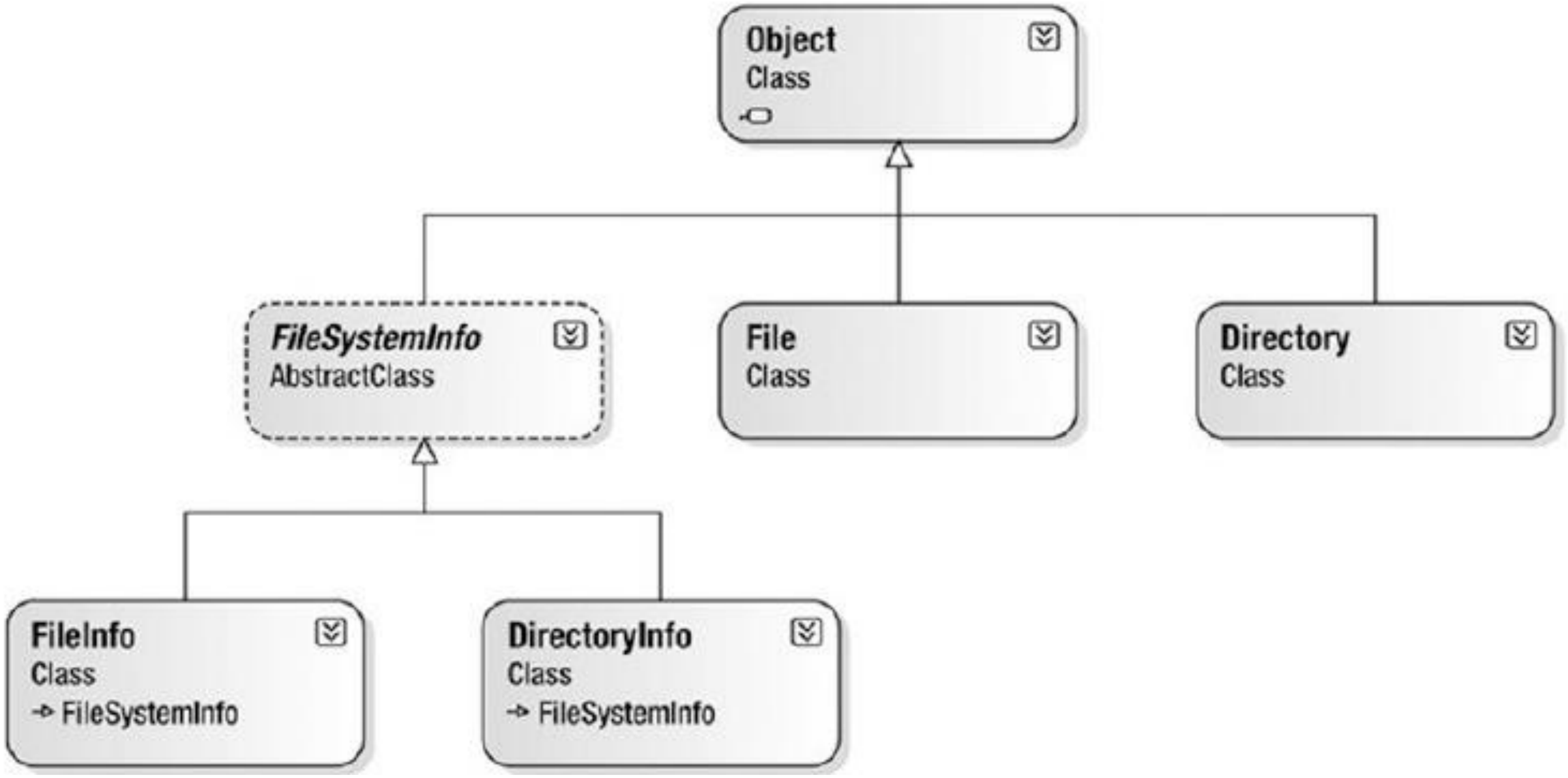
File IO

(Pro C# 7 With .Net and .NET Core – Troelsen, Japikse - apress)


Object
Class
FileSystemInfo
AbstractClass
File
Class
Directory
Class
FileInfo
Class
FileSystemInfo
DirectoryInfo
Class
FileSystemInfo

DirectoryInfo versus Directory

Directory

- Static
- Aanmaken, verplaatsen, oplijsten (sub)directories
- Gebruik : eenvoudige folder operatie (vb verwijderen folder)
- Vb :
  - Directory.CreateDirectory(path);
  - Directory.Delete(path);

DirectoryInfo

- Instance (niet static)
- Aanmaken, verplaatsen, oplijsten (sub)directories
- Gebruik : deze klasse is geassocieerd met een folder en voorziet alle operaties op deze folder
- Initialisatie door path mee te geven aan constructor
- Vb :
  - DirectoryInfo dirinfo=new DirectoryInfo(path);
  - dirinfo.Create();
  - Dirinfo.Delete();

FileInfo versus File

File

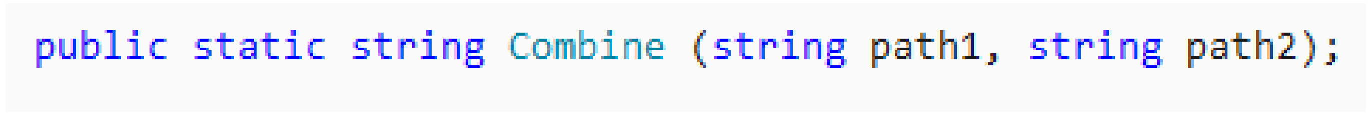
- Static
- Optimaal voor éénmalige operaties
- Aanmaken, verplaatsen, copiëren, …
- Vb :
    - FileStream fs=File.Create(filePath);
    - File.Delete(filePath);

FileInfo

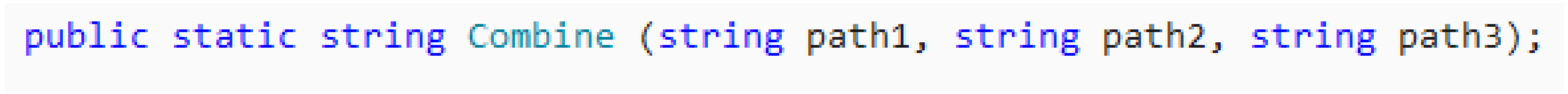
- Instance (niet static)
- Optimaal voor veelvuldige operaties
- Aanmaken, verplaatsen, copiëren, …
- Initialisatie door path mee te geven aan constructor
- Vb :
    - FileInfo finfo=new FileInfo(filePath);
    - StreamWriter str=finfo.CreateText();
    - finfo.Delete();

Path (https://docs.microsoft.com/en-us/dotnet/api/system.io.path?view=netframework-4.8)

- A path is a string that provides the location of a file or directory.
- A path does not necessarily point to a location on disk; for example, a path might map to a location in memory or on a device.
- The exact format of a path is determined by the current platform.



```
public static string Combine (string path1, string path2);
```



```
public static string Combine (string path1, string path2, string path3);
```

DirectoryInfo

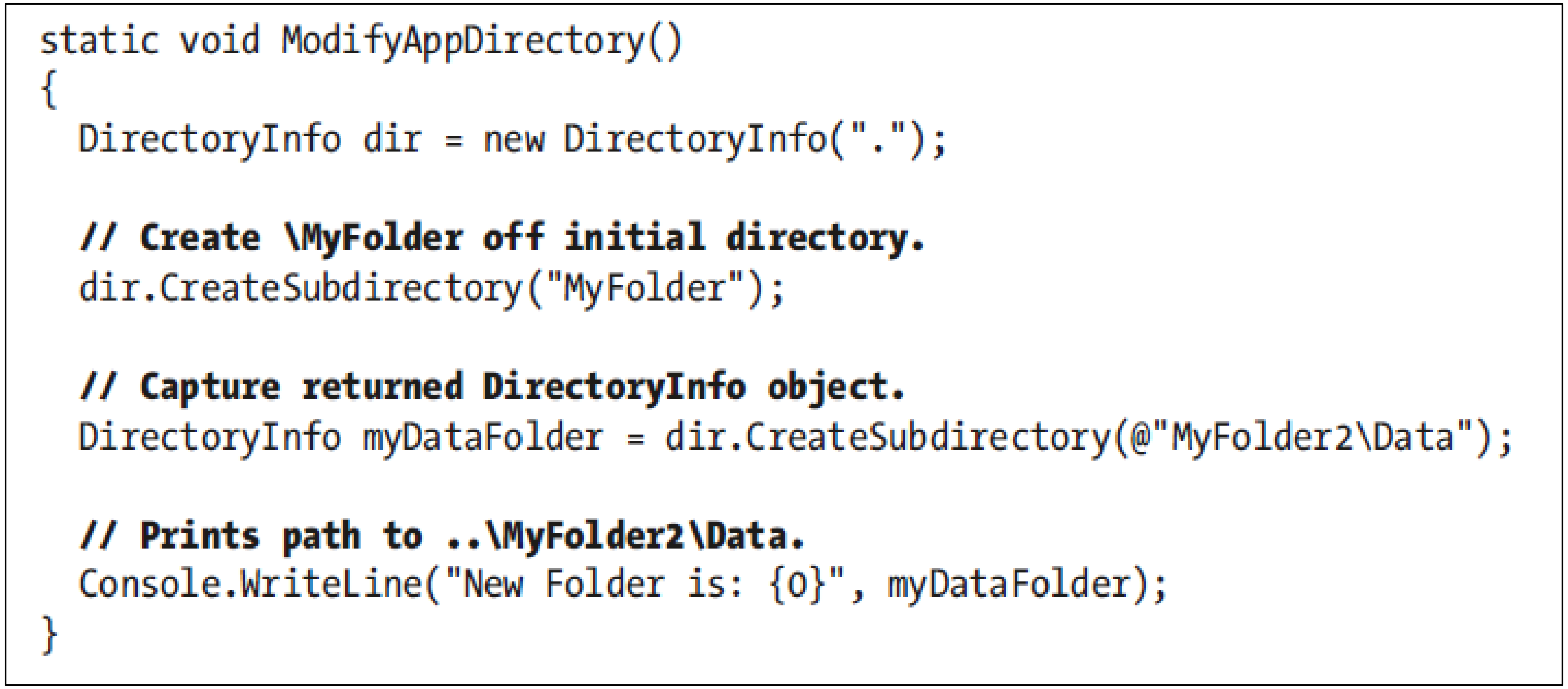
```
class Program
{
  static void Main(string[] args)
  {
    Console.WriteLine("***** Fun with Directory(Info) *****\n");
    ShowWindowsDirectoryInfo();
    Console.ReadLine();
  }

  static void ShowWindowsDirectoryInfo()
  {
    // Dump directory information.
    DirectoryInfo dir = new DirectoryInfo(@"C:\Windows");
    Console.WriteLine("***** Directory Info *****");
    Console.WriteLine("FullName: {0}", dir.FullName);
    Console.WriteLine("Name: {0}", dir.Name);
    Console.WriteLine("Parent: {0}", dir.Parent);
    Console.WriteLine("Creation: {0}", dir.CreationTime);
    Console.WriteLine("Attributes: {0}", dir.Attributes);
    Console.WriteLine("Root: {0}", dir.Root);
    Console.WriteLine("**************************\n");
  }
}
```

```
***** Fun with Directory(Info) *****

***** Directory Info *****
FullName: C:\Windows
Name: Windows
Parent:
Creation: 7/9/2017 8:52:58 AM
Attributes: Directory
Root: C:\
**************************
```

DirectoryInfo - GetFiles

```
static void DisplayImageFiles()
{
  DirectoryInfo dir = new DirectoryInfo(@"C:\Windows\Web\Wallpaper");
  // Get all files with a *.jpg extension.
  FileInfo[] imageFiles = dir.GetFiles("*.jpg", SearchOption.AllDirectories);

  // How many were found?
  Console.WriteLine("Found {0} *.jpg files\n", imageFiles.Length);

  // Now print out info for each file.
  foreach (FileInfo f in imageFiles)
  {
    Console.WriteLine("****************************");
    Console.WriteLine("File name: {0}", f.Name);
    Console.WriteLine("File size: {0}", f.Length);
    Console.WriteLine("Creation: {0}", f.CreationTime);
    Console.WriteLine("Attributes: {0}", f.Attributes);
    Console.WriteLine("***************************\n");
  }
}
```

DirectoryInfo - CreateSubdirectory



```
static void ModifyAppDirectory()
{
  DirectoryInfo dir = new DirectoryInfo(".");

  // Create \MyFolder off initial directory.
  dir.CreateSubdirectory("MyFolder");

  // Capture returned DirectoryInfo object.
  DirectoryInfo myDataFolder = dir.CreateSubdirectory(@"MyFolder2\Data");

  // Prints path to ..\MyFolder2\Data.
  Console.WriteLine("New Folder is: {0}", myDataFolder);
}
```

DriveInfo

```
class Program
{
  static void Main(string[] args)
  {
    Console.WriteLine("***** Fun with DriveInfo *****\n");

    // Get info regarding all drives.
    DriveInfo[] myDrives = DriveInfo.GetDrives();
    // Now print drive stats.
    foreach(DriveInfo d in myDrives)
    {
      Console.WriteLine("Name: {0}", d.Name);
      Console.WriteLine("Type: {0}", d.DriveType);

      // Check to see whether the drive is mounted.
      if(d.IsReady)
      {
        Console.WriteLine("Free space: {0}", d.TotalFreeSpace);
        Console.WriteLine("Format: {0}", d.DriveFormat);
        Console.WriteLine("Label: {0}", d.VolumeLabel);
      }
      Console.WriteLine();
    }
    Console.ReadLine();
  }
}
```

```
***** Fun with DriveInfo *****

Name: C:\
Type: Fixed
Free space: 791699763200
Format: NTFS
Label: Windows10_OS

Name: D:\
Type: Fixed
Free space: 23804067840
Format: NTFS
Label: LENOVO

Press any key to continue . . .
```

FileInfo – Create / Open

```
static void Main(string[] args)
{
  // Make a new file on the C drive.
  FileInfo f = new FileInfo(@"C:\Test.dat");
  FileStream fs = f.Create();

  // Use the FileStream object...

  // Close down file stream.
  fs.Close();
}
```

```
static void Main(string[] args)
{
  // Make a new file via FileInfo.Open().
  FileInfo f2 = new FileInfo(@"C:\Test2.dat");
  using(FileStream fs2 = f2.Open(FileMode.OpenOrCreate,
    FileAccess.ReadWrite, FileShare.None))
  {
    // Use the FileStream object...
  }
}
```

Member	Meaning in Life
`CreateNew`	Informs the OS to make a new file. If it already exists, an `IOException` is thrown.
`Create`	Informs the OS to make a new file. If it already exists, it will be overwritten.
`Open`	Opens an existing file. If the file does not exist, a `FileNotFoundException` is thrown.
`OpenOrCreate`	Opens the file if it exists; otherwise, a new file is created.
`Truncate`	Opens an existing file and truncates the file to 0 bytes in size.
`Append`	Opens a file, moves to the end of the file, and begins write operations (you can use this flag only with a write-only stream). If the file does not exist, a new file is created.

FileInfo – Create / Open

```
static void Main(string[] args)
{
  // Get a FileStream object with read-only permissions.
  FileInfo f3 = new FileInfo(@"C:\Test3.dat");
  using(FileStream readOnlyStream = f3.OpenRead())
  {
    // Use the FileStream object...
  }

  // Now get a FileStream object with write-only permissions.
  FileInfo f4 = new FileInfo(@"C:\Test4.dat");
  using(FileStream writeOnlyStream = f4.OpenWrite())
  {
    // Use the FileStream object...
  }
}
```

```
static void Main(string[] args)
{
  // Get a StreamReader object.
  FileInfo f5 = new FileInfo(@"C:\boot.ini");
  using(StreamReader sreader = f5.OpenText())
  {
    // Use the StreamReader object...
  }
}
```

```
static void Main(string[] args)
{
  FileInfo f6 = new FileInfo(@"C:\Test6.txt");
  using(StreamWriter swriter = f6.CreateText())
  {
    // Use the StreamWriter object...
  }

  FileInfo f7 = new FileInfo(@"C:\FinalTest.txt");
  using(StreamWriter swriterAppend = f7.AppendText())
  {
    // Use the StreamWriter object...
  }
}
```

File – Create / Open

```
// Obtain FileStream object via File.Create().
using(FileStream fs = File.Create(@"C:\Test.dat"))
{}

// Obtain FileStream object via File.Open().
using(FileStream fs2 = File.Open(@"C:\Test2.dat",
  FileMode.OpenOrCreate,
  FileAccess.ReadWrite, FileShare.None))
{}

// Get a FileStream object with read-only permissions.
using(FileStream readOnlyStream = File.OpenRead(@"Test3.dat"))
{}

// Get a FileStream object with write-only permissions.
using(FileStream writeOnlyStream = File.OpenWrite(@"Test4.dat"))
{}

// Get a StreamReader object.
using(StreamReader sreader = File.OpenText(@"C:\boot.ini"))
{}

// Get some StreamWriters.
using(StreamWriter swriter = File.CreateText(@"C:\Test6.txt"))
{}

using(StreamWriter swriterAppend = File.AppendText(@"C:\FinalTest.txt"))
{}
```

File – Read / Write

```
class Program
{
  static void Main(string[] args)
  {
    Console.WriteLine("***** Simple I/O with the File Type *****\n");
    string[] myTasks = {
      "Fix bathroom sink", "Call Dave",
      "Call Mom and Dad", "Play Xbox One"};

    // Write out all data to file on C drive.
    File.WriteAllLines(@"tasks.txt", myTasks);

    // Read it all back and print out.
    foreach (string task in File.ReadAllLines(@"tasks.txt"))
    {
      Console.WriteLine("TODO: {0}", task);
    }
    Console.ReadLine();
  }
}
```

FileStream
read / write bytes

```
// Don't forget to import the System.Text and System.IO namespaces.
static void Main(string[] args)
{
  Console.WriteLine("***** Fun with FileStreams *****\n");

  // Obtain a FileStream object.
  using(FileStream fStream = File.Open(@"myMessage.dat",
    FileMode.Create))
  {
    // Encode a string as an array of bytes.
    string msg = "Hello!";
    byte[] msgAsByteArray = Encoding.Default.GetBytes(msg);

    // Write byte[] to file.
    fStream.Write(msgAsByteArray, 0, msgAsByteArray.Length);

    // Reset internal position of stream.
    fStream.Position = 0;

    // Read the types from file and display to console.
    Console.Write("Your message as an array of bytes: ");
    byte[] bytesFromFile = new byte[msgAsByteArray.Length];
    for (int i = 0; i < msgAsByteArray.Length; i++)
    {
      bytesFromFile[i] = (byte)fStream.ReadByte();
      Console.Write(bytesFromFile[i]);
    }
```

```
    // Display decoded messages.
    Console.Write("\nDecoded Message: ");
    Console.WriteLine(Encoding.Default.GetString(bytesFromFile));
  }
  Console.ReadLine();
}
```

StreamWriter

```
static void Main(string[] args)
{
  Console.WriteLine("***** Fun with StreamWriter / StreamReader *****\n");

  // Get a StreamWriter and write string data.
  using(StreamWriter writer = File.CreateText("reminders.txt"))
  {
    writer.WriteLine("Don't forget Mother's Day this year...");
    writer.WriteLine("Don't forget Father's Day this year...");
    writer.WriteLine("Don't forget these numbers:");
    for(int i = 0; i < 10; i++)
      writer.Write(i + " ");

    // Insert a new line.
    writer.Write(writer.NewLine);
  }

  Console.WriteLine("Created file and wrote some thoughts...");
  Console.ReadLine();
}
```

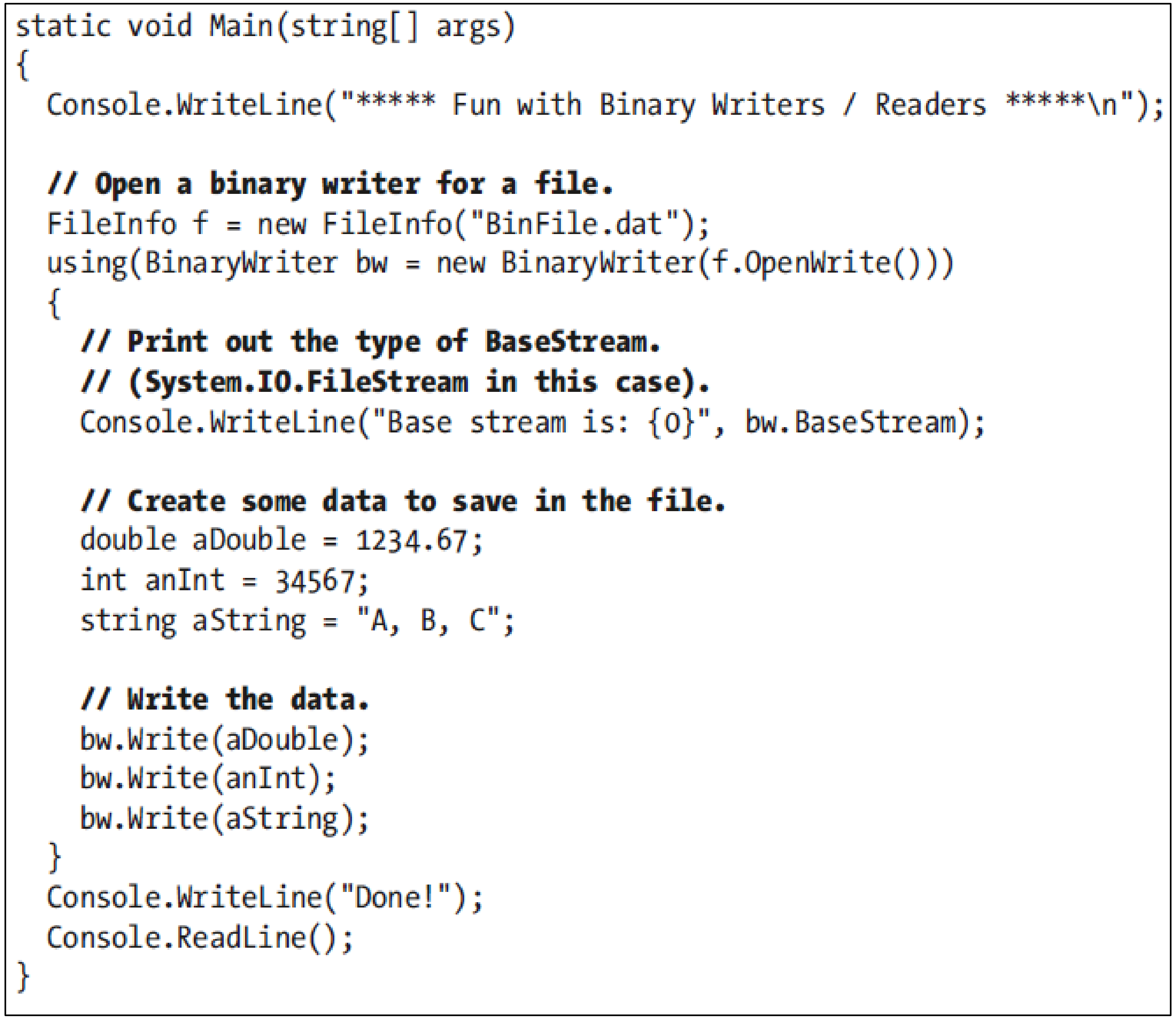
StreamReader

```
static void Main(string[] args)
{
  Console.WriteLine("***** Fun with StreamWriter / StreamReader *****\n");
...
  // Now read data from file.
  Console.WriteLine("Here are your thoughts:\n");
  using(StreamReader sr = File.OpenText("reminders.txt"))
  {
    string input = null;
    while ((input = sr.ReadLine()) != null)
    {
      Console.WriteLine (input);
    }
  }
  Console.ReadLine();
}
```

StreamReader / StreamWriter - constructor

```
static void Main(string[] args)
{
  Console.WriteLine("***** Fun with StreamWriter / StreamReader *****\n");

  // Get a StreamWriter and write string data.
  using(StreamWriter writer = new StreamWriter("reminders.txt"))
  {
    ...
  }

  // Now read data from file.
  using(StreamReader sr = new StreamReader("reminders.txt"))
  {
    ...
  }
}
```

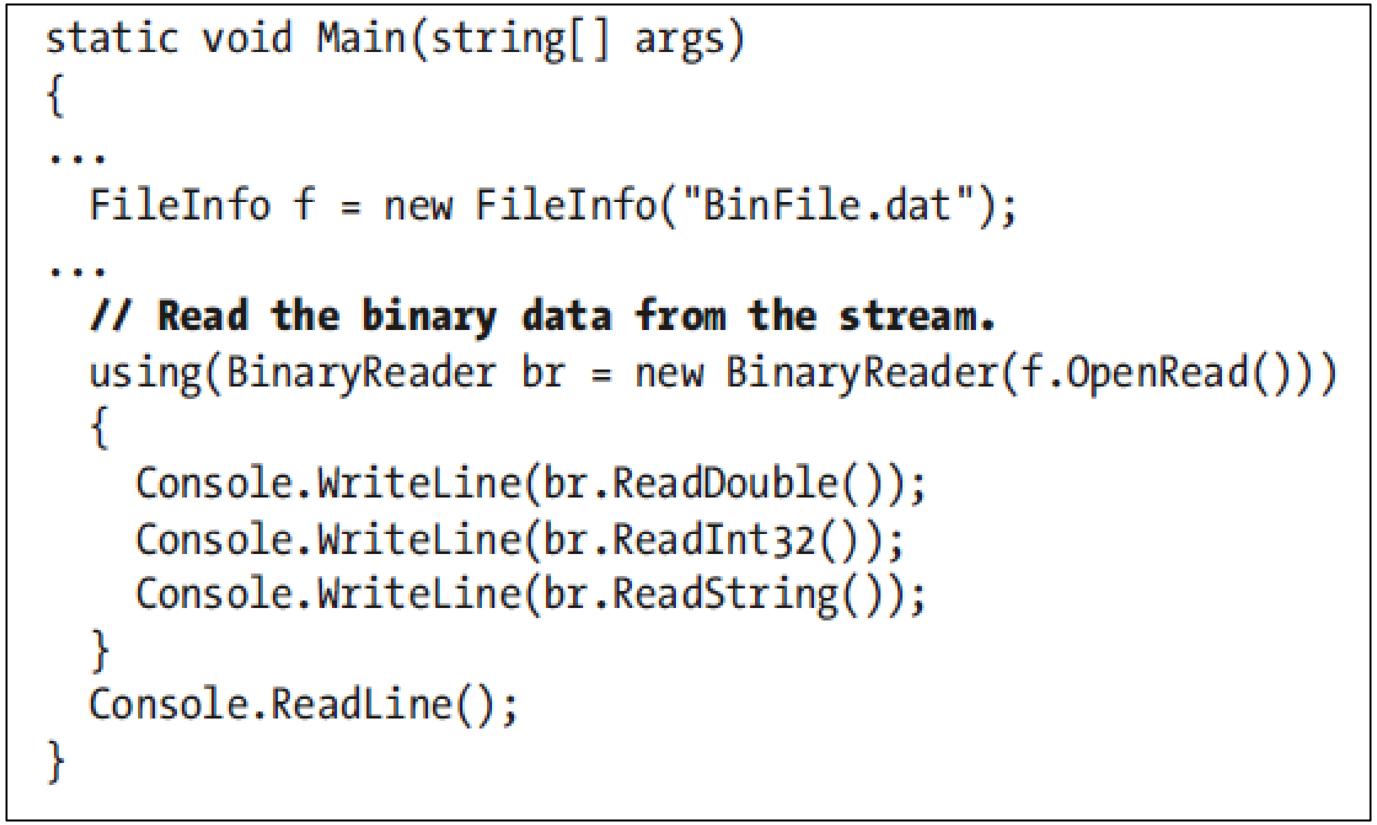
BinaryWriter



```
static void Main(string[] args)
{
  Console.WriteLine("***** Fun with Binary Writers / Readers *****\n");

  // Open a binary writer for a file.
  FileInfo f = new FileInfo("BinFile.dat");
  using(BinaryWriter bw = new BinaryWriter(f.OpenWrite()))
  {
    // Print out the type of BaseStream.
    // (System.IO.FileStream in this case).
    Console.WriteLine("Base stream is: {0}", bw.BaseStream);

    // Create some data to save in the file.
    double aDouble = 1234.67;
    int anInt = 34567;
    string aString = "A, B, C";

    // Write the data.
    bw.Write(aDouble);
    bw.Write(anInt);
    bw.Write(aString);
  }
  Console.WriteLine("Done!");
  Console.ReadLine();
}
```

Filestream als parameter

BinaryReader

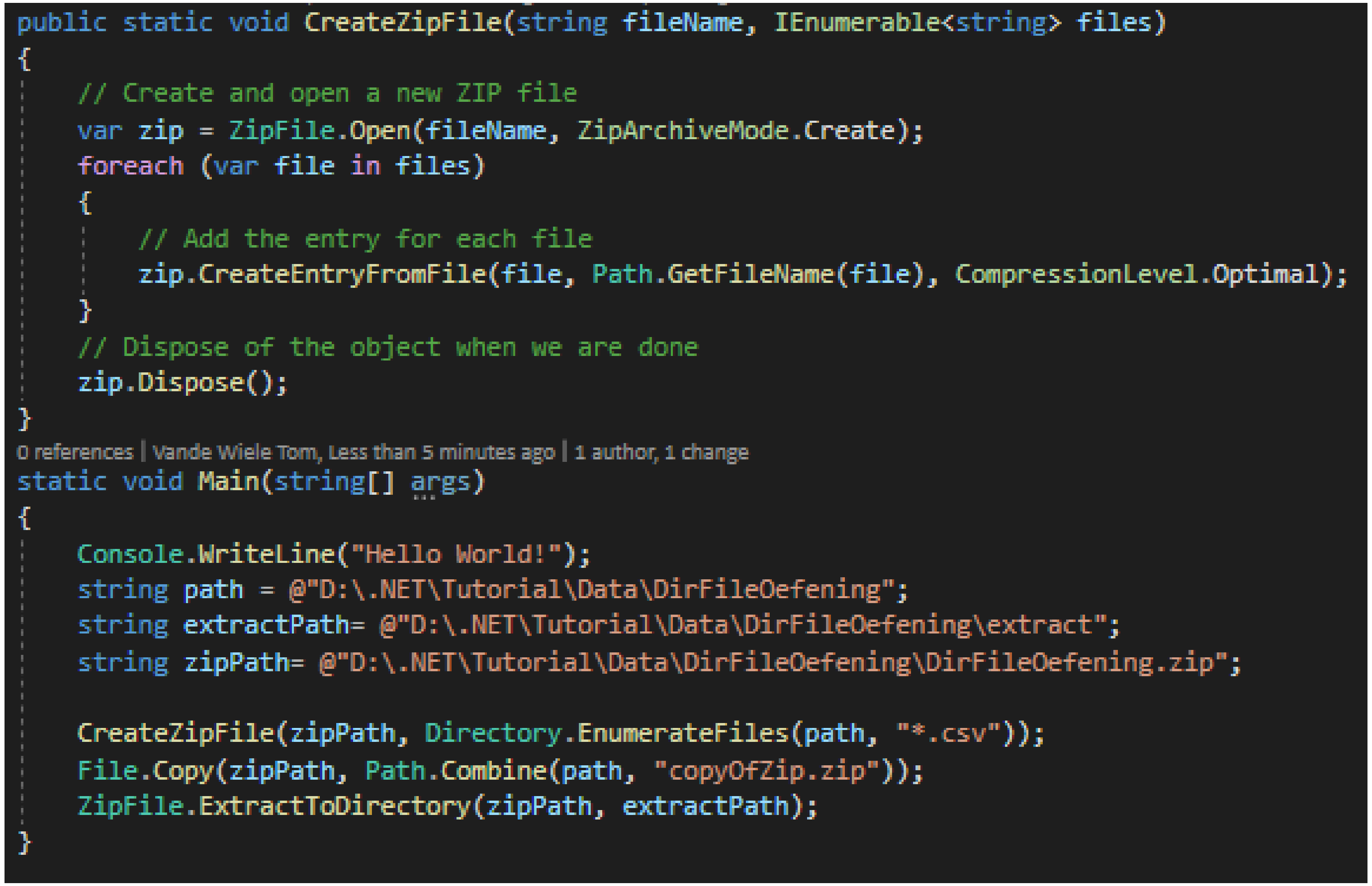


```
static void Main(string[] args)
{
...
  FileInfo f = new FileInfo("BinFile.dat");
...
  // Read the binary data from the stream.
  using(BinaryReader br = new BinaryReader(f.OpenRead()))
  {
    Console.WriteLine(br.ReadDouble());
    Console.WriteLine(br.ReadInt32());
    Console.WriteLine(br.ReadString());
  }
  Console.ReadLine();
}
```

Filestream als parameter

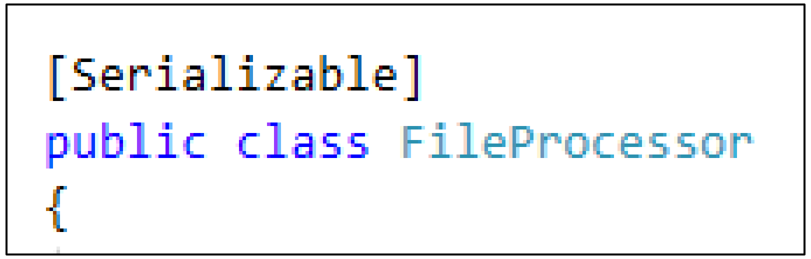
Zip

https://docs.microsoft.com/en-us/dotnet/standard/io/how-to-compress-and-extract-files

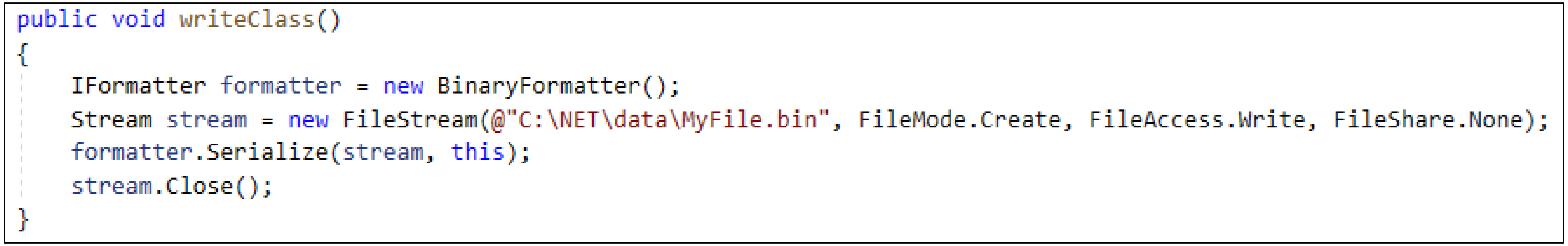


```
public static void CreateZipFile(string fileName, IEnumerable<string> files)
{
    // Create and open a new ZIP file
    var zip = ZipFile.Open(fileName, ZipArchiveMode.Create);
    foreach (var file in files)
    {
        // Add the entry for each file
        zip.CreateEntryFromFile(file, Path.GetFileName(file), CompressionLevel.Optimal);
    }
    // Dispose of the object when we are done
    zip.Dispose();
}
0 references | Vande Wiele Tom, Less than 5 minutes ago | 1 author, 1 change
static void Main(string[] args)
{
    Console.WriteLine("Hello World!");
    string path = @"D:\.NET\Tutorial\Data\DirFileOefening";
    string extractPath= @"D:\.NET\Tutorial\Data\DirFileOefening\extract";
    string zipPath= @"D:\.NET\Tutorial\Data\DirFileOefening\DirFileOefening.zip";

    CreateZipFile(zipPath, Directory.EnumerateFiles(path, "*.csv"));
    File.Copy(zipPath, Path.Combine(path, "copyOfZip.zip"));
    ZipFile.ExtractToDirectory(zipPath, extractPath);
}
```

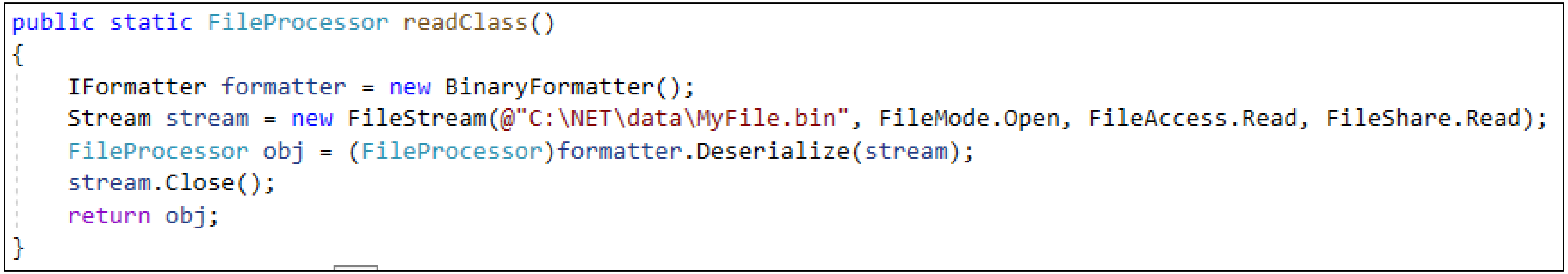
Serialisation



```
[Serializable]
public class FileProcessor
{
```



```
public void writeClass()
{
    IFormatter formatter = new BinaryFormatter();
    Stream stream = new FileStream(@"C:\NET\data\MyFile.bin", FileMode.Create, FileAccess.Write, FileShare.None);
    formatter.Serialize(stream, this);
    stream.Close();
}
```



```
public static FileProcessor readClass()
{
    IFormatter formatter = new BinaryFormatter();
    Stream stream = new FileStream(@"C:\NET\data\MyFile.bin", FileMode.Open, FileAccess.Read, FileShare.Read);
    FileProcessor obj = (FileProcessor)formatter.Deserialize(stream);
    stream.Close();
    return obj;
}
```

https://www.c-sharpcorner.com/article/serializing-objects-in-C-Sharp/