1.Items
   .Add(file)
  End With
 Next
End Sub
```

C#

```
private void listFiles_Click(object sender, System.EventArgs e)
 {
  listBox1.Items.Clear();

  string[] files = Directory.GetFileSystemEntries
(Environment.GetFolderPath
        (Environment.SpecialFolder.Personal));

  foreach (string file in files)
       listBox1.Items.Add(file);
}
```

4. StreamWriter Class

Method	***Description***
Close	Closes the current StringWriter and the underlying stream.
Flush	Clears all buffers for the current writer and causes any buffered data to be written to the underlying device.
GetStringBuilder	Returns the underlying StringBuilder.
ToString	Returns a string containing characters written to current StringWriter so far.
Write	Writes to this instance of the StringWriter.
WriteLine	Writes some data as specified by the overloaded parameters, followed by a line terminator.

Using the StreamWriter Class

VB.NET

```
Private Sub WriteToFile_Click(ByVal sender As System.Object, _
   ByVal e As System.EventArgs) Handles WriteToFile.Click
Dim sw As New StreamWriter("C:\List.txt")
Dim s As String
 For Each s In ListBox1.Items
  sw.WriteLine(s)
 Next
sw.Close()
End Sub
```

C#

```
private void WriteToFile_Click(object sender, System.EventArgs e)
 {
  StreamWriter sw = new StreamWriter(@"C:\List.txt");
  foreach (string s in listBox1.Items)
sw.WriteLine(s);
  sw.Close();
 }
```

5. StreamReader Class

Method	***Description***
Close	Closes the StreamReader and releases any associated system resources
DiscardBufferedData	Allows a StreamReader to discard its current data.
Peek	Returns the next available character but doesn't consume it.
Read	Reads the next character or next set of characters from the input stream.

5. StreamReader Class

Method	*Description*
ReadBlock	Reads a maximum of count characters from the current stream and writes the data to buffer, beginning at *index*.
ReadLine	Reads a line of characters from the stream and returns the data as a string.
ReadToEnd	Reads the stream from the current position to the end of the stream.

Using the StreamReader Class

VB.NET

```
Private Sub ReadFromFile_Click(sender As System.Object, _
   ByVal e As System.EventArgs) Handles ReadFromFile.Click
ListBox1.Items.Clear()

Dim sr As New StreamReader("C:\List.txt")

' Use Peek method to move to next character and ReadLine method
to get the next line
 While sr.Peek <> -1
  ListBox1.Items.Add(sr.ReadLine)
End While

sr.Close()
End Sub
```

C#

```
private void ReadFromFile_Click(object sender, System.EventArgs
e)
 {
  listBox1.Items.Clear();

  StreamReader sr = new StreamReader(@"C:\List.txt");
```

```
 do
  {
   listBox1.Items.Add(sr.ReadLine());
  }
  while (sr.Peek() != -1);

  sr.Close();
}
```

Notice (1)

An alternative to **StreamReader** and **StreamWriter** is using the RichTextBox

- **For saving :**

```
richTextBox1.SaveFile
("f1.txt",RichTextBoxStreamType.PlainText)
```

- **For loading :**

```
richTextBox1.LoadFile
("f1.txt",RichTextBoxStreamType.PlainText)
```

Processes and Threads

Process Class

Process class is found in the **System.Diagnostics** namespace.
Process is used to open another process (program in execution) .Using **Process.Start**.

Important Methods of the Process Class

GetProcesses	Creates an array of new Process components and associates them with existing process resources.
Kill	Immediately stops the associated process.
Start	Starts a process resource and associates it with a Process component.

Do not forget **using System.Diagnostics;**

```
private void button1_Click(object sender, System.EventArgs e)
{
        // Start Internet Explorer. Defaults to the home page.
          Process.Start("IExplore.exe");
}

private void button2_Click(object sender, System.EventArgs e)
{
       // url's not considered documents. They only opened by passing them as arguments.
          Process.Start("IExplore.exe",
"www.northwindtraders.com");

          // Start a Web page using a browser associated with .html and .asp files.
          Process.Start("IExplore.exe", "C:\\myPath\\myFile.htm");

          Process.Start("notepad.exe", "C:\\readme.txt");
}
```

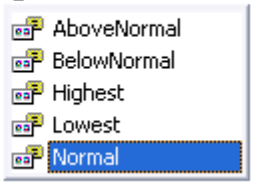
Uses of Thread class

1) Thread.Sleep(*t*) will halt the current executing thread, *t* is time in milliseconds

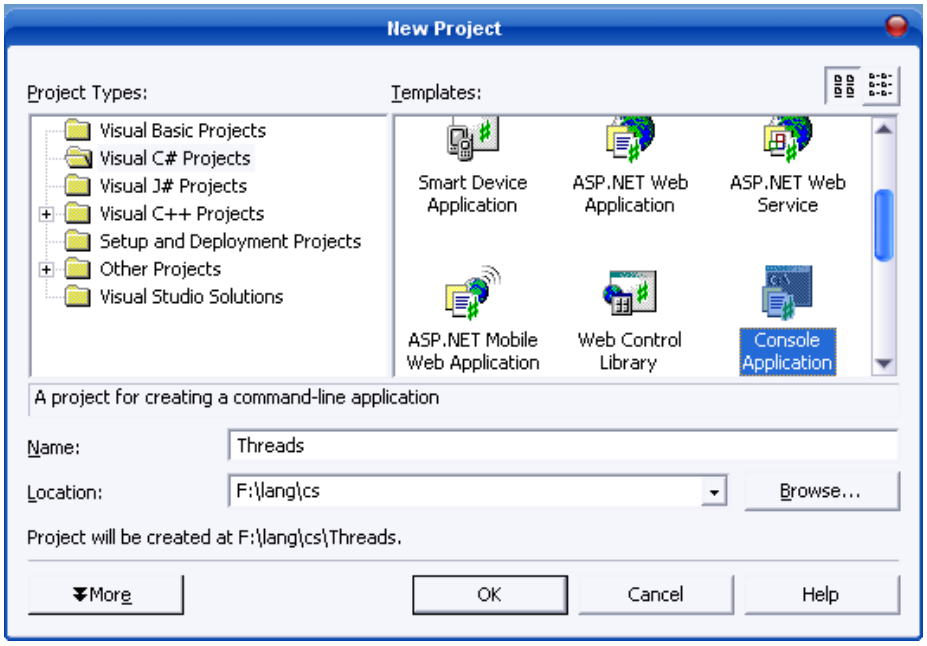
2) concurrency = multi-programming
[a] multi-user same task for many users
[b] multi-tasking different tasks

Thread constructor takes **ThreadStart** object , which takes name of the task (method)

```
t1.Priority = ThreadPriority.
```



Start( ) , Suspend( ) , and **Resume( )** for handling threads



Do not forget **using System.Threading;**

```
static void Main(string[] args )
{
        Thread t1 = new Thread (new ThreadStart(taskA) );
        Thread t2 = new Thread (new ThreadStart(taskB) );
t1.Start();
        t2.Start();
}

static void taskA()
{
        while(true)
        {
                System.Console.WriteLine("A");
                Thread.Sleep(500);
        }
}

static void taskB()
{
        while(true)
        {
                System.Console.WriteLine("B");
                Thread.Sleep(500);
        }
}
```

Windows Forms Applications

A Few Common Properties

Name	This name appears at the top of the Properties window when a control is selected on the form and is also used in programming the control. To set the text on a TextBox control from within your code, you will use say : TextBox1.Text = "My TextBox Control"
Font	determines how the text of the control will be rendered at both design and runtime.
Enabled	By default, all controls are enabled. To disable a control, set its Enabled property to False.
Size	Sets, or returns, the control's size. which exposes two properties: Width & Height. You can set the Size property to a string like "320, 80" or expand the Size property in the Properties window and set the Width and Height individually.
Tag	Holds some data you want to associate with a specific control. **For example**, you can set the Tag property to the control's default value, so that you can restore the control's default value if the user supplies invalid data
Text	The text (a string) that appears on the control. A control that displays multiple items, like ListBox or ComboBox, returns the currently selected item in the Text property.

TabStop	Only one control at a time can have the focus on a form. To move the focus from one control to the other on the same form, you press the Tab key (and Shift + Tab to move the focus in reverse order). TabStop property determines whether the control belongs to the tab order. If True (default), you can move the focus to the control with the Tab key. If False, the control will be skipped in the tab order.
TabIndex	A numeric value that determines the position of the control in the Tab order. The control with the smallest TabIndex value is the one that has the focus when the form is first loaded. If you press Tab once, the focus will be moved to the control with the next larger TabIndex value.
Visible	To make a control invisible, we set its Visible property to False

A Few Common Events

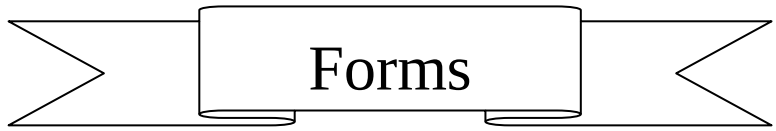
Click	The most common event in Windows programming, and is fired when a control is clicked.
DoubleClick	Fired when the control is double-clicked.
Enter	Fired when the control received the focus.
Leave	Fired when the control loses the focus. **For example**, put code to validate control content in this event handler.

MouseEnter	Fired when mouse pointer enters the area of the control. This event is fired once. If you want to monitor the movement of the mouse over a control, use the **MouseMove** event.
MouseLeave	fired when the mouse pointer moves out of the area of the control.
XXXChanged	Some events are fired when a property changes value. These events include BackColorChanged, FontChanged, VisibleChanged, and more.

A Few Common Methods

Focus	This method moves the focus to the control to which the method applies, regardless of the control that has the focus at the time. It's possible to "trap" the focus to a specific TextBox control until the user enters a valid value, by calling the Focus method from within the Leave event. The following code segment doesn't allow users to move the focus to another control while TextBox1 doesn't contain a valid numeric value: Private Sub TextBox1_Leave(ByVal sender As Object, _ ByVal e As System.EventArgs) Handles TextBox1.Leave If Not IsNumeric(TextBox1.Text) Then TextBox1.Focus() End Sub
Clear	Many controls provide a method to clear their contents, and this is the Clear method. When you call the Clear method on a TextBox control, the control's Text property is set to an empty string.
Hide/Show	Hide and Show methods reveal or conceal the control. The two methods are equivalent to setting the Visible property to True and False

PerformClick	It's common to invoke the Click event of a button from within our code. To do so, call the PerformClick method of the Button control.
Scale	To scale the TextBox1 control down to 75 percent of its current size, TextBox1.Scale(0.75)



Properties of the Form Control

- **AcceptButton, CancelButton**

Accept button is the one that's automatically activated when you press Enter, no matter which control has the focus at the time, and is usually the button with the OK caption.
Likewise, the Cancel button is the one that's automatically activated when you hit the Esc key and is usually the button with the Cancel caption.

- **AutoScale**

This is a True/False property that determines if the controls you place on the form are automatically scaled to the height of the current font. When you place a TextBox control on the form, for example, and the AutoScale property is True, the control will be tall enough to display a single line of text in the current font. Default is True.

- **AutoScroll**

AutoScroll property is a True/False value that indicates whether scroll bars will be automatically attached to the form.

- **ControlBox**

True by default. Set it to False to hide the icon and disable the Control menu.

- **MinimizeBox, MaximizeBox**

These two properties are True by default. Set them to False to hide the corresponding buttons on the title bar.

- **BorderStyle** < FormBorderStyle Enumeration>

Value	**Effect**
None **(be avoided)**	Borderless window that can't be resized
Sizable **(default)**	Resizable window that's used for displaying regular forms.
Fixed3D	Window with a visible border, "raised" relative to the main area. Can't be resized.
FixedDialog	A fixed window, used to create dialog boxes.
FixedSingle	A fixed window with a single line border.
FixedToolWindow	A fixed window with a Close button only. It looks like the toolbar displayed by the drawing and imaging applications.
SizableToolWindow	Same as the FixedToolWindow but resizable. In addition, its caption font is smaller than the usual.

Handling Keystrokes in the Form's KeyDown Event Handler

```
Sub Form1_KeyDown(sender As Object, ByVal e As KeyEventArgs) Handles Form1. KeyDown
If e.Keycode = Keys.F10 Then
MsgBox( "There are " & Contacts.Count.ToString & " contacts in the database")
e.Handled = True
End If

If e.KeyCode = Keys.Subtract And e.Modifiers = Keys.Alt Then
bttnPrevious_Click(sender, e)
End If

If e.KeyCode = Keys.Add And e.Modifiers = Keys.Alt Then
bttnNext_Click(sender, e)
End If
End Sub
```

Cancelling the Closing of a Form

```
Sub Form1_Closing(sender As Object, ByVal e As CancelEventArgs) Handles Form1.Closing
Dim reply As MsgBoxResult
reply = MsgBox( "Click OK to terminate the application, or Cancel to return ." , _
MsgBoxStyle.OKCancel)

If reply = MsgBoxResult.Cancel Then
e.Cancel = True
End If
End Sub
```

The Paint Event

```
 Sub RepaintForm()
    Dim grbrush As New _
System.Drawing.Drawing2D.LinearGradientBrush( _
New Point(0, 0), _
New Point(Me.width, Me.height), Color.Red, Color.Yellow)

Me.CreateGraphics.FillRectangle(grbrush, New Rectangle(0,_
0,Me.Width, Me.Height))
End Sub

Public Sub Form1_Paint(sender As Object, e As PaintEventArgs)
Handles Form1.Paint
RepaintForm()
End Sub

Sub Form1_Resize(sender As Object, e As EventArgs) Handles_
Form1.Resize

RepaintForm()
End Sub
```

IDE provides a visual tool for menus, then you program their Click event handlers.

The MenuItem Object's Properties

- **Checked** Some menu commands act as toggles, and they are usually checked to indicate that they are on or unchecked to indicate that they are off.

For example, to toggle status of a menu command called FntBold:

```
FntBold.Checked = Not FntBold.Checked
```

- **Enabled** Some menu commands aren't always available. The Paste command, for example, has no meaning if the Clipboard is empty. The following statements enable and disable the Undo command depending on whether the TextBox1 control can undo the most recent operation.

```
If TextBox1.CanUndo Then
cmdUndo.Enabled = True
Else
cmdUndo.Enabled = False
End If
```

- **Visible** To remove a command temporarily from the menu, set the command's Visible property to False. Enabled property is preferred.

VB.NET

```
Sub firstForm_Load(ByVal sender As Object, ByVal e As EventArgs)
Handles MyBase.Load
    ' Declare new instances of the RadioButton control class
    Dim Rd1 As RadioButton = New RadioButton()
    Dim Rd2 As RadioButton = New RadioButton()
    Dim Rd3 As RadioButton = New RadioButton()

     Rd1.Location = New System.Drawing.Point(15, 90)
     Rd2.Location = New System.Drawing.Point(15, 120)
     Rd3.Location = New System.Drawing.Point(15, 150)

    Rd1.Text = "Red"
    Rd2.Text = "White"
    Rd3.Text = "Blue"

    Me.Controls.AddRange(New Control() {Rd1, Rd2, Rd3})
```

```
    AddHandler Rd1.Click, AddressOf GenericClick
    AddHandler Rd2.Click, AddressOf GenericClick
    AddHandler Rd3.Click, AddressOf GenericClick
  End Sub

Public Sub GenericClick(ByVal sender As Object, ByVal e As
EventArgs)
        Select Case sender.text
            Case "Red"
                Me.BackColor = Color.Red
            Case "White"
                Me.BackColor = Color.White
            Case "Blue"
                Me.BackColor = Color.Blue
        End Select
    End Sub
```

C#

```
private void firstForm_Load(object sender, System.EventArgs e)
{
   // Declare new instances of the RadioButton control class
   RadioButton rd1 = new RadioButton();
   RadioButton rd2 = new RadioButton();
   RadioButton rd3 = new RadioButton();

   rd1.Location = new System.Drawing.Point(15, 90);
   rd2.Location = new System.Drawing.Point(15, 120);
   rd2.Location = new System.Drawing.Point(15, 150);

   rd1.Text = "Red";
   rd2.Text = "White";
   rd3.Text = "Blue";
   this.Controls.AddRange(new Control[]  {rd1, rd2, rd3});

   rd1.Click += new System.EventHandler(genericClick);
   rd2.Click += new System.EventHandler(genericClick);
   rd3.Click  += new System.EventHandler(genericClick);
       }
```

```
private void genericClick(object sender, System.EventArgs e)
{
   RadioButton rdb;
   rdb = (RadioButton)sender;
   this.BackColor = Color.FromName(rdb.Text);
}
```

Reading the Controls with a For Each…Next Loop

```
Sub bttnProcss_Click(ByVal sender As Object, ByVal e As
EventArgs) Handles bttnProcss.Click
Dim TB As Control
Dim Sum As Double = 0, points As Integer = 0
For Each TB In Me.Controls
If TB.GetType Is GetType(Windows.Forms.TextBox) Then
If IsNumeric(CType(TB, TextBox).Text) Then
Sum += CType(TB, TextBox).Text
points += 1
End If
End If
Next
MsgBox( “The sum of the “ & points.ToString & “ data points
is “ & Sum.ToString)
End Sub
```

With an Multiple Document Interface (MDI) application, you can maintain multiple open windows.

Create a new windows application project, rename the Form1 to ParentForm,
Change the **IsMdiContainer** property to True,
Drag a MainMenu control from the Toolbox to the form, link it with the form.
Add a new form called ChildForm.

Build the menu as follows:

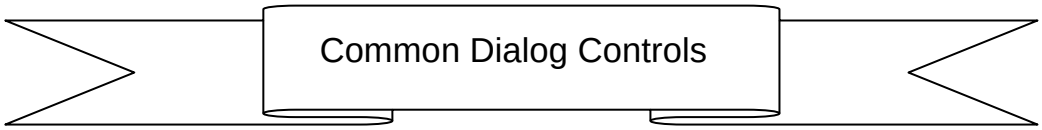
Text	**Name**
File	FileMenu
New	FileNew
Exit	FileExit
Window	WindowMenu
Tile Horizontally	WindowTileH
Tile Vertically	WindowTileV
Cascade	WindowCascade
Arrange Icons	WindowArrange

```
Sub FileNew_Click(sender As Object, ByVal e As EventArgs)
Handles FileNew.Click
Static nWindow As Integer
Dim child As New ChildForm()
child.MdiParent = Me
child.Text = “Child window # “ & nWindow.ToString
child.Show()
nWindow = nWindow + 1
End Sub

Private Sub WindowTileH_Click(ByVal sender As System.Object,
_
    ByVal e As System.EventArgs) Handles WindowTileH.Click
Me.LayoutMdi(MdiLayout.TileHorizontal)
End Sub
Private Sub WindowTileV_Click(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles WindowTileV.Click
Me.LayoutMdi(MdiLayout.TileVertical)
End Sub

Private Sub WindowCascade_Click(ByVal sender As
System.Object, _
    ByVal e As System.EventArgs) Handles WindowCascade.Click
Me.LayoutMdi(MdiLayout.Cascade)
End Sub
```

```
Private Sub WindowArrange_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles WindowArrange.Click
Me.LayoutMdi(MdiLayout.ArrangeIcons)
End Sub
```



Control	Description
ColorDialog	Displays the color picker dialog box
FontDialog	Displays a dialog box that enables users to set a font
OpenFileDialog	dialog box that enables users to navigate to and select a file
PrintDialog	dialog box that enables users to select a printer & its attributes
PrintPreviewDialog	dialog box shows how a PrintDocument object appears when printed
SaveFileDialog	Displays a dialog box that allows users to save a file

Using OpenFileDialog

VB.NET

```
Dim openFileDialog1 As New OpenFileDialog()
openFileDialog1.Filter = "Cursor Files|*.cur"
openFileDialog1.Title = "Select a Cursor File"

If openFileDialog1.ShowDialog() = DialogResult.OK Then
   If openFileDialog1.FileName <> "" Then
      ' Assign the cursor in the Stream to the Form's Cursor property.
      Me.Cursor = New Cursor(openFileDialog1.OpenFile())
   End If
End If
```

C#

```
OpenFileDialog openFileDialog = new OpenFileDialog();
openFileDialog.Filter = "Cursor Files|*.cur";
openFileDialog.Title = "Select a Cursor File";
if (openFileDialog.ShowDialog() == DialogResult.OK &&
   StringType.StrCmp(openFileDialog.FileName, "", false) != 0)
    {
     this.Cursor = new Cursor(openFileDialog.OpenFile());
    }
```

You can also read files using a combination of the System.IO classes

Controls Available in Windows Forms

Control Name	***Description***
RichTextBox	Enables text to be displayed with formatting in rich text format (RTF).
LinkLabel	Displays text as a Web-style link and triggers an event when the user clicks the special text. Usually the text is a link to another window or a Web site.
StatusBar	Displays information about the application's current state using a framed window, usually at the bottom of a parent form.
CheckedListBox	Displays a scrollable list of items, each accompanied by a check box.
ComboBox	Displays a drop-down list of items.
DomainUpDown	Displays a list of text items that users can scroll through with up and down
ListBox	Displays a list of text and graphical items (icons).

Control Name	***Description***
ListView	Displays items in one of four different views. Views include text only, text with small icons, text with large icons, and report view.
NumericUpDown	Displays a list of numerals that users can scroll through with up and down buttons.
TreeView	Displays a hierarchical collection of node objects that can consist of text with optional check boxes or icons.
PictureBox	Displays graphical files, such as bitmaps and icons, in a frame.
ImageList	Serves as a repository for images. ImageList controls and the images they contain can be reused from one application to the next.
CheckBox	Displays a check box and a label for text. Generally used to set options.
CheckedListBox	Displays a scrollable list of items, each accompanied by a check box.
RadioButton	Displays a button that can be turned on or off.
Trackbar	Allows users to set values on a scale by moving a "thumb" slider along a scale.
DateTimePicker	Displays a graphical calendar to enable users to select a date or a time.
MonthCalendar	Displays a graphical calendar to enable users to select a range of dates.
ContextMenu	Implements a menu that appears when the user right-clicks an object.

Control Name	***Description***
NotifyIcon	Displays an icon in the status notification area of the taskbar that represents an application running in the background.
ToolBar	Contains a collection of button controls.
Panel	Groups a set of controls on an unlabeled, scrollable frame.
GroupBox	Groups a set of controls (e.g. radio buttons) on a labeled, nonscrollable frame.
TabControl	tabbed page for organizing and accessing grouped objects efficiently.