Contents

Description

Purpose

This project is designed to teach you how to manipulate files and exceptions. It reinforces your understanding of class input/output operations on files and forces you to be creative with a limited toolbox.

Challenge

In short

Develop a static class to perform simple operations on files: display its content, display one of its lines, display it with truncation, and display words surrounding a particular word. A bonus (for more experienced programmers) asks to compute the winner of a race using data shared in a comma-separated values file.

In more details

Your goal is to design and implement a static `TextFileHelper` class containing 5 methods (the fifth one being optional):

1. A `Display` method that takes a `string` as an argument and displays the content of the file located at the corresponding path if it exists, and an error message otherwise.
2. A `DisplayN` method that takes a `string` and an `int`, and displays the line corresponding to the `int` argument from the file located at the corresponding path if it exists (with a line number before), and an error message otherwise.
3. A `DisplayT` method that takes a `string` and an `int`, and displays the file located at the corresponding path "truncated" at the char-

acter number given by the `int` argument if the file exists, and an error message otherwise.

4. A `DisplayW` method that takes two `string` arguments, called `path` and `word`. If the file at the corresponding path exists, then for every line in the file that contains `word`, the method should display `word`, and the words immediately before and after, if there are any.

5. (optional) A `DisplayMaxCSV` method that takes a `string` argument containing the path of a `.csv` (comma-separated value) file containing values organized as follows:

   ```
   Index,First Name,Last Name,Lap 1, Lap 2, Lap 3
   1,Shelby,Terrell,78,80,82
   2,Phillip,Summers,76,82,91
   3,Kristine,Travis,78,82,83
   4,Marty,Gave,67,73,72
   5,Harley,Test,78,91,72
   ```

   and displays the first and last name of the contestant with the best time, adding their results from the three laps.

⚠ Warning

You are allowed to use only `StreamReader`'s `ReadLine` and constructor, `StreamWriter`'s `WriteLine` and `Write` as well as their constructors, and the usual `Console`, `char`, `string` (including for example `Substring` or `Split`) and `array` methods, conditionals, etc. **In particular, *do not* use the Linq library.** `AppDomain`, `Path` and `File` methods can be used as in class or in the example below.

Submission

Start by downloading this project to help you getting started.

Please, follow our guideline on project submission. In particular, make sure you write your name and the date in a delimited comment at the beginning of your file.

Example

Here is an example of a `Program.cs` `Main` method:

```
using System;
using System.IO;

class Program
{
```

```
static void Main()
{
  /*
   * We first create three
   * string variables where
   * our three demo files
   * are located.
   */
  string directoryPath = AppDomain
    .CurrentDomain
    .BaseDirectory;
  string ex0 = Path.Combine(directoryPath, "ex0.txt");
  string ex1 = Path.Combine(directoryPath, "ex1.txt");
  string ex2 = Path.Combine(directoryPath, "ex2.csv");

  // Testing the "Display" Method:
  Console.WriteLine("*****\n* Testing Display\n*****");
  FileDisplayer.Display(ex0);
  // Testing the "DisplayN" method
  Console.WriteLine("*****\n* Testing DisplayN\n*****");
  FileDisplayer.DisplayN(ex0, 1);
  FileDisplayer.DisplayN(ex0, 3);
  FileDisplayer.DisplayN(ex0, 10);
  FileDisplayer.DisplayN(ex0, 0);
  FileDisplayer.DisplayN(ex0, -10);
  // Testing the "DisplayT" method
  Console.WriteLine(
    "*****\n* Testing DisplayT (1/2)\n*****"
  );
  FileDisplayer.DisplayT(ex0, 20);
  Console.WriteLine(
    "*****\n* Testing DisplayT (2/2)\n*****"
  );

  FileDisplayer.DisplayT(ex1, 5);
  // Testing the "DisplayW" method
  Console.WriteLine(
    "*****\n* Testing DisplayW (1/2)\n*****"
  );
  FileDisplayer.DisplayW(ex0, "line");

  Console.WriteLine(
    "*****\n* Testing DisplayW (2/2)\n*****"
  );
  FileDisplayer.DisplayW(ex1, "short");
  // Testing the "DisplayWinnerCSV" method
```

```
    Console.WriteLine(
      "*****\n* Testing DisplayWinnerCSV
       ↪  (Optional)\n*****"
    );
    FileDisplayer.DisplayWinnerCSV(ex2);
  }
}
```

(Download this code)

Executing it with the properly implemented `FileDisplayer` missing methods should give something along the lines of:

```
*****
* Testing Display
*****
This is the first demo file. It contains a long first
 ↪  line that will not wrap nicely on some display.
It also contains another line that contains the word line
 ↪  also.
It will be used to test our methods.
*****
* Testing DisplayN
*****
Line 1: "This is the first demo file. It contains a long
 ↪  first line that will not wrap nicely on some display."
Line 3: "It will be used to test our methods."
End of file was reached before reaching line number 10.
The line number should be strictly greater than 0.
The line number should be strictly greater than 0.
*****
* Testing DisplayT (1/2)
*****
This is the first de
mo file. It contains
 a long first line t
hat will not wrap ni
cely on some display
.
It also contains ano
ther line that conta
ins the word line al
so.
It will be used to t
est our methods.
*****
```

```
* Testing DisplayT (2/2)
*****
A ver
y sho
rt
demo
file.
*****
* Testing DisplayW (1/2)
*****
first line that
another line that
word line also.
*****
* Testing DisplayW (2/2)
*****
very short
*****
* Testing DisplayWinnerCSV (Optional)
*****
The winner is Marty Gave.
```

Note that it is ok if you cannot reproduce this output *exactly*.

Bonuses

- (easy) If the file doesn't exist, handle exceptions gracefully.
- (medium) Write the `DisplayWinnerCSV` method.
- (hard) Make `DisplayW` accommodate punctuation signs (`,`, `.`, `:`, etc.).