**Input/Output Operations**

* Used to work with files/folders/drives and path

.NET provides various classes under **System.IO** namespace to work with files, folders and drives.

Main classes under System.IO namespace are

* **File** class
* **Directory** class
* **Path** class
* **FileSystemInfo** class
* **DriveInfo** class
* **FileInfo** class
* **DirectoryInfo** class

**Directory** and **File** classes contain only static methods are never instantiated.

**DirectoryInfo** and **FileInfo** classes implements the roughly the same public methods as Directory and File, as well as some properties and constructors. You need to create an object of these classes to use them.

**FileInfo** class members

* Constructors
  + FileInfo(string filename)
* Properties
  + bool Exists
    - Check whether the given file name exists or not
  + long Length
    - Returns the size of file in bytes
  + string Name
    - Returns the file name
  + string FullName
    - Returns the file name with full path
  + string Extension
    - Returns only the extension name
  + string DirectoryName
    - Returns the directory name
  + DateTime CreationTime
  + DateTime LastAccessTime
  + DataTime LastWriteTime
  + DirectoryInfo Directory
    - Returns an instance of Parent Directory
* Methods
  + void Delete()
    - Permanently deletes a file
  + void MoveTo(string target)
    - Move the file name to target folder or rename a file in same folder
  + void CopyTo(string target)
    - Copy file to given folder or create duplicate copy of file in same folder
  + StreamReader OpenText()

**Example**

* Show all the information about calculator program file calc.exe under windows

using System.IO;

using c = System.Console;

class FileInfoTest

{

static void Main(string[] args)

{

FileInfo f = new FileInfo(@"C:\WINDOWS\system32\Calc.exe");

c.WriteLine("Full Name : {0}", f.FullName);

c.WriteLine("File Name : {0}", f.Name);

c.WriteLine("Directory Name : {0}", f.DirectoryName);

c.WriteLine("Length : {0}", f.Length);

c.WriteLine("Extension : {0}", f.Extension);

c.WriteLine("Creation Time : {0}", f.CreationTime);

c.WriteLine("Last Access Time : {0}", f.LastAccessTime);

c.WriteLine("Last Write Time : {0}", f.LastWriteTime);

}

}

Write a program to get some files from command prompt and delete all the files

using System.IO;

using c=System.Console;

class MyDel

{

public static void Main(string []args)

{

if(args.Length<1)

{

c.WriteLine("Syntax is : MyDel <filelist>");

return;

}

foreach(string s in args)

{

FileInfo f=new FileInfo(s);

if(f.Exists)

f.Delete();

}

}

}

**Assignments**

1. Write a program to get file name from keyboard and display its properties
2. Write a program to get a file name from command prompt and show it properties
3. Write a program to get some files from command prompt and show properties of all the files
4. Write a program to get a file name and new name. Then rename the file

**DirectoryInfo** class Members

* Constructors
  + DirectoryInfo(string directoryname)
* Properties
  + bool Exists
    - Check whether the given file name exists or not
  + string Name
    - Returns the file name
  + string FullName
    - Returns the file name with full path
  + string Root
    - Returns the root portion of the directory
  + DateTime CreationTime
  + DateTime LastAccessTime
  + DataTime LastWriteTime
  + DirectoryInfo Parent
    - Returns an instance of Parent Directory
* Methods
  + void Delete(bool recursive)
    - Permanently deletes a folder
    - Set recursive as true to remove a directory structure
  + void MoveTo(string target)
    - Move the files target folder or rename a folder
  + DirectoryInfo[] GetDirectories()
  + DirectoryInfo GetDirectories(string searchPattern)
  + DirectoryInfo GetDirectories(string searchPattern, SearchOptions options)
    - Returns an array of DirectoryInfo objects representing all sub folders of current folder
  + FileInfo [] GetFiles()
  + FileInfo [] GetFiles(string searchPattern)
  + FileInfo [] GetFiles(string SearchPatterh, SearchOptions options)
    - Returns an array of FileInfo objects representing all files of current folder
  + FileSystemInfo[] GetFileSystemInfos()
    - Returns an array of FileSystemInfo references representing FileInfo and DirectoryInfo objects under current directory

Example

* Provide the informatino about the c:\windows folder and search all the files starting with A in windows and its sub-directories. Also list all sub directories of c:\windows

using System.IO;

using c = System.Console;

class DirectoryInfoTest

{

static void Main(string[] args)

{

DirectoryInfo d = new DirectoryInfo(@"C:\WINDOWS");

c.WriteLine("Full Name : {0}", d.FullName);

c.WriteLine("File Name : {0}", d.Name);

c.WriteLine("Creation Time : {0}", d.CreationTime);

c.WriteLine("Last Access Time : {0}", d.LastAccessTime);

c.WriteLine("Last Write Time : {0}", d.LastWriteTime);

//List all sub directories

DirectoryInfo[] di = d.GetDirectories();

foreach (DirectoryInfo dr in di)

c.WriteLine(dr.FullName);

//Lookup all the files starting with a in sub sub directories

FileInfo[] fi = d.GetFiles("a\*", SearchOption.AllDirectories);

foreach (FileInfo f in fi)

c.WriteLine(f.FullName);

}

}

**Assignments**

1. Write a program to access file name from keyboard and the search pattern to search the files. Show all the files for given pattern in given files
2. Write a program to show the files and sub folders recursively
3. Write a program to get a directory name remove it will all its contents

**Path** class

A class that cannot be instantiated and provides static methods that makes operations on pathnames easier.

Members of Path class

* string Combine(string folder, string filename)
* string GetFileName(string path)
* string GetDirectoyName(string path)
* string GetExension(string path)
* string GetTempPath()
  + Returns the path of temporary folder
* string GetTempFileName()
  + returns a unique temporary file name
* string GetRandomFileName()
  + return a random name for file or folder

Example

using System.IO;

using c = System.Console;

class PathTest

{

public static void Main()

{

c.WriteLine(@"Getting information for path : c:\windows\system32\calc.exe");

c.WriteLine("File Name : "+Path.GetFileName(@"c:\windows\system32\calc.exe"));

c.WriteLine("Extension : " + Path.GetExtension(@"c:\windows\system32\calc.exe"));

c.WriteLine("Temp Folder is : " + Path.GetTempPath());

c.WriteLine("New Temp File is : " + Path.GetTempFileName());

c.WriteLine("Random File or folder name : " + Path.GetRandomFileName());

}

}

**DriveInfo** class

A class that provides information about a drive. It can help in scanning a system to provide a list of available drives and then can dig in deeper, providing details about any of the drives.

Members of DriveInfo class

* DriveInfo [] GetDrives()
  + Provides list of all drives in current machine
* string Name
* string DriveFormat
* string DriveType
* bool IsReady
* long TotalSize
* long TotalFreeSpace
* string VolumeLabel
* string RootDirectory

Example

using System.IO;

using c = System.Console;

class DriveInfoTest

{

public static void Main()

{

DriveInfo[] di = DriveInfo.GetDrives();

c.WriteLine("Total Drives are : ");

foreach (DriveInfo df in di)

c.WriteLine(df.Name);

c.WriteLine("Properties of C: Drive are");

DriveInfo d=new DriveInfo(@"c:\");

c.WriteLine("Drive : " + d.Name);

c.WriteLine("Drive Format : " + d.DriveFormat);

c.WriteLine("Driver Type : " + d.DriveType);

c.WriteLine("Ready Status : " + d.IsReady);

c.WriteLine("Total Size : " + d.TotalSize);

c.WriteLine("Free Space : " + d.TotalFreeSpace);

c.WriteLine("Volume Level : "+d.VolumeLabel);

}

}

**Assignment**

Create a GUI Application to show all drives in a combo box. When a drive get selected, show all its directories in a list box. When a directory get selected, show all files of that directory into another list box. When a file get selected then show its properties in some text box. If the file is of Image type then preview it in some picturebox.

**File** class

* Helps in reading and writing text contents with a file

Members of File class

* string ReadAll(string filepath)
* string ReadAll(string filepath, Encoding type)
* string[] ReadLineLines()
* void WriteAll(string filepath, string text)
* void WriteAll(string filepath, string text, Encoding type)
* void WriteAllLines(string []contents)

Encodings managed by Encoding class can be

* ASCII
* Unicode
* UTF7
* UTF8
* UTF32

**StreamReader** class

Used to read contents of a text file

Members of StreamReader class

* Constructor
  + StreamReader(string filename)
  + StreamReader(string filename, Encoding type)
  + StreamReader(FileStream fs)
* Methods
  + int Read()
  + int Read(char []variable, int start, int end)
  + string ReadLine()
  + string ReadToEnd()
  + void Close()

Example

using System.IO;

using c = System.Console;

class StreamReaderTest

{

public static void Main()

{

StreamReader sr = new StreamReader(@"c:\sample.txt");

string data = sr.ReadToEnd();

c.WriteLine(data);

sr.Close();

}

}

**StreamWriter class**

Used to write contents to a text file.

Members of StreamReader

* Constructors
  + StreamWriter(string filepath)
  + StreamWriter(string filepath, bool appendmode, Encoding type)
* Methods
  + void Write(string text)
  + void WriteLine(string line)
  + void Close()

Example

using System.IO;

using c = System.Console;

class StreamWriterTest

{

public static void Main()

{

StreamWriter sw = new StreamWriter(@"c:\abc.txt");

string str = "Hello";

sw.Write(str);

sw.Close();

}

}

**FileStream** class

Used to create a reference of a file to read, write, append and use by StreamReader and StreamWriter for some purpose.

Constructor

FileStream(string filename, FileMode fm, FileAccess fa, FileShare fh)

FileMode can be

* Append
* Create
* CreateNew
* Open
* OpenOrCreate
* Truncate

FileAccess can be

* Read
* Write
* ReadWrite

FileShare can be

* None
* Read
* Write
* Delete
* ReadWrite

Example

using System.IO;

using c = System.Console;

class StreamReaderTest

{

public static void Main()

{

FileStream fs = new FileStream(@"c:\sample.txt", FileMode.Open, FileAccess.Read, FileShare.None);

StreamReader sr = new StreamReader(fs);

string data = sr.ReadToEnd();

c.WriteLine(data);

sr.Close();

}

}

Note:

It is a class inherited from IDisposable interface. It allows creating disposable object.

using System.IO;

using c = System.Console;

class StreamReaderTest

{

public static void Main()

{

using (FileStream fs = new FileStream(@"c:\sample.txt", FileMode.Open, FileAccess.Read, FileShare.None))

{

StreamReader sr = new StreamReader(fs);

string data = sr.ReadToEnd();

c.WriteLine(data);

sr.Close();

}

}

}

**What are disposable objects? Why we need them? How to create them?**

* Special object that get removed automatically when they goes out of scope and do not wait from Garbage Collection
* It can be applied only those classes which are inherited from **IDisposable** interface
* Use **using** keyword to create such objects

using(SqlConnection cn=new SqlConnection())

{

using(SqlDataReader dr=cmd.ExecutReader())

{

}

}

DriveInfo []di=DriveInfo.GetDrives();

foreach(DriveInfo d in di)

{

if(d.IsReady)

{

DirectoryInfo dir=new DirectoryInfo(d.RootDirectory);

string []result=dir.GetFiles(filename).FullName;

foreach(string x in result)

ListBox1.Items.Add(x);

}

}