Telerik Academy

http://csharpfundamentals.telerik.com


# Text Files

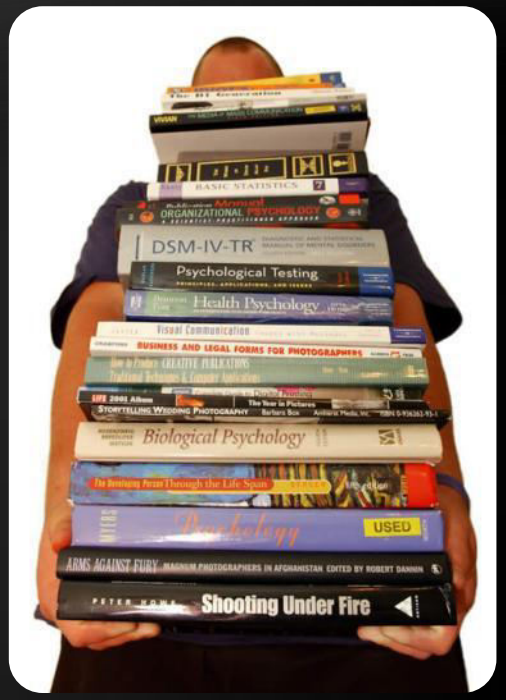
## Reading and Writing Text Files

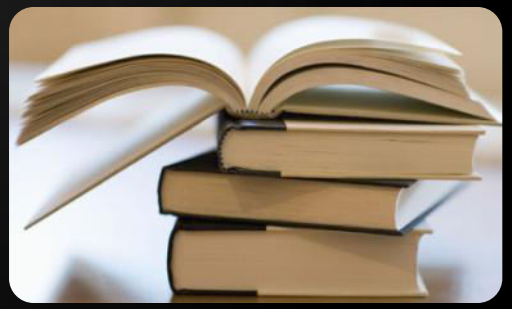
Telerik Software Academy
Learning & Development Team
http://academy.telerik.com


Telerik Academy

# Table of Contents

1. What is Stream?
   - Stream Basics
2. Reading Text Files
   - The `StreamReader` Class
3. Writing Text Files
   - The `StreamWriter` Class
4. Handling I/O Exceptions



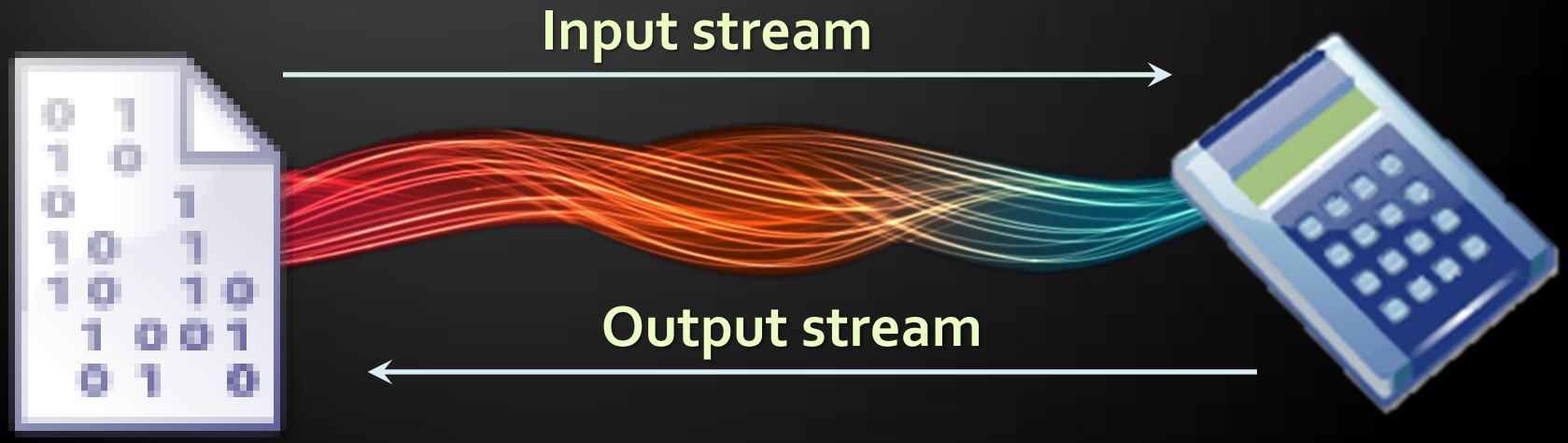
Telerik Academy

# What Is Stream?

## Streams Basic Concepts

# What is Stream?

- Stream is the natural way to transfer data in the computer world
- To read or write a file, we open a stream connected to the file and access the data through the stream



# Streams Basics

- Streams are used for reading and writing data into and from devices
- Streams are ordered sequences of bytes
  - Provide consecutive access to its elements
- Different types of streams are available to access different data sources:
  - File access, network access, memory streams and others
- Streams are open before using them and closed after that

Telerik Academy

# Reading Text Files

## Using the StreamReader Class

# The StreamReader Class

- System.IO.StreamReader
  - The easiest way to read a text file
  - Implements methods for reading text lines and sequences of characters
  - Constructed by file name or other stream
    - Can specify the text encoding (for Cyrillic use windows-1251)
  - Works like Console.Read() / ReadLine() but over text files

# StreamReader Methods

- `new StreamReader(fileName)`
  - Constructor for creating reader from given file
- `ReadLine()`
  - Reads a single text line from the stream
  - Returns `null` when end-of-file is reached
- `ReadToEnd()`
  - Reads all the text until the end of the stream
- `Close()`
  - Closes the stream reader

# Reading a Text File

- Reading a text file and printing its content to the console:

```
StreamReader reader = new StreamReader("test.txt");
string fileContents = reader.ReadToEnd();
Console.WriteLine(fileContents);
streamReader.Close();
```

- Specifying the text encoding:

```
StreamReader reader = new StreamReader(
    "cyr.txt", Encoding.GetEncoding("windows-1251"));
// Read the file contents here ...
reader.Close();
```

Telerik Academy

# Using StreamReader – Practices

- The StreamReader instances should always be closed by calling the Close() method
  - Otherwise system resources can be lost
- In C# the preferable way to close streams and readers is by the "using" construction

```
using (<stream object>)
{
   // Use the stream here. It will be closed at the end
}
```

  - It automatically calls the Close()after the using construction is completed

# Reading a Text File – Example

- Read and display a text file line by line:

```
StreamReader reader =
    new StreamReader("somefile.txt");
using (reader)
{
    int lineNumber = 0;
    string line = reader.ReadLine();
    while (line != null)
    {
        lineNumber++;
        Console.WriteLine("Line {0}: {1}",
            lineNumber, line);
        line = reader.ReadLine();
    }
}
```

Telerik Academy


# Reading Text Files

## Live Demo

Telerik Academy

# Writing Text Files

## Using the StreamWriter Class

# The StreamWriter Class

- System.IO.StreamWriter
  - Similar to StreamReader, but instead of reading, it provides writing functionality
- Constructed by file name or other stream

```
StreamWriter streamWriter =
    new StreamWriter("test.txt");
```

  - Can define encoding
  - For Cyrillic use "windows-1251"

```
StreamWriter streamWriter =
    new StreamWriter("test.txt",
        false, Encoding.GetEncoding("windows-1251"));
```

# StreamWriter Methods

- `Write()`
  - Writes string or other object to the stream
  - Like `Console.Write()`
- `WriteLine()`
  - Like `Console.WriteLine()`
- `Flush()`
  - Flushes the internal buffers to the hard drive
- `AutoFlush`
  - Flush the internal buffer after each writing

# Writing to a Text File – Example

- Create text file named "numbers.txt" and print in it the numbers from 1 to 20 (one per line):

```
StreamWriter streamWriter =
    new StreamWriter("numbers.txt");
using (streamWriter)
{
    for (int number = 1; number <= 20; number++)
    {
        streamWriter.WriteLine(number);
    }
}
```

Telerik Academy

# Writing Text Files

## Live Demo

Telerik Academy

# Handling I/O Exceptions

## Introduction

# What is Exception?

- "An event that occurs during the execution of the program that disrupts the normal flow of instructions" – definition by Google
  - Occurs when an operation can not be completed
- Exceptions tell that something unusual has happened, e. g. error or unexpected event
- I/O operations throw exceptions when operation cannot be performed (e.g. missing file)
  - When an exception is thrown, all operations after it are not processed

# How to Handle Exceptions?

- Using try{}, catch{} and finally{} blocks:

```
try
{
    // Some exception is thrown here
}
catch (<exception type>)
{
    // Exception is handled here
}
finally
{
    // The code here is always executed, no
    // matter if an exception has occurred or not
}
```

# Catching Exceptions

- Catch block specifies the type of exceptions that is caught
  - If `catch` doesn't specify its type, it catches all types of exceptions

```
try
{
  StreamReader reader = new StreamReader("file.txt");
  Console.WriteLine("File successfully open.");
}
catch (FileNotFoundException)
{
  Console.Error.WriteLine(
    "Can not find 'somefile.txt'.");
}
```

# Handling Exceptions When Opening a File

```
try
{
    StreamReader streamReader = new StreamReader(
     "c:\\NotExistingFileName.txt");
}
catch (System.NullReferenceException exc)
{
    Console.WriteLine(exc.Message);
}
catch (System.IO.FileNotFoundException exc)
{
    Console.WriteLine(
     "File {0} is not found!", exc.FileName);
}
catch
{
    Console.WriteLine("Fatal error occurred.");
}
```

Telerik Academy

# Handling I/O Exceptions

Live Demo

Telerik Academy

# Reading and Writing Text Files

## More Examples

# Counting Word Occurrences – Example

- Counting the number of occurrences of the word "foundme" in a text file:

```
StreamReader streamReader =
 new StreamReader(@"..\..\somefile.txt");
int count = 0;
string text = streamReader.ReadToEnd();
int index = text.IndexOf("foundme", 0);
while (index != -1)
{
    count++;
    index = text.IndexOf("foundme", index + 1);
}
Console.WriteLine(count);
```

What is missing in this code?

Telerik Academy


# Counting Word Occurrences

## Live Demo

# Reading Subtitles – Example

- We are given a standard movie subtitles file:

```
.....

{2757}{2803} Allen, Bomb Squad, Special Services...
{2804}{2874} State Police and the FBI!
{2875}{2963} Lieutenant! I want you to go to St. John's
Emergency...
{2964}{3037} in case we got any walk-ins from the
street.
{3038}{3094} Kramer, get the city engineer!
{3095}{3142} I gotta find out a damage report. It's
very important.
{3171}{3219} Who the hell would want to blow up a
department store?

.....
```

# Fixing Subtitles – Example

- Read subtitles file and fix it's timing:

```
static void Main()
{
  try
  {
    // Obtaining the Cyrillic encoding
    System.Text.Encoding encodingCyr =
      System.Text.Encoding.GetEncoding(1251);

    // Create reader with the Cyrillic encoding
    StreamReader streamReader =
      new StreamReader("source.sub", encodingCyr);

    // Create writer with the Cyrillic encoding
    StreamWriter streamWriter =
      new StreamWriter("fixed.sub",
        false, encodingCyr);
```

*(example continues)*

# Fixing Subtitles – Example

```
    try
    {
      string line;
      while (
        (line = streamReader.ReadLine()) != null)
      {
        streamWriter.WriteLine(FixLine(line));
      }
    }
    finally
    {
      streamReader.Close();
      streamWriter.Close();
    }
  }
  catch (System.Exception exc)
  {
    Console.WriteLine(exc.Message);
  }
}
```

`FixLine(line)` perform fixes on the time offsets: multiplication or/and addition with constant

Telerik Academy


# Fixing Movie Subtitles

## Live Demo

# Summary

- Streams are the main I/O mechanisms in .NET
- The `StreamReader` class and `ReadLine()` method are used to read text files
- The `StreamWriter` class and `WriteLine()` method are used to write text files
- Always put file handling in `using(…)` block
- Exceptions are unusual events or error conditions
  - Can be handled by try-catch-finally blocks

Telerik Academy

# Text Files

# Questions?

http://csharpfundamentals.telerik.com

# Exercises

1. Write a program that reads a text file and prints on the console its odd lines.
2. Write a program that concatenates two text files into another text file.
3. Write a program that reads a text file and inserts line numbers in front of each of its lines. The result should be written to another text file.
4. Write a program that compares two text files line by line and prints the number of lines that are the same and the number of lines that are different. Assume the files have equal number of lines.

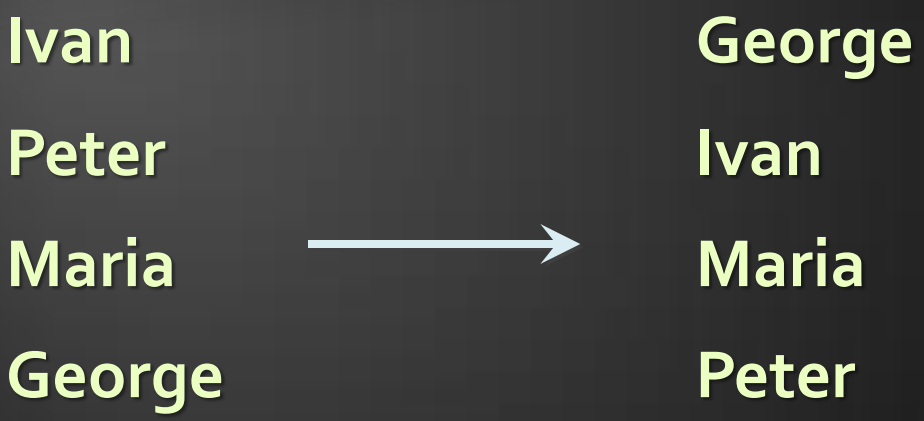
# Exercises (2)

5. Write a program that reads a text file containing a square matrix of numbers and finds in the matrix an area of size 2 x 2 with a maximal sum of its elements. The first line in the input file contains the size of matrix N. Each of the next N lines contain N numbers separated by space. The output should be a single number in a separate text file. Example:

```
4
2 3 3 4
0 2 3 4      ⟶    17
3 7 1 2
4 3 3 2
```

# Exercises (3)

6. Write a program that reads a text file containing a list of strings, sorts them and saves them to another text file. Example:




7. Write a program that replaces all occurrences of the substring "start" with the substring "finish" in a text file. Ensure it will work with large files (e.g. 100 MB).
8. Modify the solution of the previous problem to replace only whole words (not substrings).

# Exercises (4)

9. Write a program that deletes from given text file all odd lines. The result should be in the same file.
10. Write a program that extracts from given XML file all the text without the tags. Example:

```
<?xml version="1.0"><student><name>Pesho</name>
<age>21</age><interests count="3"><interest>
Games</instrest><interest>C#</instrest><interest>
Java</instrest></interests></student>
```

11. Write a program that deletes from a text file all words that start with the prefix "test". Words contain only the symbols 0…9, a…z, A…Z, _.

# Exercises (5)

12. Write a program that removes from a text file all words listed in given another text file. Handle all possible exceptions in your methods.
13. Write a program that reads a list of words from a file `words.txt` and finds how many times each of the words is contained in another file `test.txt`. The result should be written in the file `result.txt` and the words should be sorted by the number of their occurrences in descending order. Handle all possible exceptions in your methods.

# Free Trainings @ Telerik Academy

- "C# Programming @ Telerik Academy
  - csharpfundamentals.telerik.com
- Telerik Software Academy
  - academy.telerik.com
- Telerik Academy @ Facebook
  - facebook.com/TelerikAcademy
- Telerik Software Academy Forums
  - forums.academy.telerik.com