**PRACTICAL # 14**

**OBJECT:**

Reading and Writing Files in C#

**THEORY:**

In this lab we will work with text files in C#. We will learn about stream, its purpose and usage.

Streams are an essential part of any input-output library. You can use streams when your program needs to "read" or "write" data to an external data source such as files, other PCs, servers etc.

A stream is an ordered sequence of bytes, which is sent from one application or input device to another application or output device. These bytes are written and read sequence and always arrive in the same order as they were sent. Streams are an abstraction of a data communication channel that connects two devices or applications.

Every time when you read or write from or to a file, you have to open a stream to the corresponding file, do the reading or writing, and then close the stream. There are two types of streams – text streams and binary streams.

**Basic Operations with Streams**

You can do the following operations with streams: creation / opening, reading data, writing data, seeking / positioning, closing / disconnecting. The main classes for working with text streams in .NET are TextReader and TextWriter. Their more important methods are:

• ReadLine() – reads one line of text and returns a string.

• ReadToEnd() – reads the entire stream to its end and returns a string.

• Write() – writes a string to the stream.

• WriteLine() – writes one line of text into the stream.

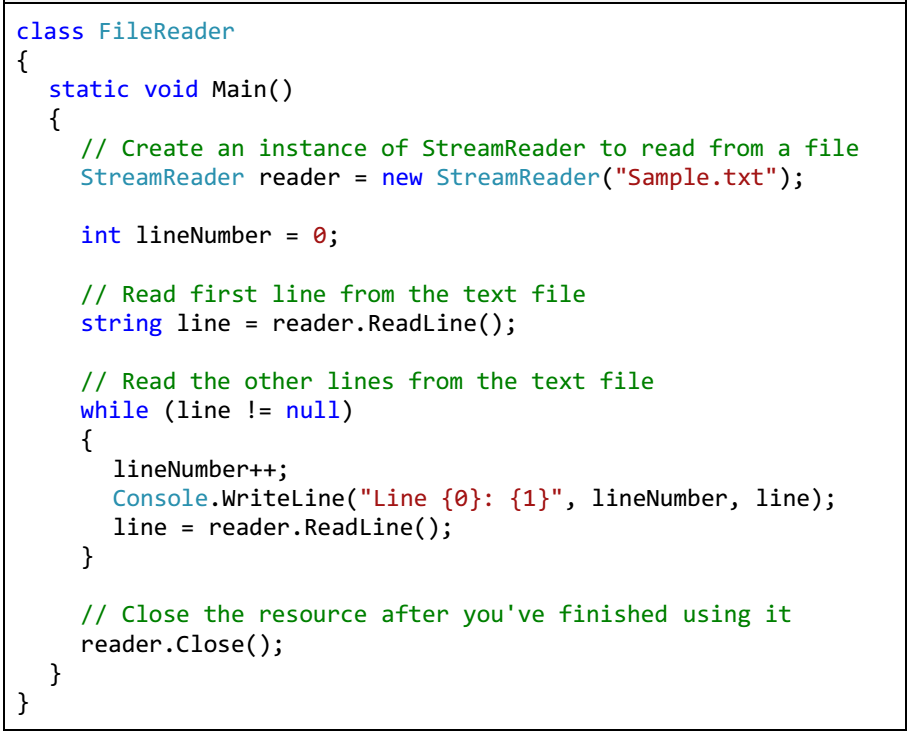
Reading a text file:

To read the entire text file line by line and print the read text on to the console. Create a text file in the Debug folder of the project (.\bin\Debug), which is the same directory in which your compiled application is generated. This avoids the need to set the full path to the file when accessing the file.

We have a text file from which to read. Now we must create an object of type StreamReader to read the file and loop though it line by line.

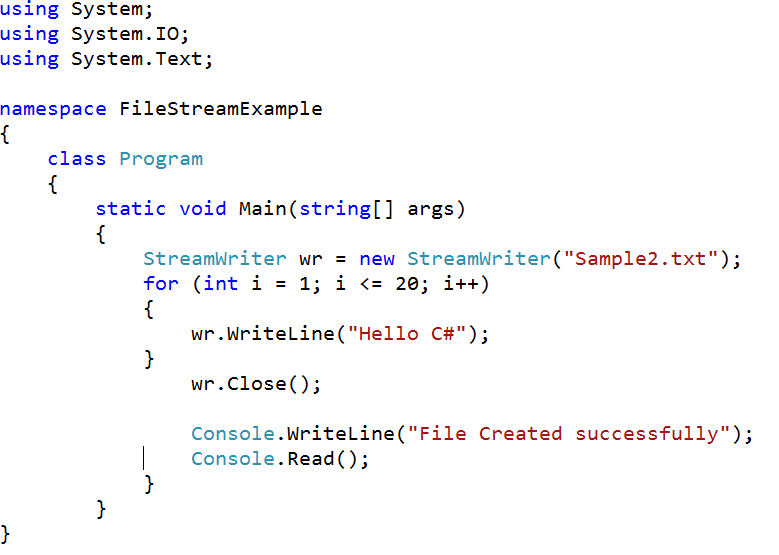
**Program:**

The program below demonstrates reading text file line by line.



**Writing to a Text File**

Text files are very convenient for storing information. For example, we can record the results of a program. We can use text files to make something like a journal (log) for the program – a convenient way to monitor it at runtime. To write text in a text file, we will use StreamWriter class as shown below.



**ACTIVITIES**

**Activity 1**

Generate prime numbers from 100 to 99999 and write these numbers in a text file separated by newline.

Activity 2

Read the file created in activity 1 and output the numbers on console.

**REVIEW QUESTIONS**

1. What are the files used for?
2. What is a stream?
3. Which classes are used to read and write files?