

ChatGPT - C# Examples

You are a C# tutor. Explain the FileInfo class in C# for a developer with around 3 years of experience. Cover the following in detail:

1. Concepts

What is the FileInfo class, its purpose, and when to use it.


Explain how it differs from the File class, and when to prefer FileInfo over File.

Discuss the object-oriented advantages of FileInfo (like instance-based operations instead of static ones).

Provide real-world scenarios where FileInfo is useful (e.g., reading file metadata, managing log or report files, copying/moving files dynamically).

2. Must-Know Properties and Methods

Explain the most commonly used properties and methods, with simple, clear code examples and outputs for each:

 Properties:

Name

FullName

Extension

Length

CreationTime

LastAccessTime

DirectoryName

Exists

⚙️ Methods:

CopyTo()

MoveTo()

Delete()

Refresh()

For each property and method, include:

A short, simple C# example

Explanation of what it does

Expected output

3. Differences Section

Compare FileInfo vs File class clearly with a comparison table or bullet points based on:

Object-oriented vs static usage

Performance

Ease of use

When to use each in real-world development

Also briefly explain how FileInfo fits alongside FileStream, StreamReader, and StreamWriter.

4. Practical Example

Provide a complete, commented C# program that demonstrates:

Creating a FileInfo object

Checking file properties

Copying and moving a file

Deleting a file safely

Handling exceptions

Using Refresh()

Include clear expected output so learners can understand the behavior.

5. Best Practices and Interview Insights

Demonstrate proper exception handling (try-catch-finally or using blocks).

Discuss performance and memory considerations.

Highlight common interview questions like:

Difference between File and FileInfo

When to use FileInfo

How to get file metadata efficiently

Excellent — let's deep dive into the `FileInfo` class in C# exactly like a **3-year-experience developer** should understand it: practically, deeply, and with real-world usage and interview focus.

1. Concepts

◆ What is `FileInfo`?

`FileInfo` is a class in the `System.IO` namespace used to work with files as objects — it provides properties and methods to get detailed information about a

file (like name, size, creation time, etc.) and to **perform operations** (copy, move, delete).

 **Namespace:** `System.IO`



 **Assembly:** `mscorlib.dll`

Purpose

It allows you to **inspect and manipulate files** in an **object-oriented** way:

- Retrieve file metadata (name, size, creation date)
 - Copy, move, or delete files
 - Refresh file info after changes
-

Difference: `FileInfo` vs `File`

Feature	<code>FileInfo</code>	<code>File</code>
Type	Instance-based class	Static class
Usage	Create an object for a specific file	Call static methods directly
Memory	Slightly more memory (object instance)	Lightweight (no object creation)
Performance	Better for multiple operations on the same file	Better for single quick actions
Example	<pre>FileInfo info = new FileInfo("data.txt"); info.CopyTo("backup.txt");</pre>	<pre>File.Copy("data.txt" , "backup.txt");</pre>
Object-Oriented	 Supports OOP features like instance reuse	 Static procedural style
Use When	Repeatedly working with the same file	Need one-off operations

Real-World Scenarios

Scenario	Why <code>FileInfo</code>
Logging	Get file size, creation time, and delete logs older than 7 days
Report Management	Move or copy generated reports dynamically
File Maintenance Tasks	Check existence, get metadata, and clean up old files
Backup Utility	Copy files to backup folders with progress logs

2. Must-Know Properties and Methods

Let's go step-by-step through the most useful members of `FileInfo`.

Properties

1. `Name`

Returns the file name with extension.

```
csharp
```

```
FileInfo fi = new FileInfo(@"C:\Data\Report.txt");  
Console.WriteLine(fi.Name);
```

Output:

```
Report.txt
```

2. `FullName`

Returns the full path of the file.

```
csharp
```

```
Console.WriteLine(fi.FullName);
```

Output:

```
makefile
```

```
C:\Data\Report.txt
```

3. Extension

Gets the file's extension.

```
csharp
```

```
Console.WriteLine(fi.Extension);
```

Output:

```
.txt
```

4. Length

Returns file size in bytes.

```
csharp
```

```
Console.WriteLine(fi.Length + " bytes");
```

Output:

```
python
```

```
1204 bytes
```

5. CreationTime

Shows when the file was created.

```
csharp
```

```
Console.WriteLine(fi.CreationTime);
```

Output:

```
swift
```

```
10/06/2025 08:42:10 PM
```

6. LastAccessTime

Shows when the file was last accessed.

```
csharp
```

```
Console.WriteLine(fi.LastAccessTime);
```

Output:

```
swift
```

```
10/06/2025 08:45:12 PM
```

7. DirectoryName

Returns the directory path where the file exists.

```
csharp
```

```
Console.WriteLine(fi.DirectoryName);
```

Output:

```
makefile
```

```
C:\Data
```

8. Exists

Checks whether the file actually exists.

```
csharp
```

```
Console.WriteLine(fi.Exists);
```

Output:

```
graphql
```

```
True
```

Methods

1. CopyTo(destinationPath)

Copies the file to a new location.

```
csharp
```

```
fi.CopyTo(@"C:\Backup\Report_Copy.txt", true);  
Console.WriteLine("File copied successfully.");
```

Output:

```
arduino
```

```
File copied successfully.
```

2. MoveTo(destinationPath)

Moves the file to a new location.

```
csharp
```

```
fi.MoveTo(@"C:\Archive\Report_2025.txt");  
Console.WriteLine("File moved successfully.");
```

Output:

```
arduino
```


File moved successfully.

3. Delete()

Deletes the file.

```
csharp

fi.Delete();
Console.WriteLine("File deleted successfully.");
```

Output:

```
arduino

File deleted successfully.
```

4. Refresh()

Refreshes the file information from the file system (useful when file metadata changes after object creation).

```
csharp

fi.Refresh();
Console.WriteLine("File information refreshed.");
```

3. Comparison Summary

Feature	FileInfo	File
Style	Object-Oriented (Instance-based)	Procedural (Static methods)
Reuse	Can reuse the same object	No reuse; call methods each time
Best for	Repeated operations on same file	One-time operations

Feature	FileInfo	File
Performance	Slightly slower to create, faster for reuse	Fast for one-time
Example	<code>fi.Delete()</code>	<code>File.Delete(path)</code>



How `FileInfo` Fits with `FileStream`, `StreamReader`, and `StreamWriter`

Class	Purpose
FileInfo	Manage files and metadata (size, creation date, etc.)
FileStream	Low-level byte reading/writing
StreamReader/Writer	High-level text reading/writing
File	Static helper methods for quick file operations

Think of it like:

- ◆ **FileInfo** = “File manager”
- ◆ **FileStream** = “Raw data pipe”
- ◆ **StreamReader/Writer** = “Text data handler”
- ◆ **File** = “Quick utility toolbox”



4. Practical Example — Full Program

csharp

```
using System;
using System.IO;

class FileInfoDemo
{
    static void Main()
    {
```

```

try
{
    string sourcePath = @"C:\Temp\Sample.txt";
    string copyPath = @"C:\Temp\Backup_Sample.txt";
    string movePath = @"C:\Temp\Moved_Sample.txt";

    // Create the file if it doesn't exist
    if (!File.Exists(sourcePath))
    {
        File.WriteAllText(sourcePath, "This is a sample file for FileInfo demo.");
    }

    // Create a FileInfo object
    FileInfo fi = new FileInfo(sourcePath);

    Console.WriteLine("=== File Info Details ===");
    Console.WriteLine($"Name: {fi.Name}");
    Console.WriteLine($"Full Path: {fi.FullName}");
    Console.WriteLine($"Extension: {fi.Extension}");
    Console.WriteLine($"Length: {fi.Length} bytes");
    Console.WriteLine($"Created On: {fi.CreationTime}");
    Console.WriteLine($"Last Accessed: {fi.LastAccessTime}");
    Console.WriteLine($"Directory: {fi.DirectoryName}");
    Console.WriteLine($"Exists: {fi.Exists}");
    Console.WriteLine();

    // Copy the file
    fi.CopyTo(copyPath, true);
    Console.WriteLine("File copied successfully.");

    // Move the copied file
    FileInfo copyFile = new FileInfo(copyPath);
    copyFile.MoveTo(movePath);
    Console.WriteLine("Copied file moved successfully.");

    // Delete the moved file safely
    if (File.Exists(movePath))
    {
        FileInfo movedFile = new FileInfo(movePath);
        movedFile.Delete();
        Console.WriteLine("Moved file deleted successfully.");
    }
}

```

```
// Refresh example
fi.Refresh();
Console.WriteLine("\nFile info refreshed successfully.");
}
catch (Exception ex)
{
    Console.WriteLine($"Error: {ex.Message}");
}
finally
{
    Console.WriteLine("\nOperation complete.");
}
}
```



Expected Output (Example)

mathematica

=== File Info Details ===

Name: Sample.txt

Full Path: C:\Temp\Sample.txt

Extension: .txt

Length: 40 bytes

Created On: 10/06/2025 08:42:12 PM

Last Accessed: 10/06/2025 08:42:12 PM

Directory: C:\Temp

Exists: True

File copied successfully.

Copied file moved successfully.

Moved file deleted successfully.

File info refreshed successfully.

Operation complete.



5. Best Practices and Interview Insights



Best Practices

- Always check `fi.Exists` before operations.
 - Use `try-catch-finally` for IO operations.
 - Call `Refresh()` if file properties might have changed.
 - Prefer `FileInfo` for repeated operations on the same file.
 - Dispose streams (`FileStream` , `StreamReader` , `StreamWriter`) properly after use.
-



Performance & Memory Considerations

- Creating a `FileInfo` object is slightly heavier (it stores metadata).
 - Ideal for long-running apps or repeated file access.
 - For quick, single operations, prefer the static `File` class.
-



Common Interview Questions

Question	Short Answer
What is the difference between <code>File</code> and <code>FileInfo</code> ?	<code>File</code> → static, quick use; <code>FileInfo</code> → instance-based, reusable object.
When should you use <code>FileInfo</code> ?	When performing multiple operations or needing metadata repeatedly.
How do you get file size or creation date?	Using <code>FileInfo.Length</code> and <code>FileInfo.CreationTime</code> .
What does <code>Refresh()</code> do?	Updates the object's data to reflect current file system info.

Would you like me to follow the **same teaching structure** (Concept → Methods/Properties → Comparison → Practical Example → Interview Insights) for

the next topic — for example, **DirectoryInfo** class or **FileStream** vs **FileInfo** real-world usage comparison?