FPT Education
FPT UNIVERSITY

Microsoft® .NET

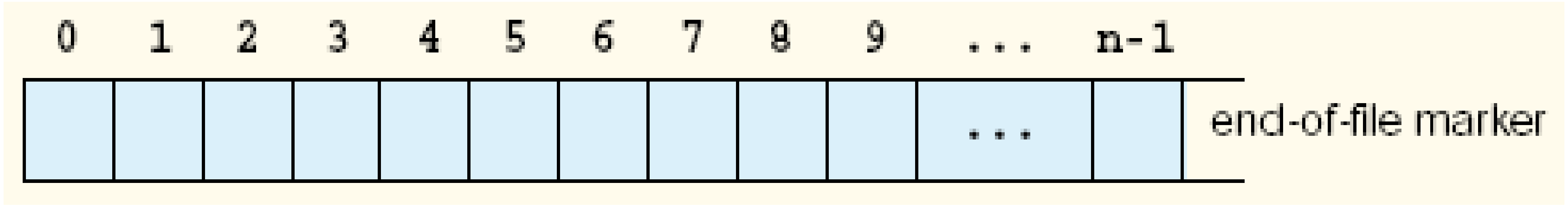
Working with Files and System.IO

Objectives

- Overview System.IO
- Explain about File and FileInfo class
- Explain about Directory and DirectoryInfo class
- Explain and Demo about FileStream class
- Demo about File and FileInfo class
- Demo about Directory and DirectoryInfo class
- Demo create Text file using StreamReader and StreamWriter
- Demo create Binary file using BinaryWriter and BinaryReader

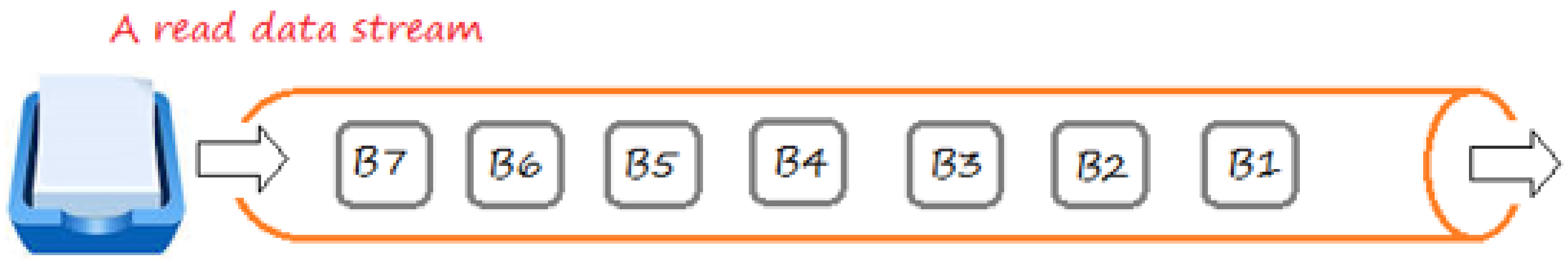
Understanding the Files

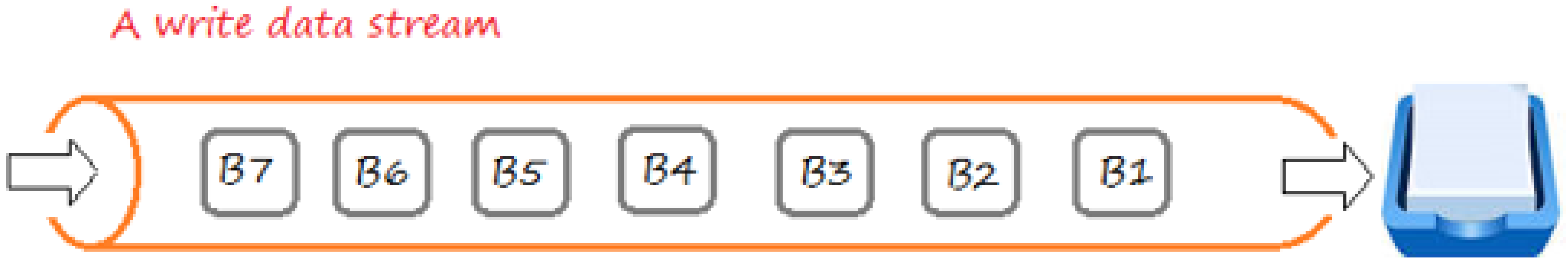
- A file is a collection of bytes stored on a secondary storage device, which is generally a disk of some kind
- A memory location that has a name
- File can be used as an extra memory to store a large amount of data temporarily or permanently
- Each file ends with a marking-end-character or a number of bytes



Understanding the Files

- **Stream**: a chunk of data (sequence of bytes) containing information being passed through to store in memory storage





Exploring the System.IO Namespace

- In the .NET, the **System.IO** namespace is the region of the base class libraries devoted to file-based (and memory-based) input and output (I/O) services
- **System.IO** defines a set of classes, interfaces, enumerations, structures, and delegates, most of which we can find in mscorlib.dll
- The types contained within mscorlib.dll, the System.dll assembly defines additional members of the **System.IO** namespace

Exploring the System.IO Namespace

Class Type	Description
BinaryReader BinaryWriter	These types allow to store and retrieve primitive data types (integers, Booleans, strings, and whatnot) as a binary value
BufferedStream	This type provides temporary storage for a stream of bytes that may be committed to storage at a later time
Directory DirectoryInfo	Use these classes to manipulate a machine’s directory structure. The Directory type exposes functionality using static members, while the DirectoryInfo type exposes similar functionality from a valid object reference
DriveInfo	This class provides detailed information regarding the drives that a given machine uses
File FileInfo	Use these classes to manipulate a machine’s set of files. The File type exposes functionality using static members, while the FileInfo type exposes similar functionality from a valid object reference

Exploring the System.IO Namespace

Class Type	Description
FileStream	This class gives you random file access (e.g., seeking capabilities) with data represented as a stream of bytes
FileSystemWatcher	This class allows you to monitor the modification of external files in a specified directory
MemoryStream	This class provides random access to streamed data stored in memory rather than in a physical file
Path	This class performs operations on System.String types that contain file or directory path information in a platform-neutral manner
StreamWriter StreamReader	You use these classes to store (and retrieve) textual information to (or from) a file. These types do not support random file access
StringWriter StringReader	Like the StreamReader/StreamWriter classes, these classes also work with textual information. However, the underlying storage is a string buffer rather than a physical file

Working with Files and 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

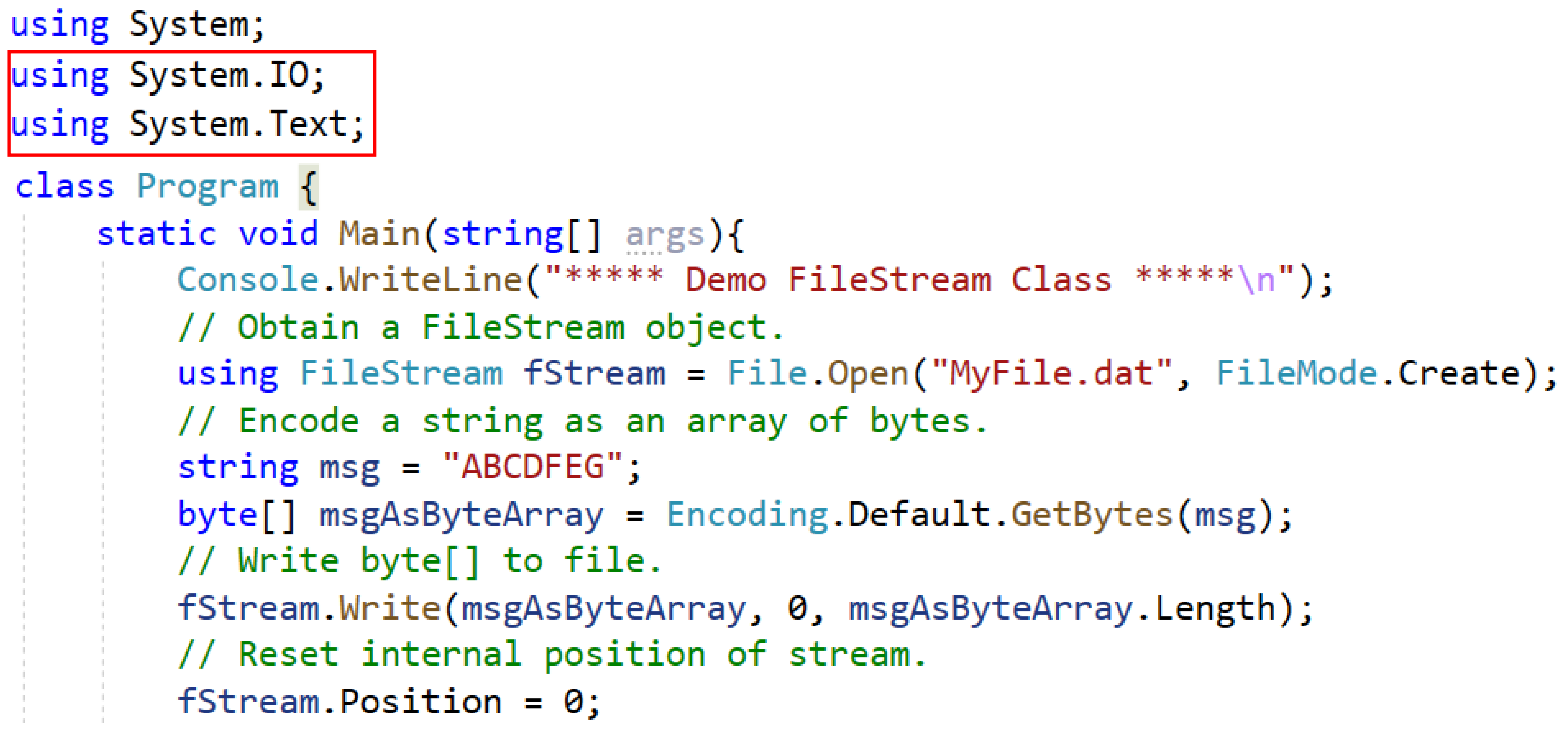
Class Name	Description
File	File is a static class that provides different functionalities like copy, create, move, delete, open for reading or /writing, encrypt or decrypt, check if a file exists, append lines or text to a file's content, get last access time, etc.
FileInfo	The FileInfo class provides the same functionality as a static File class. We 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

Working with Files and Directories

Class Name	Description
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

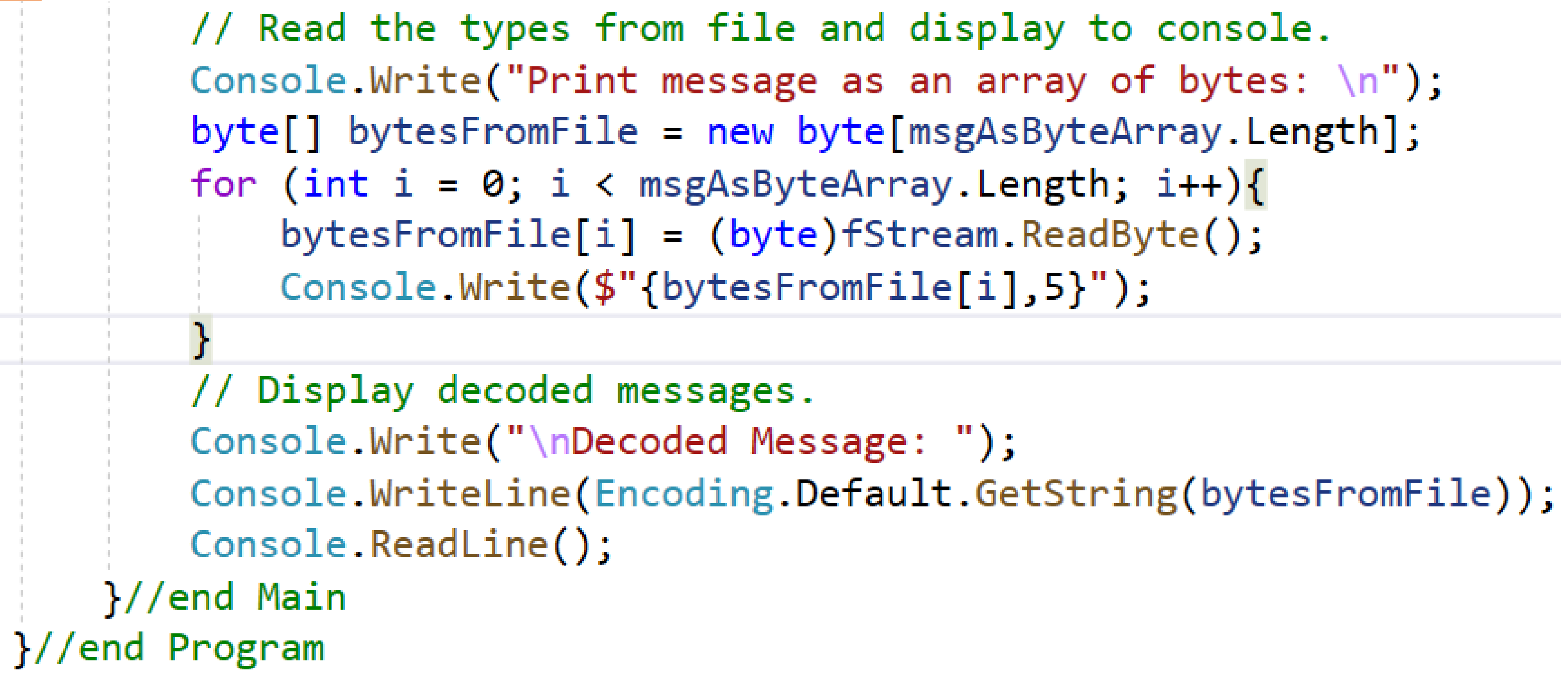
FileStream Class Demonstration

- Provides a **Stream** for a file, supporting both synchronous and asynchronous read and write operations



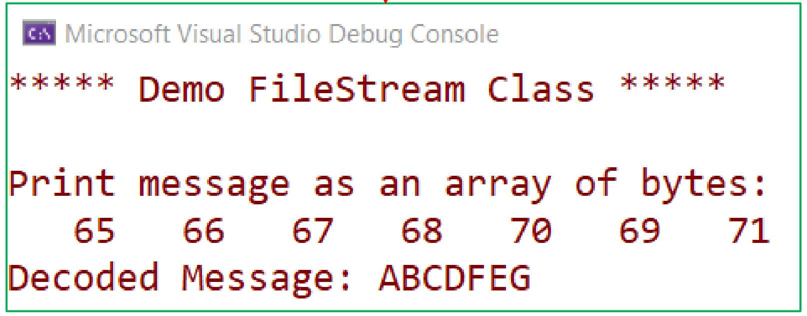
```
using System;
using System.IO;
using System.Text;
class Program {
    static void Main(string[] args){
        Console.WriteLine("***** Demo FileStream Class *****\n");
        // Obtain a FileStream object.
        using FileStream fStream = File.Open("MyFile.dat", FileMode.Create);
        // Encode a string as an array of bytes.
        string msg = "ABCDFEG";
        byte[] msgAsByteArray = Encoding.Default.GetBytes(msg);
        // Write byte[] to file.
        fStream.Write(msgAsByteArray, 0, msgAsByteArray.Length);
        // Reset internal position of stream.
        fStream.Position = 0;
```

FileStream Class Demonstration



```
        // Read the types from file and display to console.
        Console.Write("Print message as an array of bytes: \n");
        byte[] bytesFromFile = new byte[msgAsByteArray.Length];
        for (int i = 0; i < msgAsByteArray.Length; i++){
            bytesFromFile[i] = (byte)fStream.ReadByte();
            Console.Write($"{bytesFromFile[i],5}");
        }
        // Display decoded messages.
        Console.Write("\nDecoded Message: ");
        Console.WriteLine(Encoding.Default.GetString(bytesFromFile));
        Console.ReadLine();
    }//end Main
}//end Program
```

On Windows OS

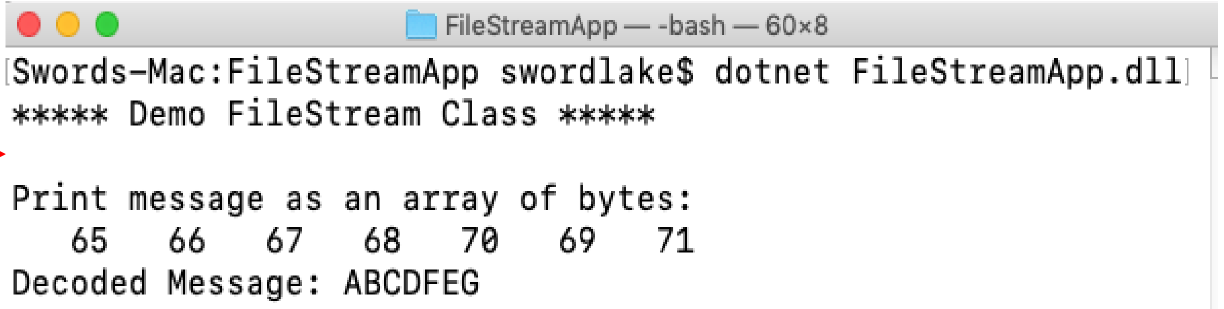


Microsoft Visual Studio Debug Console

```
***** Demo FileStream Class *****

Print message as an array of bytes:
   65   66   67   68   70   69   71
Decoded Message: ABCDFEG
```

On Mac OS



FileStreamApp — -bash — 60×8

```
Swords-Mac:FileStreamApp swordlake$ dotnet FileStreamApp.dll
***** Demo FileStream Class *****

Print message as an array of bytes:
   65   66   67   68   70   69   71
Decoded Message: ABCDFEG
```

Working with File Class

- File is a static class to read\write from physical file with less coding
- Static File class provides functionalities such as create, read\write, copy, move, delete and others for physical files
- 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

Working with File Class

- Important Methods of static File class

Method Name	Description
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

Working with File Class

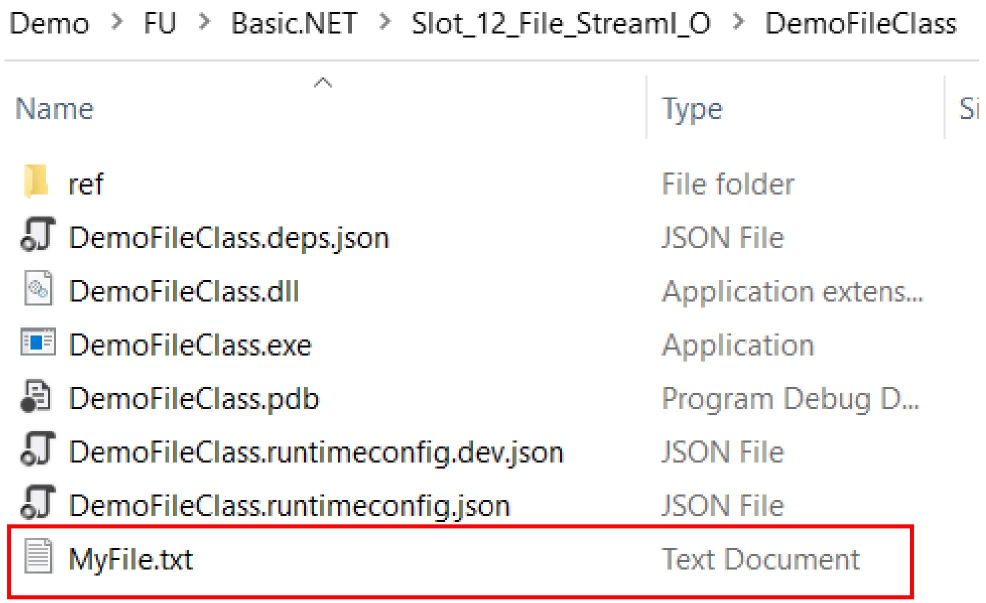
Method Name	Description
Delete	Deletes the specified file
Decrypt	Decrypts a file that was encrypted by the current account using the Encrypt method
Encrypt	Encrypts a file so that only the account used to encrypt the file can decrypt it
Exists	Determines whether the specified file exists
GetAccessControl	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

Working with File Class

Method Name	Description
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

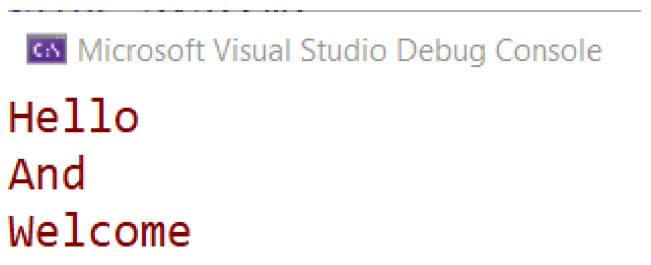
File Class Demonstration-01

```
using System;
using System.IO;
namespace DemoFileClass{
    class Program {
        static void Main(string[] args) {
            string path = @"MyFile.txt";
            if (!File.Exists(path)){
                // Create a file to write to.
                using StreamWriter sw = File.CreateText(path);
                sw.WriteLine("Hello");
                sw.WriteLine("And");
                sw.WriteLine("Welcome");
            }
            // Open the file to read from.
            using StreamReader sr = File.OpenText(path);
            string s;
            while ((s = sr.ReadLine()) != null) {
                Console.WriteLine(s);
            }
        }
    }
}
```



Demo › FU › Basic.NET › Slot_12_File_StreamI_O › DemoFileClass

Name	Type	Si
ref	File folder	
DemoFileClass.deps.json	JSON File	
DemoFileClass.dll	Application extens...	
DemoFileClass.exe	Application	
DemoFileClass.pdb	Program Debug D...	
DemoFileClass.runtimeconfig.dev.json	JSON File	
DemoFileClass.runtimeconfig.json	JSON File	
MyFile.txt	Text Document	

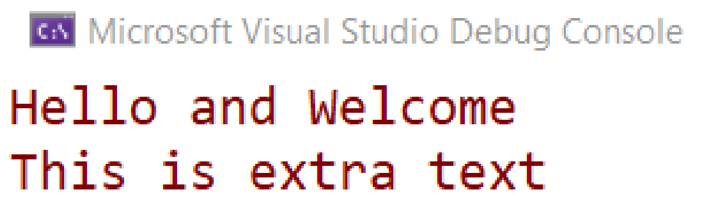


Microsoft Visual Studio Debug Console

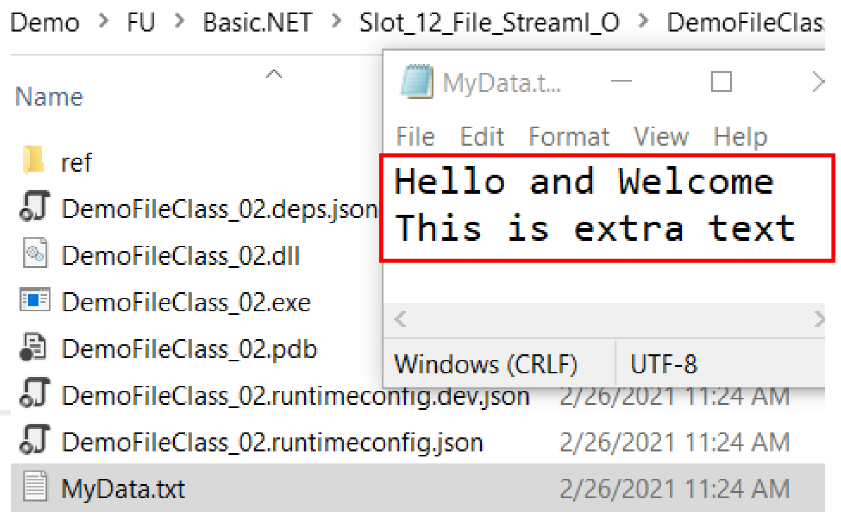
```
Hello
And
Welcome
```

File Class Demonstration-02

```
class Program {
    static void Main(string[] args) {
        string path = @"MyData.txt";
        // This text is added only once to the file.
        if (!File.Exists(path)){
            // Create a file to write to.
            string createText = "Hello and Welcome" + Environment.NewLine;
            File.WriteAllText(path, createText);
        }
        string appendText = "This is extra text" + Environment.NewLine;
        File.AppendAllText(path, appendText);
        // Open the file to read from.
        string readText = File.ReadAllText(path);
        Console.WriteLine(readText);
        Console.ReadLine();
    }
}
```



```
Hello and Welcome
This is extra text
```


Demo › FU › Basic.NET › Slot_12_File_StreamI_O › DemoFileClas
Name
MyData.t...
File Edit Format View Help

```
Hello and Welcome
This is extra text
```

Windows (CRLF) UTF-8

Name	
ref	
DemoFileClass_02.deps.json	
DemoFileClass_02.dll	
DemoFileClass_02.exe	
DemoFileClass_02.pdb	
DemoFileClass_02.runtimeconfig.dev.json	2/26/2021 11:24 AM
DemoFileClass_02.runtimeconfig.json	2/26/2021 11:24 AM
MyData.txt	2/26/2021 11:24 AM

Working with FileInfo Class

- The FileInfo class provides the same functionality as the static **File** class but we have more control on read/write operations on files by writing code manually for reading or writing bytes from a file
- Important Properties and Methods of **FileInfo** class:

Property Name	Description
Directory	Gets an instance of the parent directory
DirectoryName	Gets a string representing the directory's full path
Exists	Gets a value indicating whether a file exists

Working with FileInfo Class

Property Name	Description
Extension	Gets the string representing the extension part of the file
FullName	Gets the full path of the directory or file
IsReadOnly	Gets or sets a value that determines if the current file is read only
LastAccessTime	Gets or sets the time the current file or directory was last accessed
LastWriteTime	Gets or sets the time when the current file or directory was last written to
Length	Gets the size, in bytes, of the current file
Name	Gets the name of the file

Working with FileInfo Class

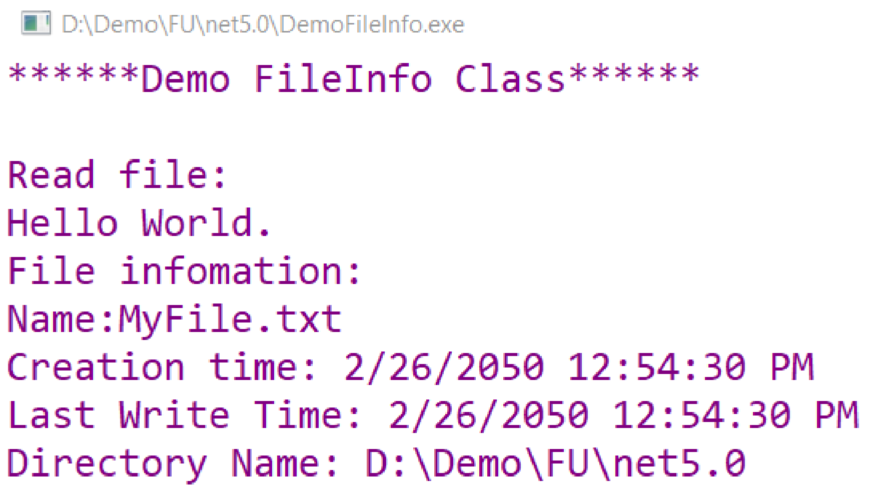
Method Name	Description
AppendText	Creates a StreamWriter that appends text to the file represented by this instance of the FileInfo
CopyTo	Copies an existing file to a new file, disallowing the overwriting of an existing file
Create	Creates a file
CreateText	Creates a StreamWriter that writes a new text file
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
GetAccessControl	Gets a FileSecurity object that encapsulates the access control list (ACL) entries for a specified file

Working with FileInfo Class

Method Name	Description
MoveTo	Moves a specified file to a new location, providing the option to specify a new file name
Open	Opens a in the specified FileMode
OpenRead	Creates a read-only FileStream
OpenText	Creates a StreamReader with UTF8 encoding that reads from an existing text file
OpenWrite	Creates a write-only FileStream
Replace	Replaces the contents of a specified file with the file described by the current FileInfo object, deleting the original file, and creating a backup of the replaced file
ToString	Returns a path as string

FileInfo Class Demonstration

```csharp
static void Main(string[] args) {
    string FileName = @"MyFile.txt";
    Console.WriteLine("******Demo FileInfo Class******\n");
    //Create a Text file
    File.WriteAllText(FileName, "Hello World.");
    //Read file content
    Console.WriteLine("Read file:");
    string content = File.ReadAllText(FileName);
    Console.WriteLine(content);
    Console.WriteLine("File infomation:");
    //Get file information
    FileInfo testFile = new FileInfo(FileName);
    Console.WriteLine($"Name:{testFile.Name}");
    // Creation time.
    Console.WriteLine($"Creation time: {testFile.CreationTime}");
    // Last Write Time
    Console.WriteLine($"Last Write Time: {testFile.LastWriteTime}");
    // Name of parent Directory.
    Console.WriteLine($"Directory Name: {testFile.DirectoryName}");
    Console.ReadLine();
}
```



D:\Demo\FU\net5.0\DemoFileInfo.exe

```
******Demo FileInfo Class******

Read file:
Hello World.
File infomation:
Name:MyFile.txt
Creation time: 2/26/2050 12:54:30 PM
Last Write Time: 2/26/2050 12:54:30 PM
Directory Name: D:\Demo\FU\net5.0
```

Working with Directory Class

- Exposes static methods for creating, moving, and enumerating through directories and subdirectories. This class cannot be inherited
- Important methods of **Directory** class:

Method Name	Description
CreateDirectory(String)	Creates all directories and subdirectories in the specified path unless they already exist.
Delete(String)	Deletes an empty directory from a specified path.
EnumerateDirectories(String)	Returns an enumerable collection of directory full names in a specified path.

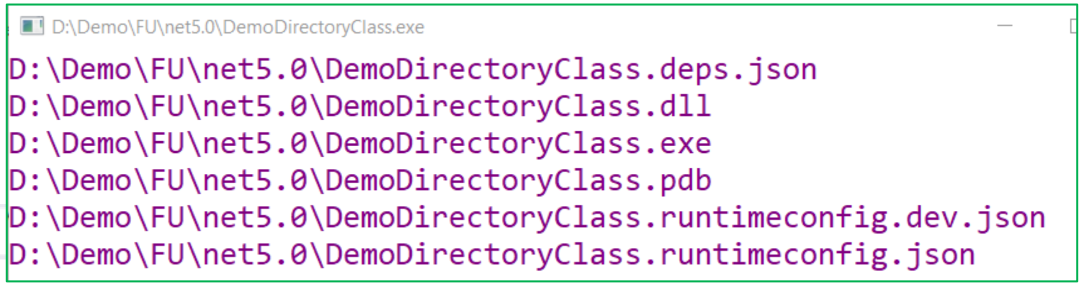
Working with Directory Class

Method Name	Description
EnumerateFiles(String)	Returns an enumerable collection of full file names in a specified path.
Exists(String)	Determines whether the given path refers to an existing directory on disk.
GetCurrentDirectory()	Gets the current working directory of the application.
GetDirectories(String)	Returns the names of subdirectories (including their paths) in the specified directory.
GetFiles(String)	Returns the names of files (including their paths) in the specified directory.
GetParent(String)	Retrieves the parent directory of the specified path, including both absolute and relative paths.
Move(String, String)	Moves a file or a directory and its contents to a new location.

Directory Class Demonstration

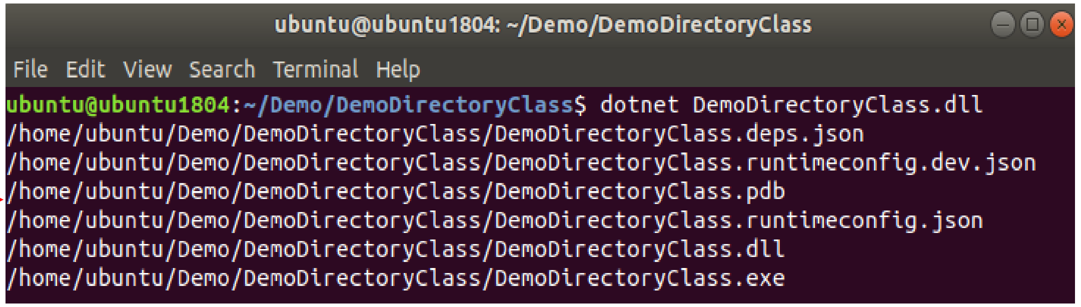
```csharp
class Program {
    static void Main(string[] args){
        //Get current directory
        string sourceDirectory = Directory.GetCurrentDirectory();
        try {
            //Get all files
            var txtFiles = Directory.EnumerateFiles(sourceDirectory, "*.*");
            foreach (string currentFile in txtFiles)  {
                Console.WriteLine(currentFile);
            }
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }
        Console.ReadLine();
    }
}
```

On Windows OS



```
D:\Demo\FU\net5.0\DemoDirectoryClass.exe
D:\Demo\FU\net5.0\DemoDirectoryClass.deps.json
D:\Demo\FU\net5.0\DemoDirectoryClass.dll
D:\Demo\FU\net5.0\DemoDirectoryClass.exe
D:\Demo\FU\net5.0\DemoDirectoryClass.pdb
D:\Demo\FU\net5.0\DemoDirectoryClass.runtimeconfig.dev.json
D:\Demo\FU\net5.0\DemoDirectoryClass.runtimeconfig.json
```

On Linux OS


ubuntu@ubuntu1804: ~/Demo/DemoDirectoryClass
File Edit View Search Terminal Help

```
ubuntu@ubuntu1804:~/Demo/DemoDirectoryClass$ dotnet DemoDirectoryClass.dll
/home/ubuntu/Demo/DemoDirectoryClass/DemoDirectoryClass.deps.json
/home/ubuntu/Demo/DemoDirectoryClass/DemoDirectoryClass.runtimeconfig.dev.json
/home/ubuntu/Demo/DemoDirectoryClass/DemoDirectoryClass.pdb
/home/ubuntu/Demo/DemoDirectoryClass/DemoDirectoryClass.runtimeconfig.json
/home/ubuntu/Demo/DemoDirectoryClass/DemoDirectoryClass.dll
/home/ubuntu/Demo/DemoDirectoryClass/DemoDirectoryClass.exe
```

Working with DirectoryInfo Class

- Exposes instance methods for creating, moving, and enumerating through directories and subdirectories. This class cannot be inherited
- Important Properties and Methods of **DirectoryInfo** class:

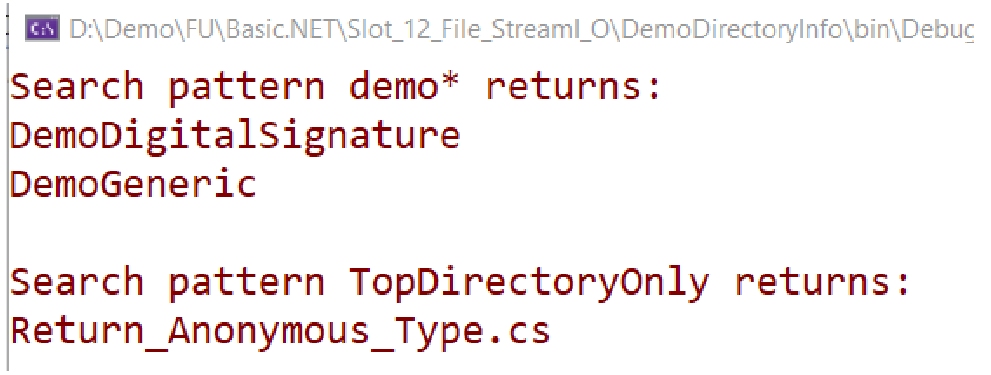
Property Name	Description
Attributes	Gets or sets the attributes for the current file or directory
CreationTime	Gets or sets the creation time of the current file or directory
Exists	Gets a value indicating whether the directory exists
Extension	Gets the string representing the extension part of the file
FullName	Gets the full path of the directory or file

Working with DirectoryInfo Class

Method Name	Description
Create()	Creates a directory
CreateSubdirectory(String)	Creates a subdirectory or subdirectories on the specified path. The specified path can be relative to this instance of the DirectoryInfo class
Delete()	Deletes this DirectoryInfo if it is empty
GetDirectories(String)	Returns an array of directories in the current DirectoryInfo matching the given search criteria
GetFiles(String)	Returns a file list from the current directory matching the given search pattern
MoveTo(String)	Moves a DirectoryInfo instance and its contents to a new path
GetFileSystemInfos(String)	Retrieves an array of strongly typed FileSystemInfo objects representing the files and subdirectories that match the specified search criteria

Working with DirectoryInfo Class Demonstration

```
class Program{
    static void Main(string[] args) {
        DirectoryInfo di = new DirectoryInfo(@"D:\Demo\C#");
        Console.WriteLine("Search pattern demo* returns:");
        foreach (var fi in di.GetDirectories("demo*")){
            Console.WriteLine(fi.Name);
        }
        Console.WriteLine();
        Console.WriteLine("Search pattern TopDirectoryOnly returns:");
        foreach (var fi in di.GetFiles("*.cs", SearchOption.TopDirectoryOnly)){
            Console.WriteLine(fi.Name);
        }
        Console.ReadLine();
    }
}
```

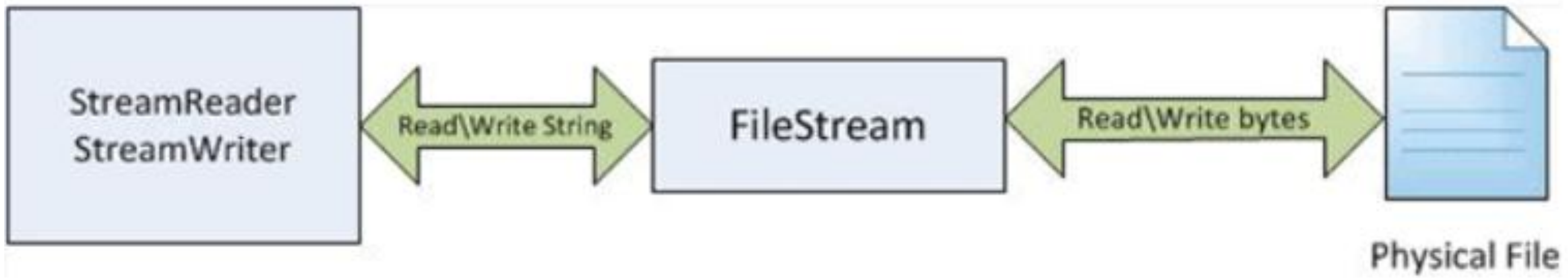

D:\Demo\FU\Basic.NET\Slot_12_File_StreamI_O\DemoDirectoryInfo\bin\Debug

```
Search pattern demo* returns:
DemoDigitalSignature
DemoGeneric

Search pattern TopDirectoryOnly returns:
Return_Anonymous_Type.cs
```

Working with StreamWriter and StreamReader

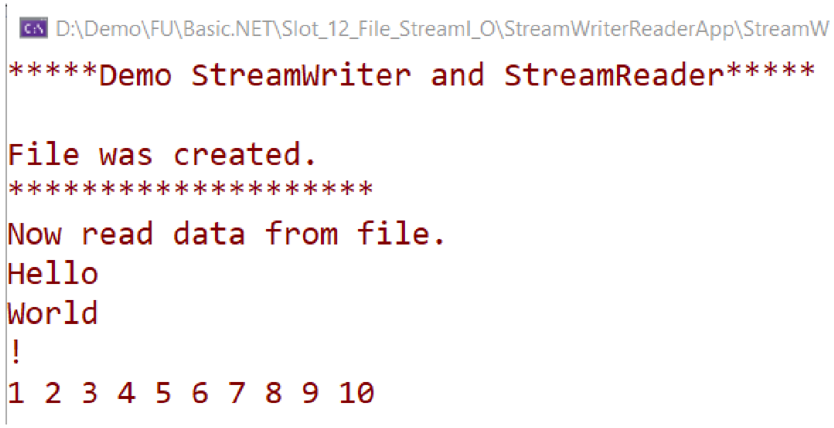
- **StreamReader**: StreamReader is a helper class for reading characters from a Stream by converting bytes into characters using an encoded value. It can be used to read strings (characters) from different Streams like FileStream, MemoryStream, etc
- **StreamWriter**: StreamWriter is a helper class for writing a string to a Stream by converting characters into bytes. It can be used to write strings to different Streams such as FileStream, MemoryStream, etc



Stream IO

StreamWriter and StreamReader Demonstration

```
class Program{
    static void Main(string[] args){
        string input = null;
        string fileName = @"MyData.txt";
        Console.WriteLine("*****Demo StreamWriter and StreamReader*****\n");
        // Get a StreamWriter and write string data.
        using StreamWriter writer = new StreamWriter(fileName);
        writer.WriteLine("Hello");
        writer.WriteLine("World");
        writer.WriteLine("!");
        for (int i = 1; i <= 10; i++) {
            writer.Write(i + " ");
        }
        // Insert a new line.
        writer.Write(writer.NewLine);
        writer.Close();
        Console.WriteLine("File was created.");
```



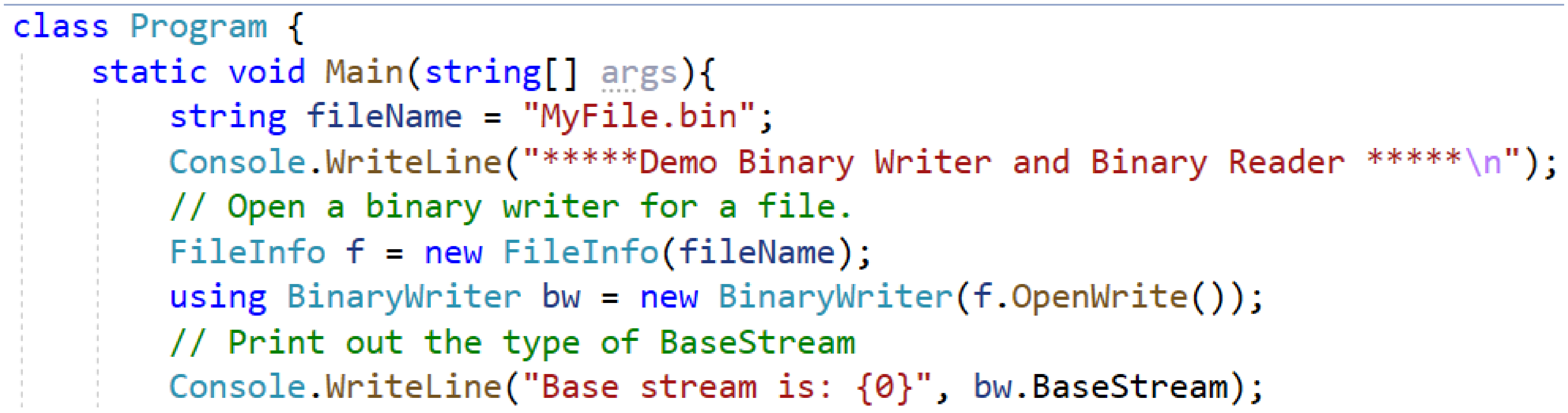
```
D:\Demo\FU\Basic.NET\Slot_12_File_StreamI_O\StreamWriterReaderApp\StreamW
*****Demo StreamWriter and StreamReader*****

File was created.
*******************
Now read data from file.
Hello
World
!
1 2 3 4 5 6 7 8 9 10
```

```
        Console.WriteLine("*******************");
        Console.WriteLine("Now read data from file.");
        using StreamReader sr = new StreamReader(fileName);
        while ((input = sr.ReadLine()) != null){
            Console.WriteLine(input);
        }
        sr.Close();
        Console.ReadLine();
    }//end Main
}//end Program
```

Working with BinaryWriter and BinaryReader

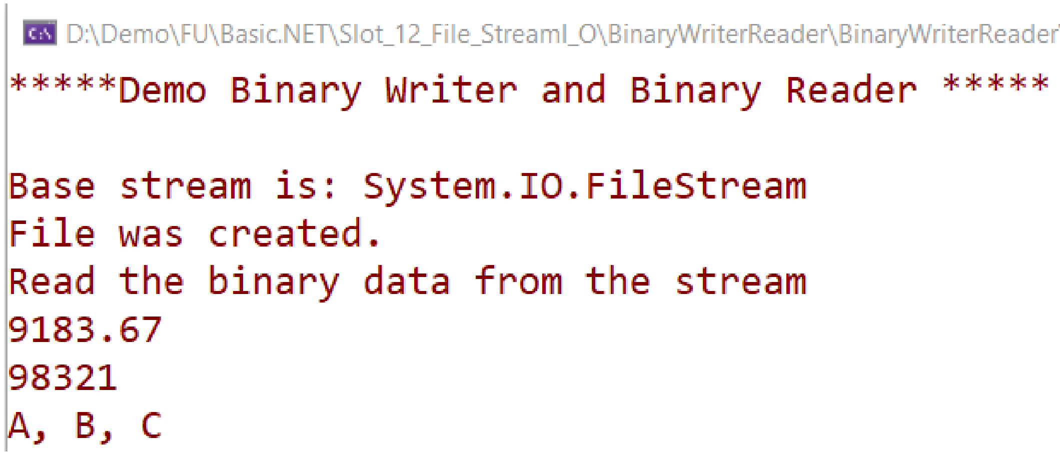
- **BinaryReader** and **BinaryWriter** allow to read and write discrete data types to an underlying stream in a compact binary format



```
class Program {
    static void Main(string[] args){
        string fileName = "MyFile.bin";
        Console.WriteLine("*****Demo Binary Writer and Binary Reader *****\n");
        // Open a binary writer for a file.
        FileInfo f = new FileInfo(fileName);
        using BinaryWriter bw = new BinaryWriter(f.OpenWrite());
        // Print out the type of BaseStream
        Console.WriteLine("Base stream is: {0}", bw.BaseStream);
```

BinaryWriter and BinaryReader Demonstration

```
        // Create some data to save in the file
        double aDouble = 9183.67;
        int anInt = 98321;
        string aString = "A, B, C";
        // Write the data
        bw.Write(aDouble);
        bw.Write(anInt);
        bw.Write(aString);
        bw.Close();
        Console.WriteLine("File was created.");
        Console.WriteLine("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();
    }//end Main
}//end Program
```



D:\Demo\FU\Basic.NET\Slot_12_File_StreamI_O\BinaryWriterReader\BinaryWriterReader

```
*****Demo Binary Writer and Binary Reader *****

Base stream is: System.IO.FileStream
File was created.
Read the binary data from the stream
9183.67
98321
A, B, C
```

Summary

- Concepts were introduced:
  - Overview about System.IO
  - Explain and Demo about FileStream class
  - Explain about File and FileInfo class
  - Explain about Directory and Directory Info class
  - Demo about File and FileInfo class
  - Demo about Directory and DirectoryInfo class
  - Demo create Text file using StreamReader and StreamWriter
  - Demo create Binary file using BinaryWriter and BinaryReader

Discuss - Overview about System.IO

- What is System.IO?
  - Explain the overall purpose of the System.IO namespace in .NET. What kind of operations does it support?
- Key Classes in System.IO
  - What are some of the main classes provided by System.IO? Provide a brief description of each.
- Common Use Cases
  - What are some common use cases for System.IO in software development? Have you used System.IO in any of your projects?
- Streams in System.IO
  - Compare and contrast the Stream and FileStream classes in System.IO. When would you use each one?
- Synchronous vs. Asynchronous Operations
  - Discuss the differences between synchronous and asynchronous methods in System.IO. What are the benefits and drawbacks of each approach?

Discuss – FileStream class

- Understanding FileStream
  - What is the FileStream class in C#? How does it differ from other classes in System.IO?
- Reading from a File
  - Write a C# code snippet to read the contents of a text file using FileStream. Explain how the code works.
- Writing to a File
  - Write a C# code snippet to write data to a binary file using FileStream. Explain how the code works.
- Key Properties and Methods
  - Discuss the key properties and methods of FileStream such as Read, Write, Seek, and Flush.
- When to Use FileStream
  - In what scenarios would you prefer using FileStream over other methods like File.ReadAllText or File.WriteAllText?

Discuss - File and FileInfo class

- File vs. FileInfo
    - Explain the difference between the File and FileInfo classes in System.IO. When should you use File and when should you use FileInfo?
- Using the File Class
    - Write a C# code snippet that uses the File class to check if a file exists and delete it if it does.
- Using the FileInfo Class
    - Write a C# code snippet that uses the FileInfo class to create a new file and write a string to it.
- Static Methods of File
    - Discuss the static methods of the File class such as File.Create, File.Delete, File.Copy, and File.Move. Provide examples of how to use each method.
- Instance Methods of FileInfo
    - Discuss the instance methods of the FileInfo class such as Create, Delete, CopyTo, and MoveTo. Provide examples of how to use each method.

Discuss - Directory and DirectoryInfo class

- Directory vs. DirectoryInfo
  - Explain the difference between the Directory and DirectoryInfo classes in System.IO. When should you use Directory and when should you use DirectoryInfo?
- Using the Directory Class
  - Write a C# code snippet that uses the Directory class to create a new directory and list all files within it.
- Using the DirectoryInfo Class
  - Write a C# code snippet that uses the DirectoryInfo class to check if a directory exists and delete it if it does.
- Static Methods of Directory
  - Discuss the static methods of the Directory class such as CreateDirectory, Delete, Exists, GetFiles, and GetDirectories. Provide examples of how to use each method.
- Instance Methods of DirectoryInfo
  - Discuss the instance methods of the DirectoryInfo class such as Create, Delete, GetFiles, and GetDirectories. Provide examples of how to use each method.