2024-11-19

# Files (Solutions)

## Questions

1. What is the name of the class we use to read from files?
   * FileReader
   * StreamOpener
   * ReadFile
   * StreamReader
   * FileStream
2. What is of crucial importance?
   * Always call the Open method before reading from or writing into a file.
   * Always assume the user has C:\ as their main drive.
   * Always call the Close method after being done reading from or writing into a file.
   * Never read from or write into a file inside a try…catch block.

## Problems

1. Write a program that create a text file called HelloWorld.txt in its bin/Debug folder and store “Hello” followed by a name entered by the user in it.

* Solution
* string uName;  
   Console.WriteLine("Please, enter your name.");  
   uName = Console.ReadLine();  
    
   try  
   {  
   string filePath = Path.Combine(  
   AppDomain.CurrentDomain.BaseDirectory,  
   "HelloWorld.txt"  
   );  
   // This string contains the path for the file we create.  
   StreamWriter sw = new StreamWriter(filePath);  
   // We create the file  
   sw.WriteLine("Hello " + uName);  
   sw.Close();  
   Console.WriteLine("Check out " + filePath + "!");  
   }  
   catch (Exception e)  
   {  
   Console.WriteLine("Exception: " + e.Message);  
   }
* [*(Download this code)*](https:/princomp.github.io/code/projects/HelloWorldFile.zip)

1. Write a program that ask the user for a filename, makes sure the filename ends with “.txt” and does not begin with a “.” (otherwise, the file would be hidden on unix systems), does not match a file with the same name in the bin/Debug folder of your program, then create it in the bin/Debug folder of your program and write in it all the number from 1 to 1,000,000 (both included). Out of curiosity, what is the file size?

* Solution
* string fName,  
   filePath;  
   do  
   {  
   Console.WriteLine("Enter a file name.");  
   fName = Console.ReadLine();  
   filePath = Path.Combine(  
   AppDomain.CurrentDomain.BaseDirectory,  
   fName  
   );  
   } while (  
   !fName.EndsWith(".txt")  
   || fName.StartsWith(".")  
   || File.Exists(filePath)  
   );  
   try  
   {  
   StreamWriter sw = new StreamWriter(filePath);  
   for (int i = 1; i <= 1000000; i++)  
   sw.WriteLine(i);  
   sw.Close();  
   Console.WriteLine("Check out " + filePath + "!");  
   }  
   catch (Exception e)  
   {  
   Console.WriteLine("Exception: " + e.Message);  
   }
* [*(Download this code)*](https:/princomp.github.io/code/projects/FileCreation.zip)
* The resulting file is about 6.6 MB on Unix system, 7.52 MB on Windows system.

1. Execute the following program, then write a program that find the greatest number in the RandomNumber.txt file.

* string filePath = Path.Combine(  
   AppDomain.CurrentDomain.BaseDirectory,  
   "RandomNumbers.txt"  
   );  
   Random gen = new Random();  
   try  
   {  
   StreamWriter sw = new StreamWriter(filePath);  
   for (int i = 1; i <= 100; i++)  
   sw.WriteLine(gen.Next(1000));  
   sw.Close();  
   }  
   catch (Exception e)  
   {  
   Console.WriteLine("Exception: " + e.Message);  
   }
* Solution
* string line;  
   int number;  
   int maxSoFar;  
   try  
   {  
   StreamReader sr = new StreamReader(filePath);  
   line = sr.ReadLine();  
   if (int.TryParse(line, out number))  
   {  
   maxSoFar = number;  
   }  
   else  
   {  
   throw new ArgumentException(  
   "File contains string that is not a number."  
   );  
   }  
   while (line != null)  
   {  
   if (int.TryParse(line, out number))  
   {  
   if (maxSoFar < number)  
   {  
   maxSoFar = number;  
   }  
   }  
   else  
   {  
   throw new ArgumentException(  
   "File contains string that is not a number."  
   );  
   }  
   line = sr.ReadLine();  
   }  
   Console.WriteLine(  
   "The maximum number in the file is "  
   + maxSoFar  
   + "."  
   );  
   }  
   catch (Exception e)  
   {  
   Console.WriteLine("Exception: " + e.Message);  
   }
* [*(Download this code)*](https:/princomp.github.io/code/projects/FileRandomNumber.zip)

1. Suppose that at filePath is located a file where each line is either
   * a decimal (e.g., 12.4, -14, 0.34),
   * the word “STOP”,
   * some other string (“Test”, “The sky is blue”, “Ignore this line”, “My file contains”), that may contain the characters “STOP”.

* Write a program that displays the average of the decimals in the file knowing that
  + your program should ignore the values after a line containing “STOP” and only “STOP” if it is present,
  + all the other strings should simply be ignored.
* For example, for the following three files, “4.0”, “10.0” and “7.5” should be displayed, as (12.48 - 2.48 + 2) / 3 = 4 (13 been ignored), (15 + 5) / 2 = 10, and (11 + 4) / 2 = 7.5 (12 being ignored).
* ┌────────────────┐  
  │12.48 │  
  │This is a test │   
  │-2.48 │  
  │2 │  
  │STOP │  
  │13 │  
  └────────────────┘  
    
  ┌────────────────┐  
  │My file contains│  
  │STOP but │  
  │averages │  
  │15 │  
  │ and │  
  │5 │   
  └────────────────┘  
    
  ┌────────────────┐  
  │This 12 will be │  
  │ignored │  
  │but not │  
  │11 │  
  │ nor │  
  │4 │   
  └────────────────┘
* Solution
* double number;  
   double sum = 0;  
   int counter = 0;  
   double average;  
   try  
   {  
   // Opening file  
   StreamReader sr = new StreamReader(filePathP);  
   // Reading first line  
   string line = sr.ReadLine();  
    
   // Looping through the file until we  
   // reach the end, or read the word  
   // "STOP" on its own line.  
   while (line != null && line != "STOP")  
   {  
   // We test if the line contains a double.  
   if (double.TryParse(line, out number))  
   {  
   sum += number;  
   counter++;  
   }  
   // We read the next line.  
   line = sr.ReadLine();  
   }  
   sr.Close();  
   }  
   catch { }  
   // if to prevent division by 0.  
   if (counter != 0)  
   {  
   average = sum / counter;  
   }  
   else  
   {  
   average = -1;  
   }
* [*(Download this code)*](https:/princomp.github.io/code/projects/AverageNumberFromFiles.zip)

1. Write a program that asks the user to enter a sentence, and store it in a file *where the maximum width is 40*: if the string entered is more than 40 characters long, then it should span over multiple lines of no more than 40 characters each. For example, if the user enters

* In publishing and graphic design, Lorem ipsum is a placeholder text commonly used to demonstrate the visual form of a document or a typeface without relying on meaningful content. Lorem ipsum may be used as a placeholder before the final copy is available.
* then the text file should contain
* In publishing and graphic design, Lorem   
  ipsum is a placeholder text commonly use  
  d to demonstrate the visual form of a do  
  cument or a typeface without relying on   
  meaningful content. Lorem ipsum may be u  
  sed as a placeholder before the final co  
  py is available.
* Solution
* string filePath = Path.Combine(  
   AppDomain.CurrentDomain.BaseDirectory,  
   "TextTruncate.txt"  
   );  
   Console.WriteLine("Enter a string.");  
   string uString = Console.ReadLine();  
    
   const int MAXWIDTH = 40;  
    
   try  
   {  
   StreamWriter sw = new StreamWriter(filePath);  
   while (uString.Length > MAXWIDTH)  
   {  
   sw.WriteLine(uString.Substring(0, MAXWIDTH));  
   uString = uString.Substring(MAXWIDTH);  
   }  
   sw.WriteLine(uString);  
   sw.Close();  
   }  
   catch (Exception e)  
   {  
   Console.WriteLine("Exception: " + e.Message);  
   }
* [*(Download this code)*](https:/princomp.github.io/code/projects/FileTruncate.zip)

1. Write a program that counts the number of words in itself! Ideally, empty lines should not count toward the word count.

* Hint: Program.cs is normally located at
* Path.Combine(  
   new DirectoryInfo(Directory.GetCurrentDirectory()).Parent.Parent.ToString(),  
   "Program.cs"  
  )
* Solution
* ﻿using System;  
  using System.IO;  
    
  class Program  
  {  
   static void Main()  
   {  
   // The following goes to /bin/Debug, then two folders up,  
   // where Program.cs is located.  
   string filePath = Path.Combine(  
   new DirectoryInfo(  
   Directory.GetCurrentDirectory()  
   ).Parent.Parent.ToString(),  
   "Program.cs"  
   );  
    
   if (!File.Exists(filePath))  
   {  
   Console.WriteLine(  
   "There seems to be an issue. Impossible to locate Program.cs"  
   );  
   }  
   else  
   {  
   Console.WriteLine("Program.cs located, processing.");  
   string line;  
   string[] words;  
   int wCount = 0;  
   try  
   {  
   StreamReader sr = new StreamReader(filePath);  
   line = sr.ReadLine();  
   while (line != null)  
   {  
   words = line.Split(  
   new string[] { " " },  
   StringSplitOptions.RemoveEmptyEntries  
   );  
   wCount += words.Length;  
   line = sr.ReadLine();  
   }  
   sr.Close();  
   Console.WriteLine(  
   "Your program contains " + wCount + " words!"  
   );  
   // Should display "Your program contains 121 words!"  
   }  
   catch (Exception e)  
   {  
   Console.WriteLine("Exception: " + e.Message);  
   }  
   }  
   }  
  }
* [*(Download this code)*](https:/princomp.github.io/code/projects/FileCountProgram.zip)