Streams

Handling Files

Streams represent a source that provides a sequence of bytes, such as a file, an input/output (I/O) device, an interprocess communication pipe, or a TCP/IP socket (Harwani, 2015, p. 454).

Two (2) Types of Streams

- **Output streams** – It refers to a printer, remote server location, or a printer where the data is written.
- **Input streams** – It refers to a file or any source where the data can be read and assigned to the memory variables in the program.

In C#, the **`System.IO`** namespace contains classes that allow to write and read files. This namespace needs to be included in all programs that use streams. All files are byte-oriented, meaning the data is written or read into the files in terms of bytes.

FileStream

It is a class that is used to create a byte-oriented stream attached to a file. The code to be used for creating a FileStream object is

```
FileStream(string filename,FileMode mode)
```

The *mode* parameter specifies how the file needs to be opened.

The options for `FileMode` are as follows:

- `FileMode.Create` – It creates a new output file, which will be overwritten if a file already exists.
- `FileMode.CreateNew` – It creates a new output file that is not existing.
- `FileMode.Open` – It opens an existing file.
- `FileMode.OpenOrCreate` – It opens an existing. If not, it creates a new one.
- `FileMode.Truncate` – It opens an existing file and truncates the content that already exists.

The `FileStream` constructor is used to open a file that has access to read or write. This class also contains the following methods to perform read and write operations:

- `ReadByte()` – This method reads a single byte from a file and returns as an integer value.
- `Read()` – This method reads the specified number of bytes from a file into an array.
- `WriteByte()` – It writes the specified byte into the file.
- `Write()` – It writes an array of bytes into the file.
- `Flush()` – This method instantly writes the data into the file.
- `void Close()` – This method closes the file, releasing the system resources that are allocated to it.

Reading Files

To access a file randomly, position the file pointer inside the file at the required location. A file pointer determines the location of the next read/write operation to take place on the file. The method used to relocate the file pointer in the file is **Seek** (Harwani, 2015, p. 462). **`Seek()`** is a method that allows setting the file position indicator of file pointer to the preferred location in the file. The syntax for this method is

```
long Seek(long n, SeekOrigin  location)
```

The file pointer can get or set the position using the `Position` property of `Stream` class. Aside from the `Position` property, these properties are commonly used:

- `bool CanRead` – It returns true if the stream can be read.
- `bool CanSeek` – It returns true if the stream supports position requests.
- `bool CanTimeout` – It returns true if the stream can time out.
- `bool CanWrite` – It returns true if the stream can be written.
- `long length` – It contains the size of the stream.
- `int ReadTimeout` – It indicates the time before a timeout occurs for read operations.
- `int WriteTimeout` – It indicates the time before a timeout occurs for write operations.

Performing Character-Based File I/O

To perform a character-based file for managing text files, use character streams to read and write into the file. Since, internally, each file consists of bytes, the `FileStream` is wrapped inside either a `StreamReader` or a `StreamWriter`. These classes automatically convert a byte stream into a character stream, and vice versa. (Harwani, 2015, p. 468)

`StreamWriter` writes characters to a stream and contains several constructors, such as the following:

- `SteamWriter(Stream stream)` – It is used to create a character-based output stream.
- `StreamWriter(string fileName)` – It is used to open a file directly.

This class contains the following common methods:

- `Close()` – It closes the file.
- `Flush()` – It instantly saves the file content from buffer to memory
- `Write()` – Using a `File` stream class, this writes into the specified file.
- `WriteLine()` – Line by line, it writes into a file.

`StreamReader` is a class that reads characters from a byte stream. It defines the following constructors:

- `StreamReader(Stream stream)` - It is the name of an open stream such I/O devices or a file.
- `StreamReader(string fileName)` – It specifies the name of the file to open.

This class also contains the following common methods:

- `Flush()` – From buffer to memory, it instantly saves the file content.
- `Close()` – It closes the file and is mandatory to this class.
- `Read()` – From the file stream, it reads the content.
- `ReadLine()` – From the given file stream, it reads the content line by line.
- `ReadToEnd()` – From the current location until the end of the stream, it reads all characters.
- `Peek()` – It returns the value in the stream without moving the file pointer.
- `Seek()` – Sets the file pointer at the desired position in a file.

REFERENCES:

Deitel, P. & Deitel, H. (2015). *Visual C# 2012 how to program* (5th ed.). USA: Pearson Education, Inc.

Gaddis, T. (2016). *Starting out with visual C#* (4th ed.). USA: Pearson Education, Inc.

Harwani, B. (2015). *Learning object-oriented programming in C# 5.0.* USA: Cengage Learning PTR.

Miles, R. (2016). *Begin to code with C#.* Redmond Washington: Microsoft Press.

Doyle, B. (2015). *C# programming: from problem analysis to program design* (5th ed.). Boston, MA: Cengage Learning.