VTC Academy
Thinking Ahead, Beyond Boundaries

Object Oriented Programming

Files I/O

Session 10

Objectives

- File Streams
- File System

Files and Streams

- A **file** is a collection of data stored in a disk with a specific name and a directory path. When a file is opened for reading or writing, it becomes a **stream**.
- The stream is basically the sequence of bytes passing through the communication path
- There are two main streams:
  - **Input stream:** used for reading data from file (read operation)
  - **Output stream:** used for writing into the file (write operation)

Common System.IO Classes

Classes	Description
BinaryReader	Reads primitive data from a binary stream
BinaryWriter	Writes primitive data in binary format
BufferedStream	A temporary storage for a stream of bytes
Directory	Helps in manipulating a directory structure
DirectoryInfo	Used for performing operations on directories
File	Helps in manipulating files
FileInfo	Used for performing operations on files
FileStream	Used to read from and write to any location in a file
MemoryStream	Used for random access to streamed data stored in memory
StreamReader	Used for reading characters from a byte stream
StreamWriter	Is used for writing characters to a stream
StringReader	Is used for reading from a string buffer
StringWriter	Is used for writing into a string buffer

FileStream Class

- The **FileStream** class in the System.IO namespace helps in reading from, writing to and closing files. This class derives from the abstract class Stream.
- You need to create a **FileStream** object to create a new file or open an existing file
- For example, we create a FileStream object **F** for reading a file named **sample.txt** as shown:

```
FileStream F = new FileStream("sample.txt", FileMode.Open,
                             FileAccess.Read, FileShare.Read);
```

FileStream Example

```
using System;
using System.IO;

namespace FileIOApplication
{
   class Program
   {
      static void Main(string[] args)
         FileStream F = new FileStream("sample.dat", FileMode.OpenOrCreate,
                                                     FileAccess.ReadWrite);
         for (int i = 1; i <= 20; i++)
         {
            F.WriteByte((byte)i);
         }

         F.Position = 0;
         for (int i = 0; i <= 20; i++)
         {
            Console.Write(F.ReadByte() + " ");
         }
         F.Close();
         Console.ReadKey();
      }
   }
}
```

Reading from & Writing into Binary File

- The **BinaryReader** and **BinaryWriter** classes are used for reading from and writing to a binary file
- **BinaryReader** class is used to read binary data from a file.
    - A **BinaryReader** object is created by passing a **FileStream** object to its constructor
- **BinaryWriter** class is used to write binary data to a stream.
    - A BinaryWriter object is created by passing a FileStream object to its constructor

Methods of BinaryReader

No.	Methods
1	**public override void Close()** It closes the BinaryReader object and the underlying stream.
2	**public virtual int Read()** Reads the characters from the underlying stream
4	**public virtual byte ReadByte()** Reads the next byte from the current stream
6	**public virtual char ReadChar()** Reads the next character from the current stream
8	**public virtual double ReadDouble()** Reads an 8-byte floating point value from the current stream
9	**public virtual int ReadInt32()** Reads a 4-byte signed integer from the current stream
10	**public virtual string ReadString()** Reads a string from the current stream

Methods of BinaryWriter

No.	Functions
1	**public override void Close()** It closes the BinaryWriter object and the underlying stream
2	**public virtual void Flush()** Clears all buffers for the current writer and causes any buffered data to be written to the underlying device
3	**public virtual long Seek(int offset, SeekOrigin origin)** Sets the position within the current stream
5	**public virtual void Write(byte value)** Writes an unsigned byte to the current stream
7	**public virtual void Write(char ch)** Writes a Unicode character to the current stream
9	**public virtual void Write(double value)** Writes an eight-byte floating-point value to the current stream
10	**public virtual void Write(int value)** Writes a four-byte signed integer to the current stream
11	**public virtual void Write(string value)** Writes a length-prefixed string to this stream in the current encoding of BinaryWriter

BinaryReader & BinaryWriter Example

```
using System;
using System.IO;
namespace BinaryFileApplication {
   class Program {
      static void Main(string[] args) {
         BinaryWriter bw;
         BinaryReader br;
         int i = 25;
         double d = 3.14157;
         bool b = true;
         string s = "I am happy!;
         //create the file
         try {
            bw = new BinaryWriter(new FileStream("mydata", FileMode.Create));
         } catch (IOException e){
            Console.WriteLine(e.Message + "\n Cannot create file.");
            return;
         }
         //writing into the file
         try {
            bw.Write(i);
            bw.Write(d);
            bw.Write(b);
            bw.Write(s);
         } catch (IOException e) {
            Console.WriteLine(e.Message + "\n Cannot write to file.");
            return;
         }
         bw.Close();
         //reading from the file
         try {
            br = new BinaryReader(new FileStream("mydata", FileMode.Open));
         } catch (IOException e) {
            Console.WriteLine(e.Message + "\n Cannot open file.");
            return;
         }
         tr {
            i = br.ReadInt32();
            Console.WriteLine("Integer data: {0}", i);
            d = br.ReadDouble();
            Console.WriteLine("Double data: {0}", d);
            b = br.ReadBoolean();
            Console.WriteLine("Boolean data: {0}", b);
            s = br.ReadString();
            Console.WriteLine("String data: {0}", s);
         } catch (IOException e) {
            Console.WriteLine(e.Message + "\n Cannot read from file.");
            return;
         }
         br.Close();
         Console.ReadKey();
      }
   }
```

Reading from and Writing to Text File

- **StreamReader** and **StreamWriter** classes are used for reading from and writing data to text files. These classes inherit from the abstract base class Stream, which supports reading and writing bytes into a file stream.
- **StreamReader** class also inherits from the abstract base class TextReader that represents a reader for reading series of characters
- **StreamWriter** class inherits from the abstract class TextWriter that represents a writer, which can write a series of character

Methods of StreamReader Class

No.	Methods
1	**public override void Close()** It closes the StreamReader object and the underlying stream, and releases any system resources associated with the reader
2	**public override int Peek()** Returns the next available character but does not consume it
3	**public override int Read()** Reads the next character from the input stream and advances the character position by one

StreamReader Example

```
using System;
using System.IO;
namespace StreamReaderDemo {
   class Program {
      static void Main(string[] args) {
         try
         {
            // create an instance of StreamReader to read from a file
            // the using statement also closes the StreamReader
            using (StreamReader sr = new StreamReader("sample.txt"))
            {
               string line;
               // read lines from the file until the end of the file is reached
               while ((line = sr.ReadLine()) != null)
               {
                  Console.WriteLine(line);
               }
            }
         }
         catch (Exception e)
         {
            Console.WriteLine("The file could not be read!”);
         }
         Console.ReadKey();
      }
   }
}
```

Methods of StreamWriter Class

No.	Methods
1	**public override void Close()** It closes the StreamReader object and the underlying stream, and releases any system resources associated with the reader
2	**public override int Peek()** Returns the next available character but does not consume it
3	**public override int Read()** Reads the next character from the input stream and advances the character position by one

StreamWriter Example

```
using System;
using System.IO;
namespace StreamWriterDemo {
   class Program {
      static void Main(string[] args) {
         string[] names = new string[] {“Steve Jobs", “Bill Gates"};
         using (StreamWriter sw = new StreamWriter("names.txt"))
         {
            foreach (string s in names)
            {
               sw.WriteLine(s);
            }
         }
         // read and show each line from the file
         string line = "";
         using (StreamReader sr = new StreamReader("names.txt"))
         {
            while ((line = sr.ReadLine()) != null)
            {
               Console.WriteLine(line);
            }
         }
         Console.ReadKey();
      }
   }
}
```

Working with File System

- C# allows you to work with the directories and files using various directory and file related classes such as the **DirectoryInfo** class and the **FileInfo** class

- **DirectoryInfo** class is derived from the **FileSystemInfo** class. It has various methods for creating, moving, and browsing through directories and subdirectories. This class cannot be inherited.

- **FileInfo** class is derived from the **FileSystemInfo** class. It has properties and instance methods for creating, copying, deleting, moving, and opening of files, and helps in the creation of FileStream objects. This class cannot be inherited.

Properties of DirectoryInfo Class

Sr.No.	Properties
1	**Attributes** Gets the attributes for the current file or directory
2	**CreationTime** Gets the creation time of the current file or directory
3	**Exists** Gets a Boolean value indicating whether the directory exists
4	**Extension** Gets the string representing the file extension
5	**FullName** Gets the full path of the directory or file
6	**LastAccessTime** Gets the time the current file or directory was last accessed
7	**Name** Gets the name of this DirectoryInfo instance

Methods of DirectoryInfo Class

No.	Methods
1	**public void Create()** Creates a directory
2	**public DirectoryInfo CreateSubdirectory(string path)** Creates a subdirectory or subdirectories on the specified path. The specified path can be relative to this instance of the DirectoryInfo class.
3	**public override void Delete()** Deletes this DirectoryInfo if it is empty
4	**public DirectoryInfo[] GetDirectories()** Returns the subdirectories of the current directory
5	**public FileInfo[] GetFiles()** Returns a file list from the current directory

Properties of FileInfo Class

No.	Properties
1	**Attributes** Gets the attributes for the current file
2	**CreationTime** Gets the creation time of the current file
3	**Directory** Gets an instance of the directory which the file belongs to
4	**Exists** Gets a Boolean value indicating whether the file exists
5	**Extension** Gets the string representing the file extension
7	**Length** Gets the size, in bytes, of the current file
8	**Name** Gets the name of the file.

Methods of FileInfo Class

Sr.No.	Methods
1	**public FileStream Create()** Creates a file
2	**public override void Delete()** Deletes a file permanently
3	**public FileStream Open(FileMode mode)** Opens a file in the specified mode
4	**public FileStream Open(FileMode mode, FileAccess access)** Opens a file in the specified mode with read, write, or read/write access
5	**public FileStream Open(FileMode mode, FileAccess access, FileShare share)** Opens a file in the specified mode with read, write, or read/write access and the specified sharing option
6	**public FileStream OpenRead()** Creates a read-only FileStream
7	**public FileStream OpenWrite()** Creates a write-only FileStream

File System Example

```
using System;
using System.IO;
namespace WindowsFileDemo
{
   class Program
   {
      static void Main(string[] args)
      {
         //creating a DirectoryInfo object
         DirectoryInfo mydir = new DirectoryInfo(@"C:\Windows");

         // getting the files in the directory, their names and size
         FileInfo [] f = mydir.GetFiles();
         foreach (FileInfo file in f)
         {
            Console.WriteLine("File Name: {0} Size: {1}", file.Name,
                                                        file.Length);
         }
         Console.ReadKey();
      }
   }
}
```

Summary

- A **file** is a collection of data stored in a disk with a specific name and a directory path. When a file is opened for reading or writing, it becomes a **stream**.
- The **FileStream** class in the System.IO namespace helps in reading from, writing to and closing files. This class derives from the abstract class Stream.
- The **BinaryReader** and **BinaryWriter** classes are used for reading from and writing to a binary file
- **StreamReader** and **StreamWriter** classes are used for reading from and writing data to text files.
- C# allows you to work with the directories and files using various directory and file related classes such as the **DirectoryInfo** class and the **FileInfo** class