**Working with Files & Directories:**

C# provides the following classes to work with the File system. They can be used to access directories, access files, open files for reading or writing, create a new file or move existing files from one location to another, etc.

| Class Name | Usage |
| --- | --- |
| File | File is a static class that provides different functionalities like copy, create,  move, delete, open for reading or /writing, encrypt or decrypt, check if a file  exists, append lines or text to a file’s content, get last access time, etc. |
| FileInfo | The FileInfo class provides the same functionality as a static File class. You have more  control on how you do read/write operations on a file by writing code manually for  reading or writing bytes from a file. |
| Directory | Directory is a static class that provides functionality for creating, moving, deleting and  accessing subdirectories. |
| DirectoryInfo | DirectoryInfo provides instance methods for creating, moving, deleting and accessing  subdirectories. |
| Path | Path is a static class that provides functionality such as retrieving the extension of a file,  changing the extension of a file, retrieving the absolute physical path, and other path  related functionalities. |

**File:**

C# includes static File class to perform I/O operation on physical file system. The static File class includes various utility method to interact with physical file of any type e.g. binary, text etc.

Use this static File class to perform some quick operation on physical file. It is not recommended to use File class for multiple operations on multiple files at the same time due to performance reasons. Use FileInfo class in that scenario.

**Important Methods of Static File Class:**

| Method | Usage |
| --- | --- |
| AppendAllLines | Appends lines to a file, and then closes the file. If the specified file does not exist, this method creates a file, writes the specified lines to the file, and then closes the file. |
| AppendAllText | Opens a file, appends the specified string to the file, and then closes the file. If the file does not exist, this method creates a file, writes the specified string to the file, then closes the file. |
| AppendText | Creates a StreamWriter that appends UTF-8 encoded text to an existing file, or to a new file if the specified file does not exist. |
| Copy | Copies an existing file to a new file. Overwriting a file of the same name is not allowed. |
| Create | Creates or overwrites a file in the specified path. |
| CreateText | Creates or opens a file for writing UTF-8 encoded text. |
| Decrypt | Decrypts a file that was encrypted by the current account using the Encrypt method. |
| Delete | Deletes the specified file. |
| Encrypt | Encrypts a file so that only the account used to encrypt the file can decrypt it. |
| Exists | Determines whether the specified file exists. |
| GetAccessControl | Gets a FileSecurity object that encapsulates the access control list (ACL) entries for a specified file. |
| Move | Moves a specified file to a new location, providing the option to specify a new file name. |
| Open | Opens a FileStream on the specified path with read/write access. |
| ReadAllBytes | Opens a binary file, reads the contents of the file into a byte array, and then closes the file. |
| ReadAllLines | Opens a text file, reads all lines of the file, and then closes the file. |
| ReadAllText | Opens a text file, reads all lines of the file, and then closes the file. |
| Replace | Replaces the contents of a specified file with the contents of another file, deleting the original file, and creating a backup of the replaced file. |
| WriteAllBytes | Creates a new file, writes the specified byte array to the file, and then closes the file. If the target file already exists, it is overwritten. |
| WriteAllLines | Creates a new file, writes a collection of strings to the file, and then closes the file. |
| WriteAllText | Creates a new file, writes the specified string to the file, and then closes the file. If the target file already exists, it is overwritten. |

**Append text lines:**

Use AppendAllLines() method to append multiple text lines to the specified file as shown below.

Example: Append all text lines to a file

string dummyLines = "This is first line." + Environment.NewLine +

"This is second line." + Environment.NewLine +

"This is third line.";

//Opens DummyFile.txt and append lines. If file is not exists then create and open.

File.AppendAllLines(@"C:\DummyFile.txt", dummyLines.Split(Environment.NewLine.ToCharArray()).ToList<string>());

**Append text string:**

Use File.AppendAllText() method to append string to a file in single line of code as shown below.

//Opens DummyFile.txt and append Text. If file is not exists then create and open.

File.AppendAllText(@"C:\ DummyFile.txt", "This is File testing");

**Overwrite existing texts:**

Use File.WriteAllText() method to write texts to the file. Please note that it will not append text but overwrite existing texts.

//Opens DummyFile.txt and write texts. If file is not exists then create and open.

File.WriteAllText(@"C:\DummyFile.txt", "This is dummy text");

The following example shows how to perform different operations using static File class.

Example: Multiple File operations

//Check whether file is exists or not at particular location

bool isFileExists = File.Exists(@"C:\ DummyFile.txt"); // returns false

//Copy DummyFile.txt as new file DummyFileNew.txt

File.Copy(@"C:\DummyFile.txt", @"D:\NewDummyFile.txt");

//Get when the file was accessed last time

DateTime lastAccessTime = File.GetLastAccessTime(@"C:\DummyFile.txt");

//get when the file was written last time

DateTime lastWriteTime = File.GetLastWriteTime(@"C:\DummyFile.txt");

// Move file to new location

File.Move(@"C:\DummyFile.txt", @"D:\DummyFile.txt");

//Open file and returns FileStream for reading bytes from the file

FileStream fs = File.Open(@"D:\DummyFile.txt", FileMode.OpenOrCreate);

//Open file and return StreamReader for reading string from the file

StreamReader sr = File.OpenText(@"D:\DummyFile.txt");

//Delete file

File.Delete(@"C:\DummyFile.txt");

Thus, it is easy to work with physical file using static File class. However, if you want more flexibility then use FileInfo class. The same way, use static Directory class to work with physical directories.

**Points to Remember:**

1. File is a static class to read\write from physical file with less coding.
2. Static File class provides functionalities such as create, read\write, copy, move, delete and others for physical files.
3. Static Directory class provides functionalities such as create, copy, move, delete, etc. for physical directories with less coding.
4. FileInfo and DirectoryInfo class provides same functionality as static File and Directory class.