**Міністерство освіти і науки України**

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

**Факультет інформатики та обчислювальної техніки**

**Кафедра обчислювальної техніки**

**Лабораторна робота №2**

з дисципліни

«Компоненти програмної інженерії. Якість та тестування програмного забезпечення»

на тему

«Unit тестування»

Виконав:

Перевірив:

студент групи ІП-93

Бабарикін Ігор Владиславович

Домінський Валентин Олексійович

номер залікової книжки: 9311

Київ 2021

Зміст

[Мета: 3](#_Toc67749075)

[Завдання: 3](#_Toc67749076)

[Хід роботи: 3](#_Toc67749077)

[Початок роботи: 3](#_Toc67749078)

[FilePath 4](#_Toc67749079)

[GetFileName 5](#_Toc67749080)

[GetFullPath 6](#_Toc67749081)

[ReadAll 6](#_Toc67749082)

[ReadLines 9](#_Toc67749083)

[TryToWrite 11](#_Toc67749084)

[Write 14](#_Toc67749085)

[IsPathValid 16](#_Toc67749086)

[MkDir 17](#_Toc67749087)

[TryCopy 17](#_Toc67749088)

[Тестування: 19](#_Toc67749089)

[ReadLines 19](#_Toc67749090)

[TryCopy 19](#_Toc67749091)

[Результати тестування 19](#_Toc67749092)

[Сирцеві коди: 20](#_Toc67749093)

[Program.cs (де проводив досліди): 20](#_Toc67749094)

[TestFileWorkingUtils (тести) 23](#_Toc67749095)

[Висновки: 46](#_Toc67749096)

[Джерела: 47](#_Toc67749097)

Мета:

Написати Unit тести з використанням методів Black Box Testing

Завдання:

|  |  |  |
| --- | --- | --- |
| N п/п | 9311 mod 3 | Library |
| 1 | 0 | PasswordHasher |
| 2 | 1 | BinaryFlag |
| 3 | 2 | FileWorker (any) |

Варіант = 9311 mod 3 = 2, отже провести тестування FileWorker (будь-якого)

Отже, мій вибір для лабораторної:

* .NET 5
* Бібліотека для тестування xUnit
* Бібліотека IIG.Core.FileWorkingUtils

Хід роботи:

Початок роботи:

Я створив проект “Lab2” на NET 5, додав xUnit Test Project “TestFileWorkingUtils” та бібліотеку IIG.Core.FileWorkingUtils:



Для початку Нам треба зрозуміти, як з цим працювати, тому початок роботи буде у файлі Program. Нам треба підключити простір імен бібліотеки:



Тепер спробую створити екземпляр класу FileWorker:



Бачимо, що конструктор потребує string path. З назви можна зрозуміти, що треба шлях до якогось файлу, тому я створив TempFile.txt та додав його відносний шлях до конструктору:



Тепер я хочу переглянути публічні поля/методи цього екземпляру:  


FilePath

Першим полем, що відноситься до Нашого екземпляру є FilePath, яке можна тільки отримати (встановити його значення можна лише при створенні):



При спробі запуску проекту видає помилку:



Отже конструктор вимагає повний шлях:



Результат запуску:



Як Ми бачимо значення поля FilePath ідентичне до того, що Ми писали у конструкторі.

Тепер спробуємо змінити поле:



Отже поле дійсно можна лише отримати.

GetFileName





Як можна помітити, метод може працювати без параметрів, тому виклик повинен видати Нам “TempFile.txt”:



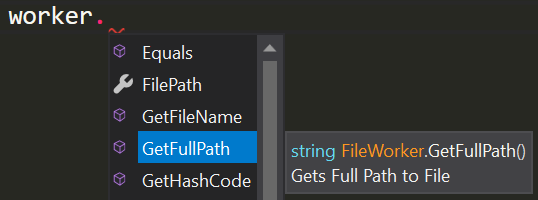
І дійсно, сталося те, про що я і казав.

Також Ми можемо викликати цей метод використовуючи клас (я дізнався про це, бо Visual Studio показує, що метод має одне додаткове перевантаження):

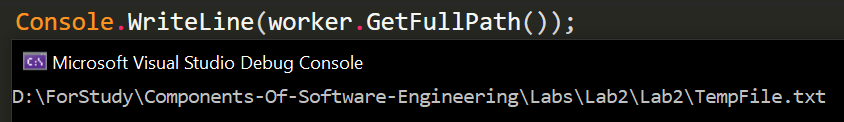




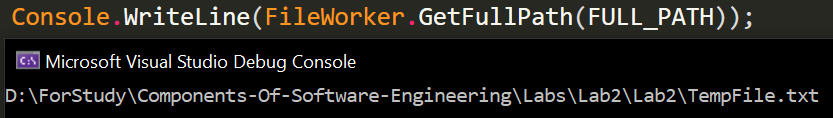
GetFullPath



З назви можна здогадатися, що метод повинен повертати повний шлях до файлу, який Ми вказали при створенні екземпляру. Давайте перевіримо це:

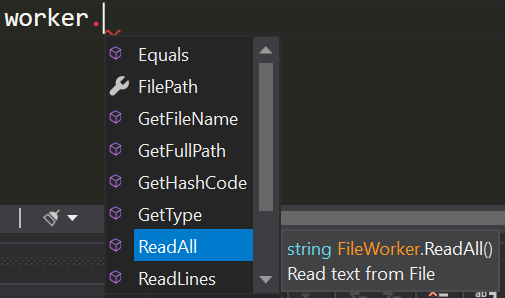


Тут теж є одне перевантаження методу, тому давайте за аналогією з минулою функцією спробуємо його викликати:

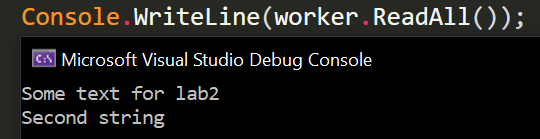
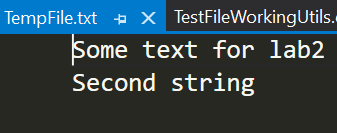


Наша теорія вірна і метод дійсно працює так само. Але викликати його не за допомогою екземпляру немає сенсу, оскільки Ми, по суті, отримуємо той самий результат, що і передали як параметри.

ReadAll

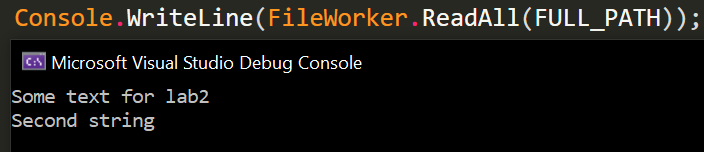


Visual Studio каже Нам, що цей метод може зчитувати дані з файлу, отже створення текстового файлу TempFile не було помилкою і Нам дійсно треба формат, який буде легко читатися.

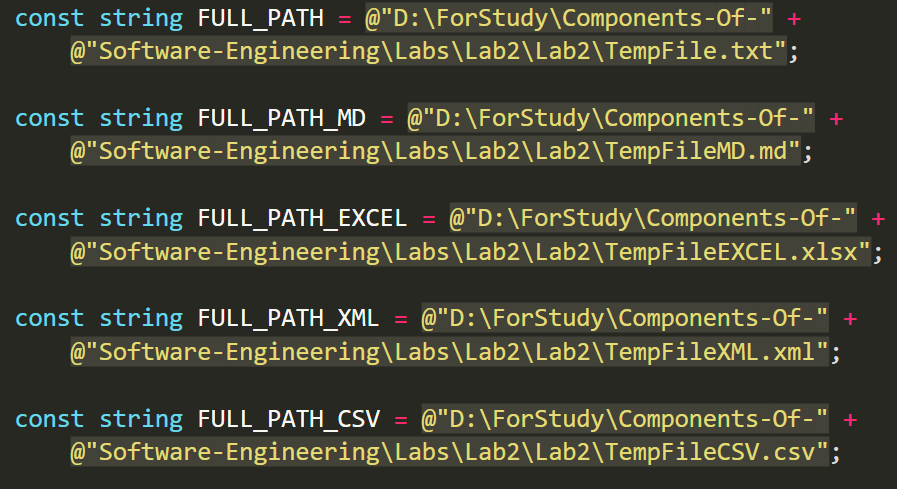
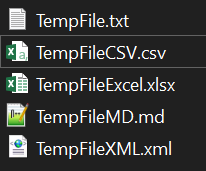
 

Як бачимо, метод дійсно читає усі стрічки з файлу, які за допомогою Console.WriteLine() можна вивести на екран.

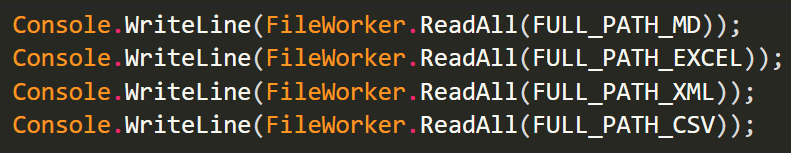
Перевіримо другий варіант виклику даної функції:

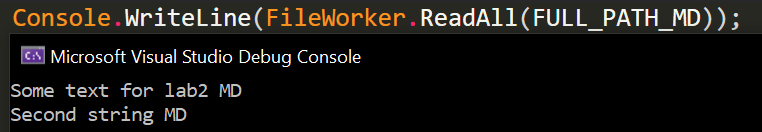
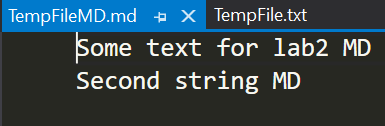


До консолі вивелися дані ідентичні до минулих, які Ми отримали використовуючи екземпляр.

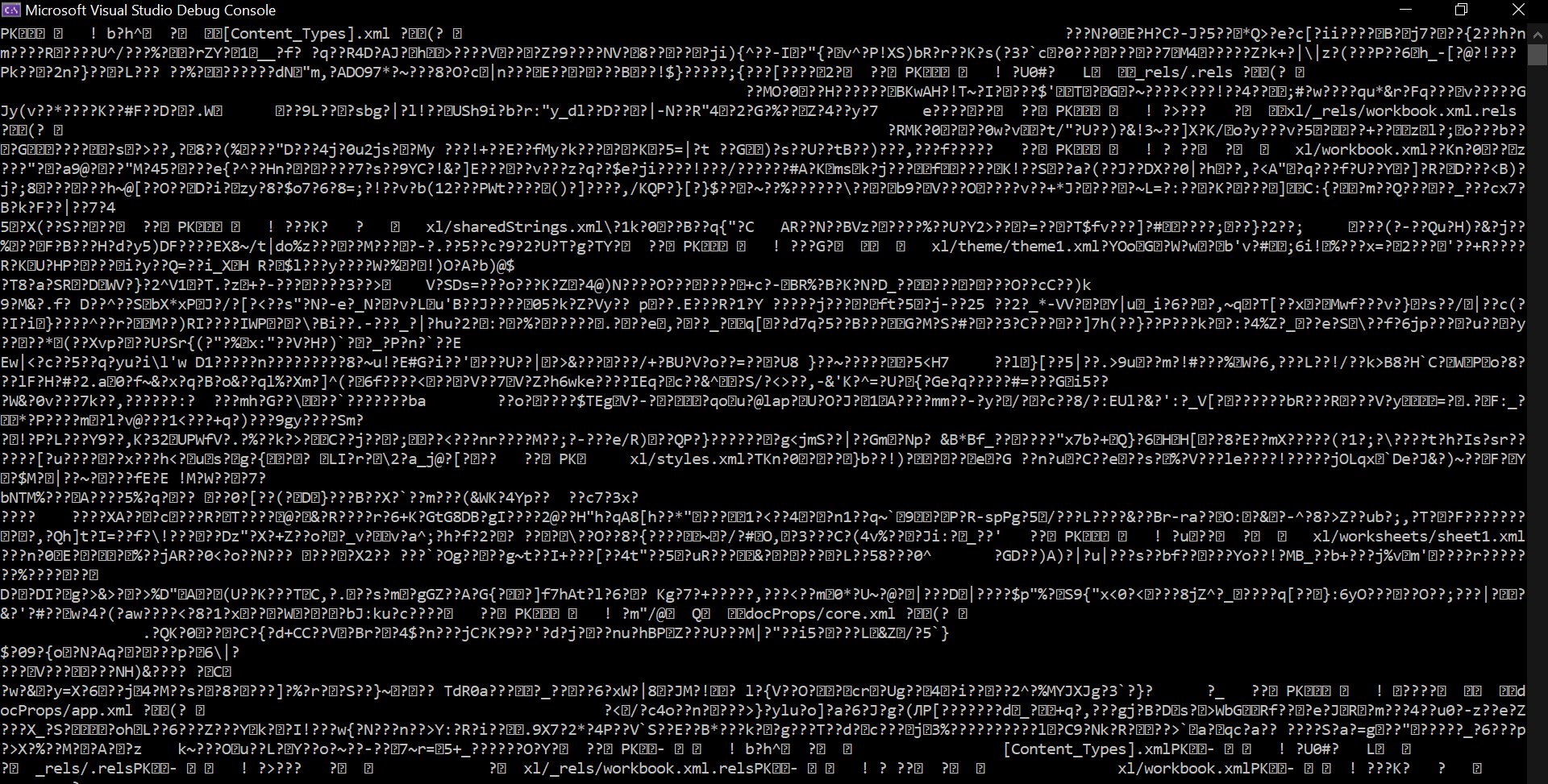
Під час тестування цього методу у мене виникло питання: «А які файли можна ще відкрити?». Далі я й спробую це вияснити: 

Я створив чотири додаткові файли з розширеннями \*.csv, \*.xlsx, \*.md та \*.xml, створив змінні з відповідними шляхами вивівдо консолі результати:

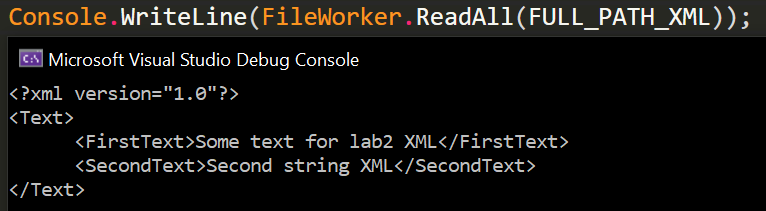


Метод чудово справився з форматом .\*md. Перейдемо до наступного:

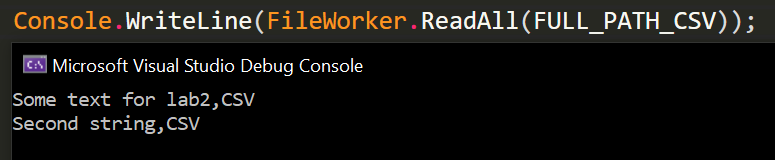


Коли я створював файл \*.xslx відразу було зрозуміло результат, оскільки для правильного читання такого формату треба доволі потужні програми, які розроблюються гігантами на кшталт Microsoft. А що там з XML?



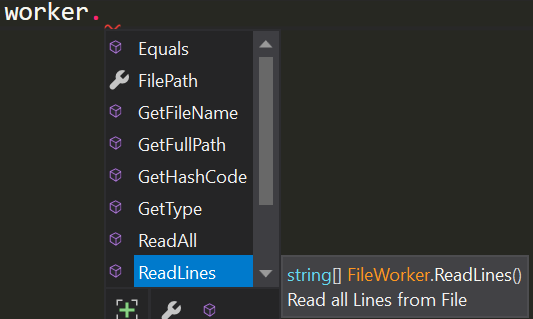
Як бачимо, файл був успішно прочитаний.

Перейдемо до \*.csv. Це, на мою думку, найцікавіший експеримент, оскільки першочергово це був документ \*.xlsx, якому змінили розширення:



Експеримент пройшов успішно. Отже, дійдемо до висновку, що файли, які потрібно відкривати спеціальними програмами (\*.xlsx) не можна успішно прочитати за допомогою цього методу. У той же час Ми маємо \*.csv файл, який при відкриванні запускається у excel. Отримуємо другий висновок: файли, які не повинні обов’язково відкриватися в окремих програмах, можна спокійно використовувати як параметри для даної функції. Ну і останнє, це те, що файли зі звичайним форматом тексту (\*.txt, \*.md, \*.xml) можна прочитати без зусиль.

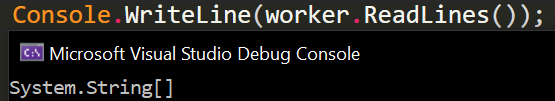
ReadLines



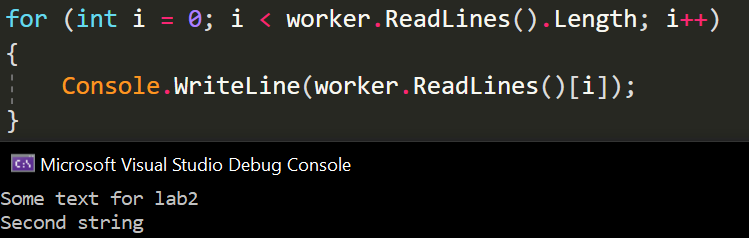
На перший погляд не зрозуміло, яка різниця між цим методом та минулим, але, якщо придивитися, то можна її помітити у значенні, яке повертається.

У даному випадку Ми отримуємо масив стрічок, на відміну від ReadAll, де була одна суцільна стрічка тексту.

Звичайний Console.WriteLine тут не підійде,

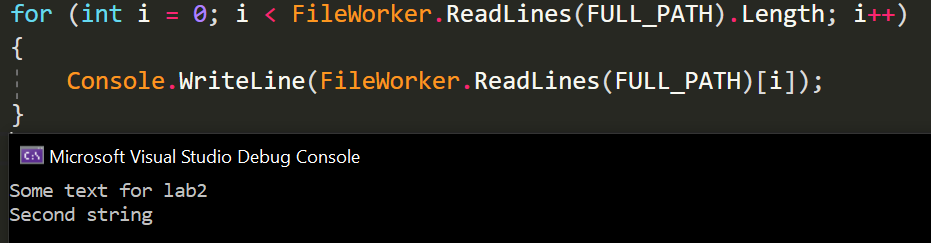


Тому треба використати цикл:



І маємо правильний вивід.

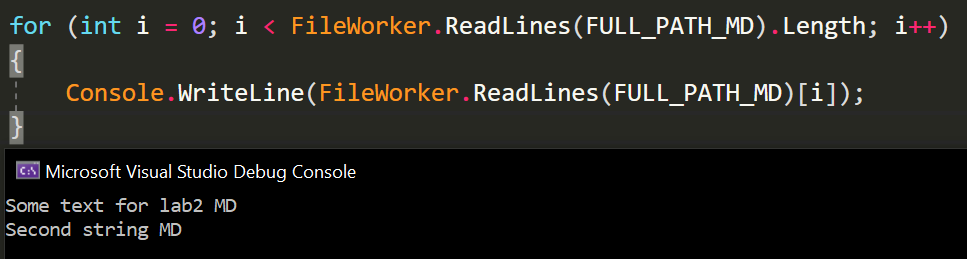
Зробимо те саме, але без екземпляру:



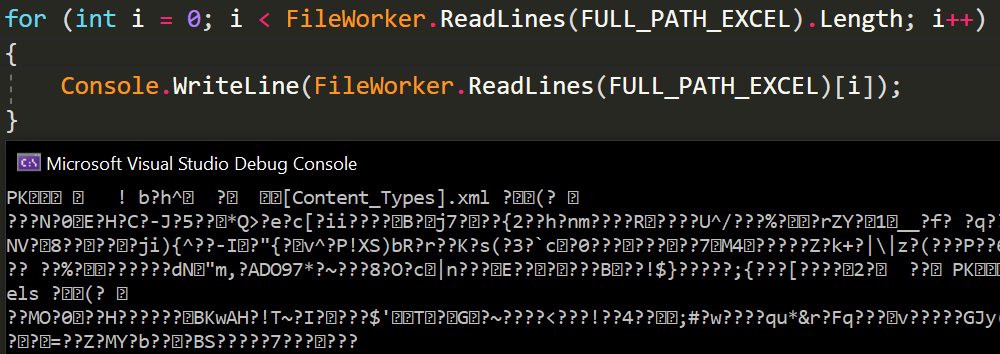
Як бачимо Стрічки ідентичні

Тепер спробуємо провернути ті самі експерименти, що й з методом ReadAll:

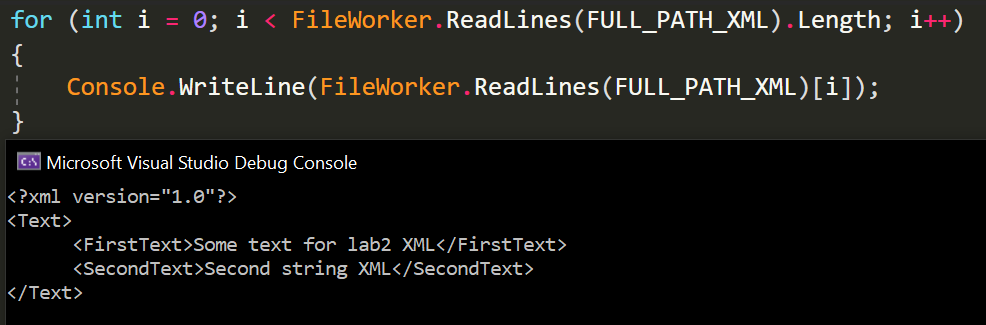
\*.md:



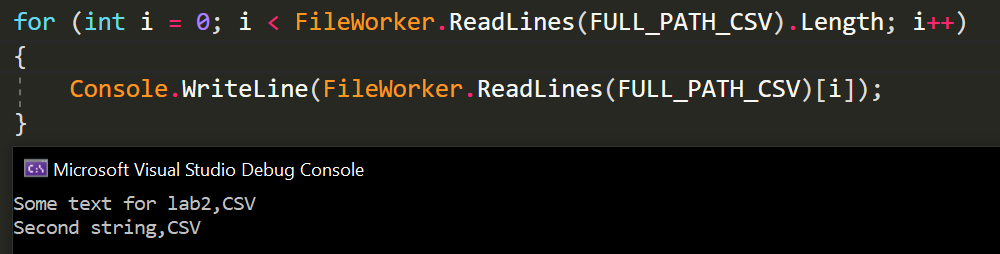
\*.xslx:



\*.xml:

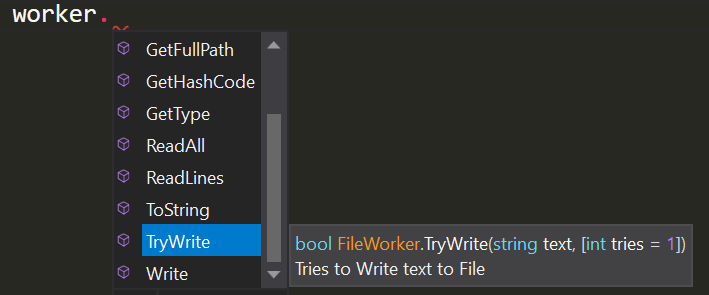


Та \*.csv:

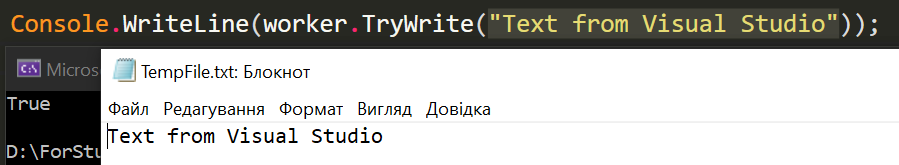


Можемо дійти до того самого висновку, що й з минулим методом.

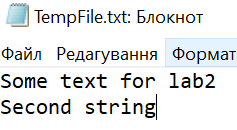
TryToWrite



Скоріше всього даний метод намагається лише один раз записати певну стрічку до файлу, тому давайте спробуємо це зробити:

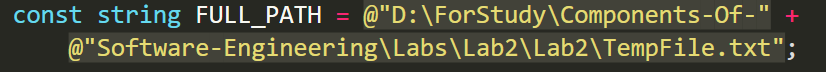


Як бачимо, минулий текст

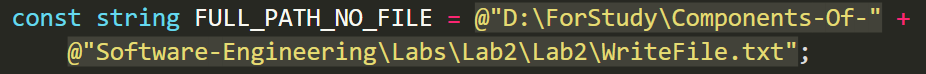


був замінений на “Text from Visual Studio”. Але ж к-сть спроб = 1, тому давайте спробуємо не дати функції записати текст до файлу:

Спочатку були спроби знайти такий символ, який не зможе розпізнатися, але дуже швидко ця ідея зійшла нанівець. Тому наступним кроком стала зміна шляху з:



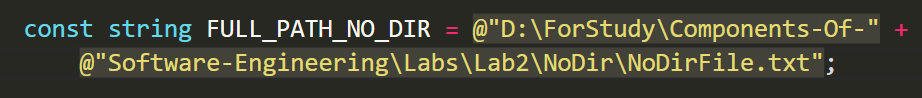
на:



Але при запуску проекту я помітив стрічку “True”. Спочатку Я не розумів, чому метод пройшов успішно, адже такого файлу просто нема. Лише через хвилину до мене дійшло, що був створений новий файл з назвою, яка задана у константі і текстом, який передається як параметр!



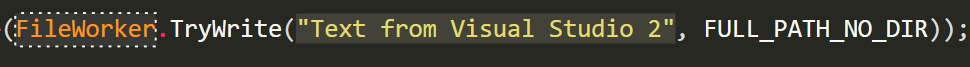
«А що Ти скажеш, якщо не буде директорії?»,- подумав я.



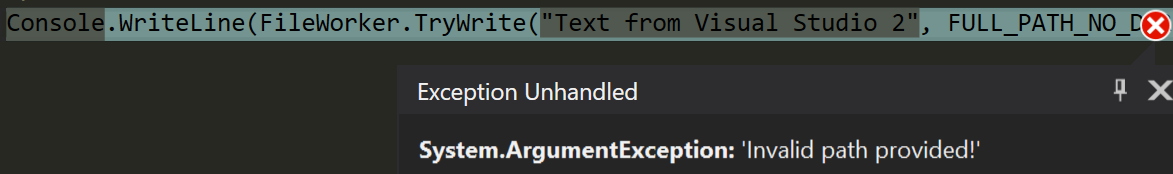
Тут мене вже зупинив компілятор, який сказав, що шлях невірний.

«Добре, я якраз не тестував для цього методу виклик через клас».

На жаль, даний варіант теж не пройшов, оскільки методу треба шлях, якого у нього нема і він ніяк не передається. Але у той же час у xml документі для бібліотеки було одне перевантаження, яке могло приймати шлях до файлу як параметр:

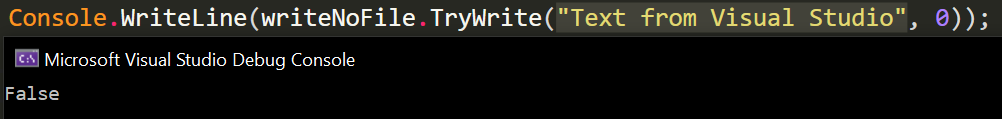


І ось, що вийшло:

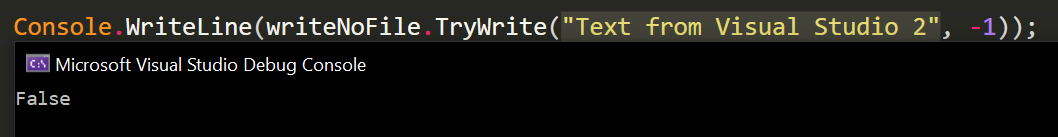


Видно, що Visual Studio скаржиться на неправильний шлях.

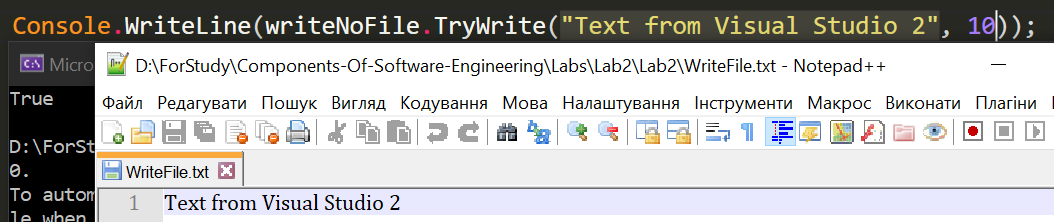
Далі я вирішив спробувати змінити кількість спроб. У першу чергу мене цікавило, що буде при значенні 0:



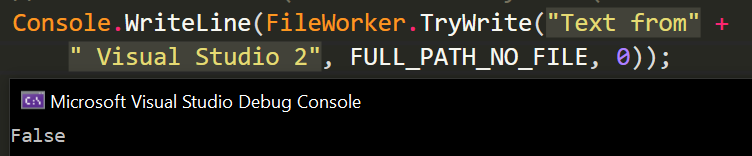
Як бачимо, метод вичерпав усі Свої спроби для запису тексту до файлу та/або створення самого документу, через що й видав Нам “False”. А як відреагує на від’ємне значення?

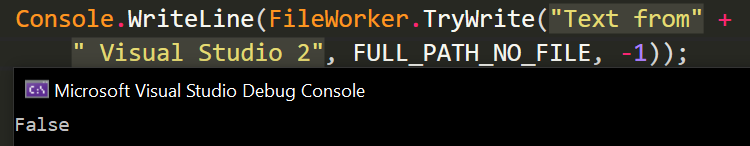


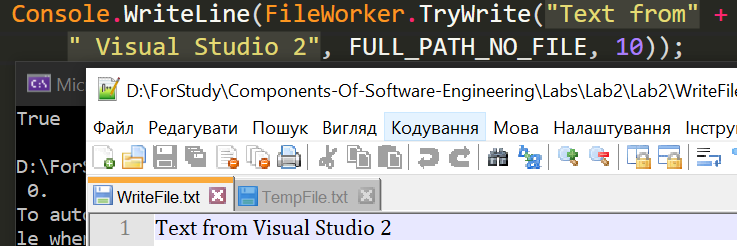
Результат не змінився. Добре, тепер спробуємо записати щось більше:



Результат показує, що функція виконалася успішно. Тепер зробимо те саме, але без екземплярів:

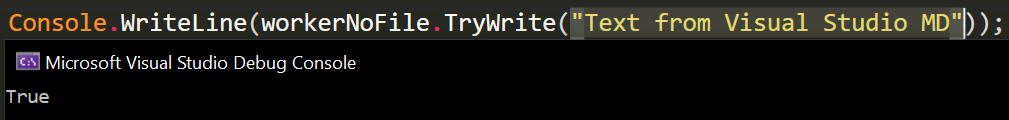




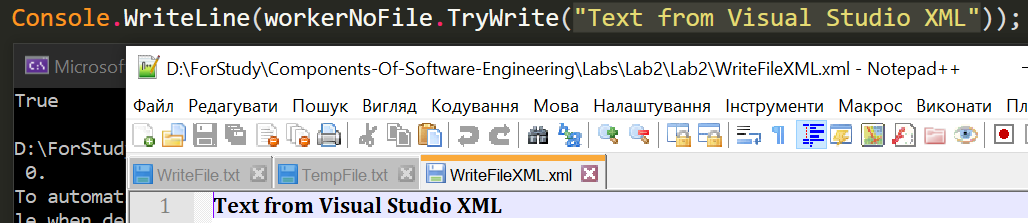


Результати виявилися ідентичними до минулих.

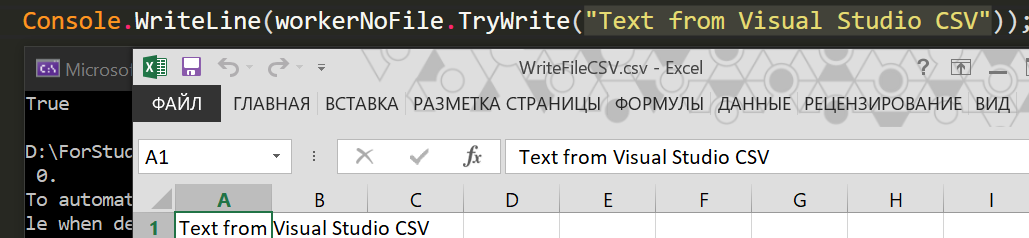
Тепер мене цікавить, які файли можна створювати за допомогою цього методу:



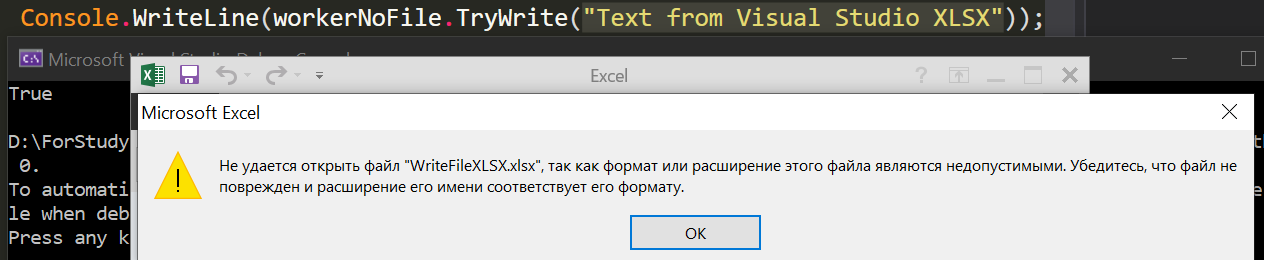
Файл \*.md пройшов.



XML теж.



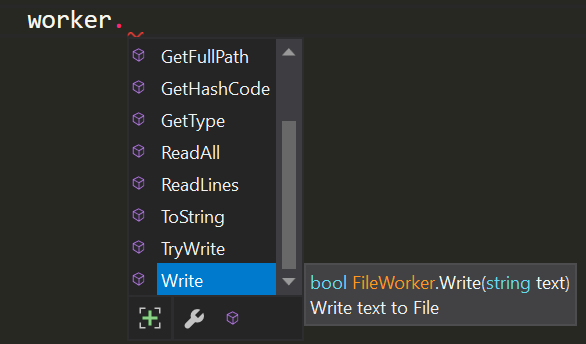
З CSV теж все успішно.



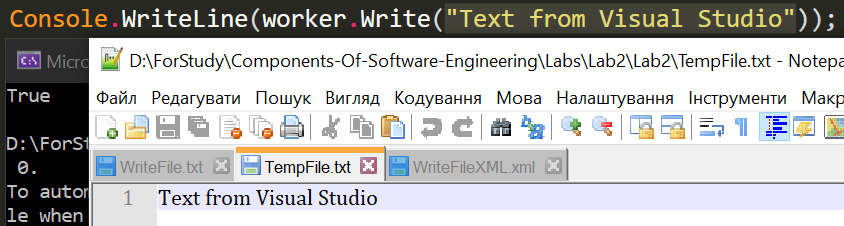
Як бачимо, файл \*.xlsx теж створився, але відкритися не може.

Отже, можемо дійти до висновку, що метод створює будь-який файл з заданим розширенням.

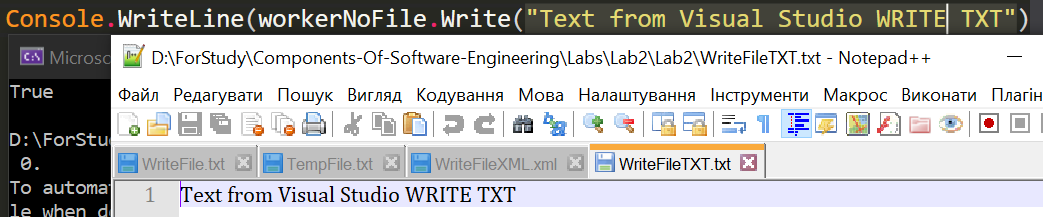
Write

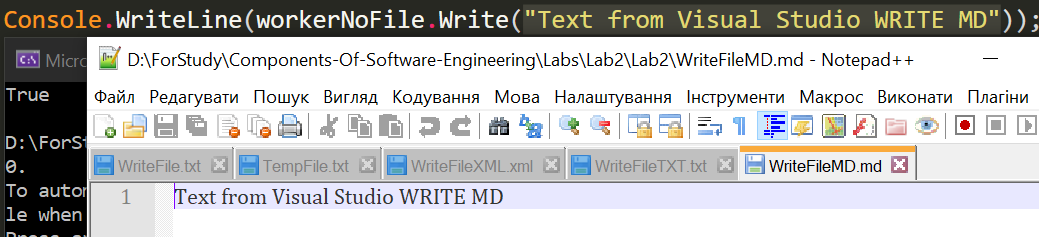


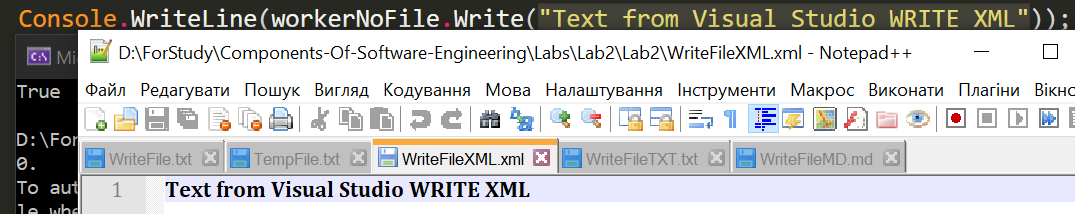
Цей метод дуже схожий на минулий, єдина різниця в тому, що спроб на запис немає.

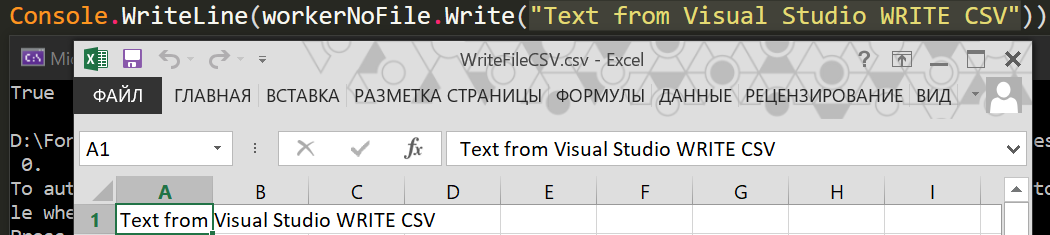


Давайте протестуємо цю команду так само, як і TryWrite():



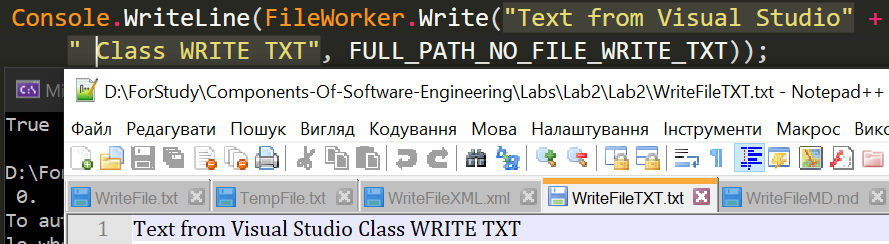


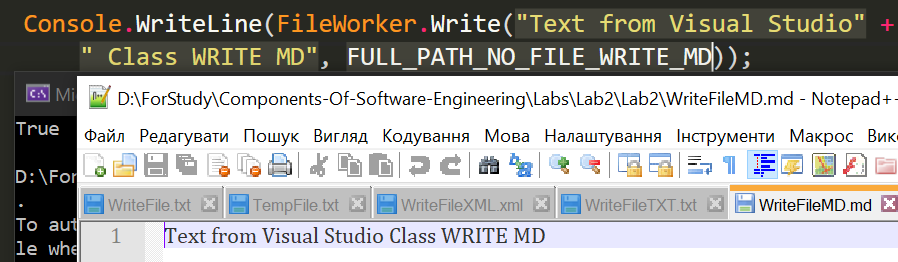


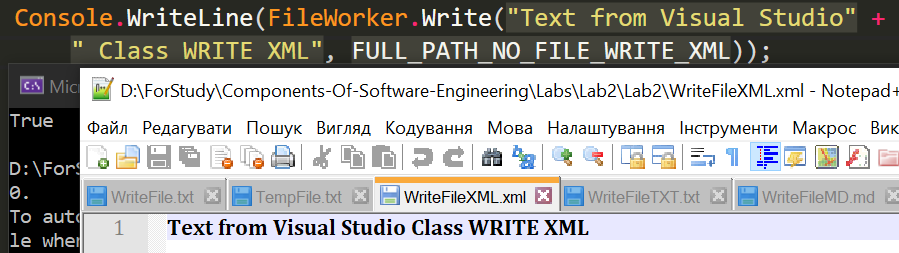


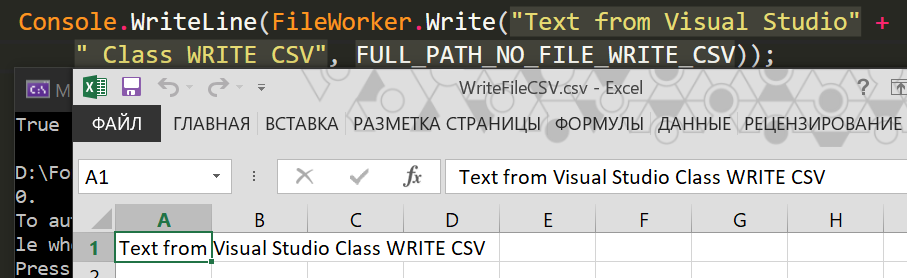
Дійсно, усе ідентично до минулої команди.

Тепер перевіримо виклик через клас:





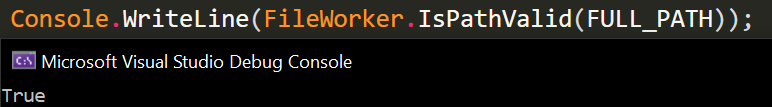




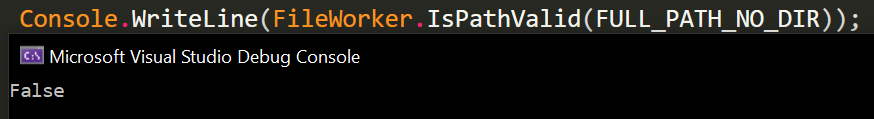
Як бачимо, усе працює.

Це всі команди, які показує Visual Studio, але в xml документі їх більше, тому давайте спробуємо їх написати.

IsPathValid

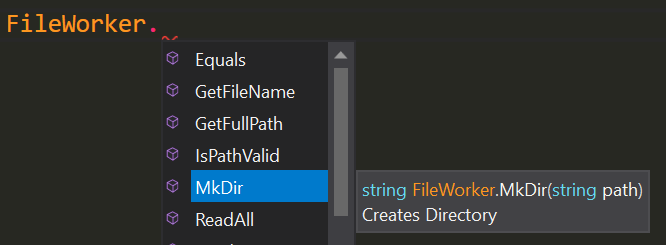


Ми записали як параметр правильний шлях і дійсно, програма розпізнала його. А тепер спробуємо задати невірний шлях:

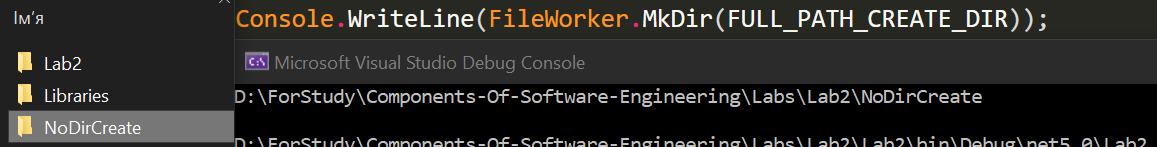


Дійсно, такого шляху не існує.

MkDir

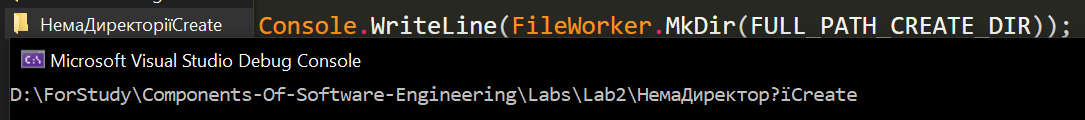


Як бачимо з опису, метод створює директорію. Давайте спробуємо:



Функція спрацювала правильно.

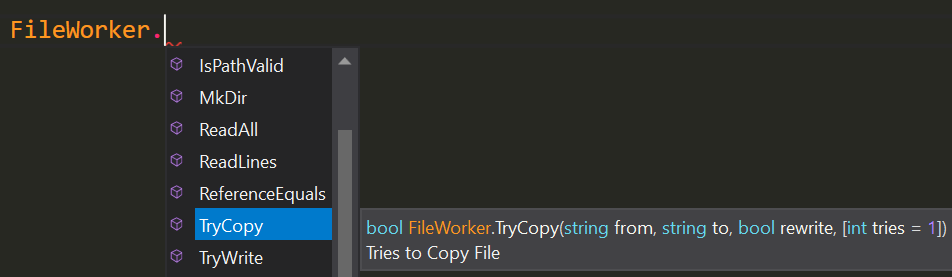
Спробуємо застосувати кирилицю:



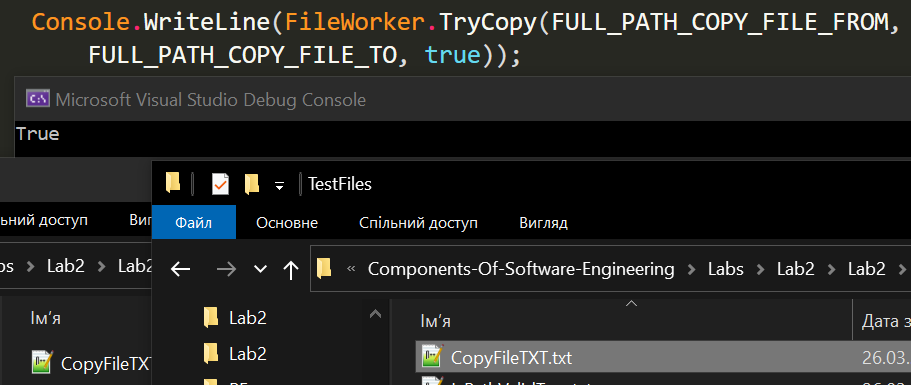
А якщо директорія вже є?

Після повторного запуску програми мені видало те саме, що і минулого разу. Метод побачив, що папка існує і тому просто видав шлях. Файли у цій директорії залишилися.

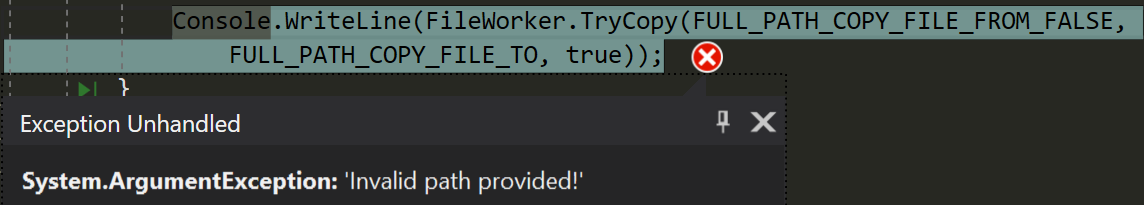
TryCopy

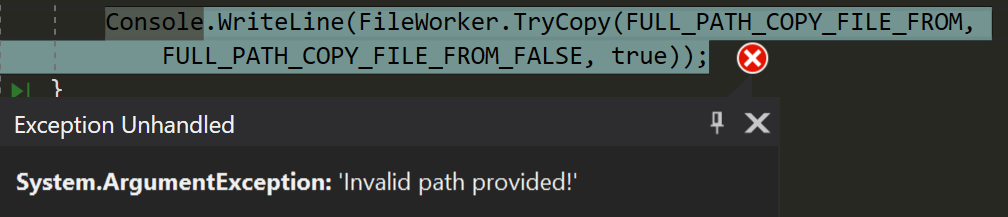
****

Як видно зі скріншоту, даний метод копіює З певного місця В іншу директорію з можливим (але не обов’язковим перезаписом).



Метод вдало виконав Свою роботу. А якщо одного з шляхів не буде існувати?





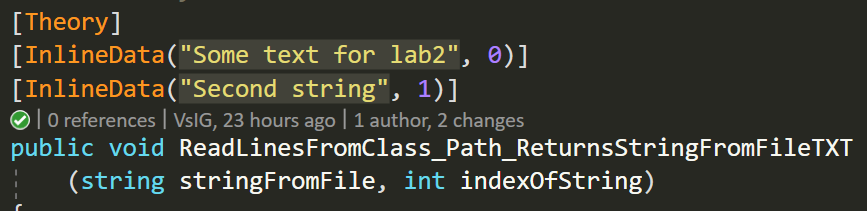
Як видно, visual studio не дає це зробити.

**Тестування**:

Тут хочу розказати деякі речі з якими я стикнувся під час тестування:

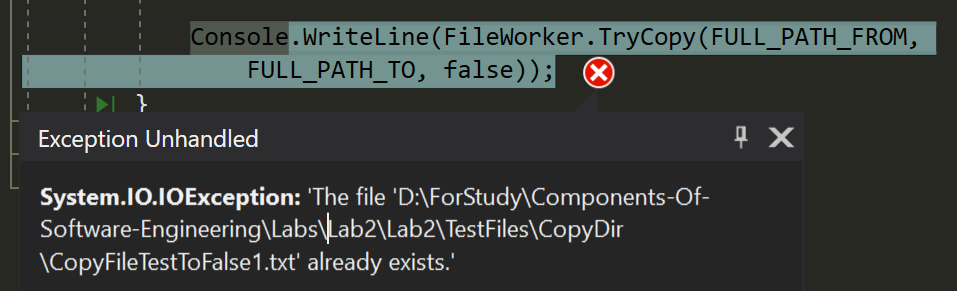
ReadLines

Даний метод повертає масив стрічок, які читає за рядками. [Офіційна документація](https://docs.microsoft.com/en-us/dotnet/core/testing/unit-testing-best-practices#characteristics-of-a-good-unit-test) каже, що в тестах потрібно використовувати якомога менше логіки, саме тому я використав InlineData для вхідних параметрів та замінив [Fact] на [Theory]:



TryCopy

Я стикнувся з проблемою, коли при копіюванні файлу до директорії без перезапису, де він уже є, виникає помилка, яку вдалося виправити лише видаленням файлу після проходження тесту.



Результати тестування



Сирцеві коди:

Program.cs (де проводив досліди):

**using** System**;**

**using** IIG**.**Core**.**FileWorkingUtils**;**

**namespace** Lab2

**{**

class Program

**{**

static void Main**()**

**{**

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TempFile.txt"**;**

const string FULL\_PATH\_MD **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TempFileMD.md"**;**

const string FULL\_PATH\_EXCEL **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TempFileEXCEL.xlsx"**;**

const string FULL\_PATH\_XML **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TempFileXML.xml"**;**

const string FULL\_PATH\_CSV **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TempFileCSV.csv"**;**

const string FULL\_PATH\_NO\_TRY\_FILE **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TryWriteFile.txt"**;**

const string FULL\_PATH\_NO\_DIR **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\NoDir\NoDirFile.txt"**;**

const string FULL\_PATH\_NO\_FILE\_TRY\_MD **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TryWriteFileMD.md"**;**

const string FULL\_PATH\_NO\_FILE\_TRY\_XML **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TryWriteFileXML.xml"**;**

const string FULL\_PATH\_NO\_FILE\_TRY\_CSV **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TryWriteFileCSV.csv"**;**

const string FULL\_PATH\_NO\_FILE\_TRY\_XLSX **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TryWriteFileXLSX.xlsx"**;**

const string FULL\_PATH\_NO\_FILE\_WRITE\_TXT **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\WriteFileTXT.txt"**;**

const string FULL\_PATH\_NO\_FILE\_WRITE\_MD **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\WriteFileMD.md"**;**

const string FULL\_PATH\_NO\_FILE\_WRITE\_XML **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\WriteFileXML.xml"**;**

const string FULL\_PATH\_NO\_FILE\_WRITE\_CSV **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\WriteFileCSV.csv"**;**

const string FULL\_PATH\_NO\_FILE\_WRITE\_XLSX **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\WriteFileXLSX.xlsx"**;**

const string FULL\_PATH\_CREATE\_DIR **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\" **+**

@"TestFiles\NoDirCreate"**;**

const string FULL\_PATH\_COPY\_FILE\_FROM **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\" **+**

@"TestFiles\CopyFileTXT.txt"**;**

const string FULL\_PATH\_COPY\_FILE\_TO **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\" **+**

@"TestFiles\CopyDir\CopyFileTXT.txt"**;**

const string FULL\_PATH\_COPY\_FILE\_FROM\_FALSE **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\" **+**

@"TestFiles\NoDir\CopyFileTXT.txt"**;**

FileWorker worker **=** **new(**FULL\_PATH**);**

FileWorker workerNoFile **=** **new(**FULL\_PATH\_NO\_FILE\_WRITE\_CSV**);**

// FileWorker writeNoDir = new(FULL\_PATH\_NO\_DIR);

// Print field (should be FULL\_PATH)

// Console.WriteLine(worker.FilePath);

/\*

//Tried to change field and failed

worker.FilePath = "D:/ForStudy/Components-Of-" +

"Software-Engineering/Labs/Lab2/Lab2/TempFile1.txt";

\*/

/\*

Console.WriteLine(worker.GetFileName());

Console.WriteLine(FileWorker.GetFileName(FULL\_PATH));

\*/

/\*

Console.WriteLine(worker.GetFullPath());

Console.WriteLine(FileWorker.GetFullPath(FULL\_PATH));

\*/

/\*

Console.WriteLine(worker.ReadAll());

Console.WriteLine(FileWorker.ReadAll(FULL\_PATH));

Console.WriteLine(FileWorker.ReadAll(FULL\_PATH\_MD));

Console.WriteLine(FileWorker.ReadAll(FULL\_PATH\_EXCEL));

Console.WriteLine(FileWorker.ReadAll(FULL\_PATH\_XML));

Console.WriteLine(FileWorker.ReadAll(FULL\_PATH\_CSV));

\*/

/\*

for (int i = 0; i < worker.ReadLines().Length; i++)

{

Console.WriteLine(worker.ReadLines()[i]);

}

for (int i = 0; i < FileWorker.ReadLines(FULL\_PATH).Length; i++)

{

Console.WriteLine(FileWorker.ReadLines(FULL\_PATH)[i]);

}

for (int i = 0; i < FileWorker.ReadLines(FULL\_PATH\_MD).Length; i++)

{

Console.WriteLine(FileWorker.ReadLines(FULL\_PATH\_MD)[i]);

}

for (int i = 0; i < FileWorker.ReadLines(FULL\_PATH\_EXCEL).Length; i++)

{

Console.WriteLine(FileWorker.ReadLines(FULL\_PATH\_EXCEL)[i]);

}

for (int i = 0; i < FileWorker.ReadLines(FULL\_PATH\_XML).Length; i++)

{

Console.WriteLine(FileWorker.ReadLines(FULL\_PATH\_XML)[i]);

}

for (int i = 0; i < FileWorker.ReadLines(FULL\_PATH\_CSV).Length; i++)

{

Console.WriteLine(FileWorker.ReadLines(FULL\_PATH\_CSV)[i]);

}

\*/

/\*

// doesn't work

// Console.WriteLine(FileWorker.TryWrite("Text from " +

// "Visual Studio 2"));

// doesn't work

// Console.WriteLine(FileWorker.TryWrite("Text from Visual" +

// " Studio 2", FULL\_PATH\_NO\_DIR));

Console.WriteLine(workerNoFile.TryWrite("Text from Visual Studio 2", 0));

Console.WriteLine(workerNoFile.TryWrite("Text from Visual Studio 2", -1));

Console.WriteLine(workerNoFile.TryWrite("Text from Visual Studio 2", 10));

Console.WriteLine(FileWorker.TryWrite("Text from" +

" Visual Studio 2", FULL\_PATH\_NO\_FILE, 0));

Console.WriteLine(FileWorker.TryWrite("Text from" +

" Visual Studio 2", FULL\_PATH\_NO\_FILE, -1));

Console.WriteLine(FileWorker.TryWrite("Text from" +

" Visual Studio 2", FULL\_PATH\_NO\_FILE, 10));

Console.WriteLine(workerNoFile.TryWrite("Text from Visual Studio XLSX"));

\*/

/\*

Console.WriteLine(FileWorker.Write("Text from Visual Studio" +

" Class WRITE CSV", FULL\_PATH\_NO\_FILE\_WRITE\_CSV));

\*/

// Console.WriteLine(FileWorker.IsPathValid(FULL\_PATH\_NO\_DIR));

// Console.WriteLine(FileWorker.MkDir(FULL\_PATH\_CREATE\_DIR));

/\*

Console.WriteLine(FileWorker.TryCopy(FULL\_PATH\_COPY\_FILE\_FROM,

FULL\_PATH\_COPY\_FILE\_TO, true));

\*/

**}**

**}**

**}**

TestFileWorkingUtils (тести)

**using** IIG**.**Core**.**FileWorkingUtils**;**

**using** Xunit**;**

**using** System**.**IO**;**

**namespace** TestFileWorkingUtils

**{**

/// <summary>

/// This class contain all tests for

/// library IIG.Core.FileWorkingUtils

/// </summary>

**public** class TestFileWorkingUtils

**{**

/\*

Naming:

1. The name of the method being tested.

2. The scenario under which it's being tested.

3. The expected behavior when the scenario is invoked.

\*/

#region FilePath

/// <summary>

/// Test public property FilePath

/// Check if we get exact string as it is in constructor

/// </summary>

**[**Fact**]**

**public** void FilePath\_None\_ReturnsSameString**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TempFile.txt"**;**

FileWorker worker **=** **new(**FULL\_PATH**);**

string expected **=** worker**.**FilePath**;**

// Act

string actual **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TempFile.txt"**;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

#endregion FilePath

#region GetFileName

/// <summary>

/// Test public method GetFileName using constructor

/// Check if we get correct name of the file

/// </summary>

**[**Fact**]**

**public** void GetFileNameFromConstructor\_None\_ReturnsSameString**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TempFile.txt"**;**

FileWorker worker **=** **new(**FULL\_PATH**);**

string expected **=** worker**.**GetFileName**();**

// Act

string actual **=** "TempFile.txt"**;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

/// <summary>

/// Test public method GetFileName using class

/// Check if we get correct name of the file

/// </summary>

**[**Fact**]**

**public** void GetFileNameFromClass\_Path\_ReturnsSameString**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TempFile.txt"**;**

string expected **=** FileWorker**.**GetFileName**(**FULL\_PATH**);**

// Act

string actual **=** "TempFile.txt"**;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

#endregion GetFileName

#region GetFullPath

/// <summary>

/// Test public method GetFullPath using constructor

/// Check if we get correct full path to the file

/// </summary>

**[**Fact**]**

**public** void GetFullPathFromConstructor\_None\_ReturnsSameString**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TempFile.txt"**;**

FileWorker worker **=** **new(**FULL\_PATH**);**

string expected **=** worker**.**GetFullPath**();**

// Act

string actual **=** FULL\_PATH**;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

/// <summary>

/// Test public method GetFullPath using class

/// Check if we get correct full path to the file

/// </summary>

**[**Fact**]**

**public** void GetFullPathFromClass\_Path\_ReturnsSameString**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TempFile.txt"**;**

string expected **=** FileWorker**.**GetFullPath**(**FULL\_PATH**);**

// Act

string actual **=** FULL\_PATH**;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

#endregion GetFullPath

#region ReadAll

#region ReadAllTXT

/// <summary>

/// Test public method ReadAll using constructor

/// Check if we get correct string from txt file

/// </summary>

**[**Fact**]**

**public** void ReadAllFromConstructor\_None\_ReturnsStringFromFileTXT**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TempFile.txt"**;**

const string STRING\_FROM\_FILE **=** "Some text for lab2\r\nSecond string"**;**

FileWorker worker **=** **new(**FULL\_PATH**);**

string expected **=** worker**.**ReadAll**();**

// Act

string actual **=** STRING\_FROM\_FILE**;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

/// <summary>

/// Test public method ReadAll using class

/// Check if we get correct string from txt file

/// </summary>

**[**Fact**]**

**public** void ReadAllFromClass\_Path\_ReturnsStringFromFileTXT**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TempFile.txt"**;**

const string STRING\_FROM\_FILE **=** "Some text for lab2" **+**

"\r\nSecond string"**;**

string expected **=** FileWorker**.**ReadAll**(**FULL\_PATH**);**

// Act

string actual **=** STRING\_FROM\_FILE**;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

#endregion ReadAllTXT

#region ReadAllMD

/// <summary>

/// Test public method ReadAll using constructor

/// Check if we get correct string from md file

/// </summary>

**[**Fact**]**

**public** void ReadAllFromConstructor\_None\_ReturnsStringFromFileMD**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TempFileMD.md"**;**

const string STRING\_FROM\_FILE **=** "Some text for lab2 MD" **+**

"\r\nSecond string MD"**;**

FileWorker worker **=** **new(**FULL\_PATH**);**

string expected **=** worker**.**ReadAll**();**

// Act

string actual **=** STRING\_FROM\_FILE**;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

/// <summary>

/// Test public method ReadAll using class

/// Check if we get correct string from md file

/// </summary>

**[**Fact**]**

**public** void ReadAllFromClass\_Path\_ReturnsStringFromFileMD**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TempFileMD.md"**;**

const string STRING\_FROM\_FILE **=** "Some text for lab2 MD" **+**

"\r\nSecond string MD"**;**

string expected **=** FileWorker**.**ReadAll**(**FULL\_PATH**);**

// Act

string actual **=** STRING\_FROM\_FILE**;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

#endregion ReadAllMD

#region ReadAllXML

/// <summary>

/// Test public method ReadAll using constructor

/// Check if we get correct string from xml file

/// </summary>

**[**Fact**]**

**public** void ReadAllFromConstructor\_None\_ReturnsStringFromFileXML**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TempFileXML.xml"**;**

const string STRING\_FROM\_FILE **=** "<?xml version=\"1.0\"" **+**

"?>\n<Text>\n <FirstText>Some text for lab2 " **+**

"XML</FirstText>\n <SecondText>Second string " **+**

"XML</SecondText>\n</Text>\n"**;**

FileWorker worker **=** **new(**FULL\_PATH**);**

string expected **=** worker**.**ReadAll**();**

// Act

string actual **=** STRING\_FROM\_FILE**;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

/// <summary>

/// Test public method ReadAll using class

/// Check if we get correct string from xml file

/// </summary>

**[**Fact**]**

**public** void ReadAllFromClass\_Path\_ReturnsStringFromFileXML**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TempFileXML.xml"**;**

const string STRING\_FROM\_FILE **=** "<?xml version=\"1.0\"" **+**

"?>\n<Text>\n <FirstText>Some text for lab2 " **+**

"XML</FirstText>\n <SecondText>Second string " **+**

"XML</SecondText>\n</Text>\n"**;**

string expected **=** FileWorker**.**ReadAll**(**FULL\_PATH**);**

// Act

string actual **=** STRING\_FROM\_FILE**;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

#endregion ReadAllXML

#region ReadAllCSV

/// <summary>

/// Test public method ReadAll using constructor

/// Check if we get correct string from csv file

/// </summary>

**[**Fact**]**

**public** void ReadAllFromConstructor\_None\_ReturnsStringFromFileCSV**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TempFileCSV.csv"**;**

const string STRING\_FROM\_FILE **=** "Some text for lab2,CSV\r\nSecond string,CSV\r\n"**;**

FileWorker worker **=** **new(**FULL\_PATH**);**

string expected **=** worker**.**ReadAll**();**

// Act

string actual **=** STRING\_FROM\_FILE**;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

/// <summary>

/// Test public method ReadAll using class

/// Check if we get correct string from csv file

/// </summary>

**[**Fact**]**

**public** void ReadAllFromClass\_Path\_ReturnsStringFromFileCSV**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TempFileCSV.csv"**;**

const string STRING\_FROM\_FILE **=** "Some text for lab2,CSV\r\nSecond string,CSV\r\n"**;**

string expected **=** FileWorker**.**ReadAll**(**FULL\_PATH**);**

// Act

string actual **=** STRING\_FROM\_FILE**;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

#endregion ReadAllCSV

#endregion ReadAll

#region ReadLines

#region ReadLinesTXT

/// <summary>

/// Test public method ReadLines using constructor

/// Check if we get correct array of strings from txt file

/// </summary>

**[**Theory**]**

**[**InlineData**(**"Some text for lab2"**,** 0**)]**

**[**InlineData**(**"Second string"**,** 1**)]**

**public** void ReadLinesFromConstructor\_None\_ReturnsStringFromFileTXT

**(**string stringFromFile**,** int indexOfString**)**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles\TempFile.txt"**;**

FileWorker worker **=** **new(**FULL\_PATH**);**

string expected **=** worker**.**ReadLines**()[**indexOfString**];**

// Act

string actual **=** stringFromFile**;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

/// <summary>

/// Test public method ReadAll using class

/// Check if we get correct array of strings from txt file

/// </summary>

**[**Theory**]**

**[**InlineData**(**"Some text for lab2"**,** 0**)]**

**[**InlineData**(**"Second string"**,** 1**)]**

**public** void ReadLinesFromClass\_Path\_ReturnsStringFromFileTXT

**(**string stringFromFile**,** int indexOfString**)**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles\TempFile.txt"**;**

string expected **=** FileWorker**.**ReadLines**(**FULL\_PATH**)[**indexOfString**];**

// Act

string actual **=** stringFromFile**;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

#endregion ReadLinesTXT

#region ReadLinesMD

/// <summary>

/// Test public method ReadLines using constructor

/// Check if we get correct array of strings from md file

/// </summary>

**[**Theory**]**

**[**InlineData**(**"Some text for lab2 MD"**,** 0**)]**

**[**InlineData**(**"Second string MD"**,** 1**)]**

**public** void ReadLinesFromConstructor\_None\_ReturnsStringFromFileMD

**(**string stringFromFile**,** int indexOfString**)**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles\TempFileMD.md"**;**

FileWorker worker **=** **new(**FULL\_PATH**);**

string expected **=** worker**.**ReadLines**()[**indexOfString**];**

// Act

string actual **=** stringFromFile**;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

/// <summary>

/// Test public method ReadAll using class

/// Check if we get correct array of strings from md file

/// </summary>

**[**Theory**]**

**[**InlineData**(**"Some text for lab2 MD"**,** 0**)]**

**[**InlineData**(**"Second string MD"**,** 1**)]**

**public** void ReadLinesFromClass\_Path\_ReturnsStringFromFileMD

**(**string stringFromFile**,** int indexOfString**)**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles\TempFileMD.md"**;**

string expected **=** FileWorker**.**ReadLines**(**FULL\_PATH**)[**indexOfString**];**

// Act

string actual **=** stringFromFile**;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

#endregion ReadLinesMD

#region ReadLinesXML

/// <summary>

/// Test public method ReadLines using constructor

/// Check if we get correct array of strings from xml file

/// </summary>

**[**Theory**]**

**[**InlineData**(**"<?xml version=\"1.0\"?>"**,** 0**)]**

**[**InlineData**(**"<Text>"**,** 1**)]**

**[**InlineData**(**" <FirstText>Some text for lab2 XML</FirstText>"**,** 2**)]**

**[**InlineData**(**" <SecondText>Second string XML</SecondText>"**,** 3**)]**

**[**InlineData**(**"</Text>"**,** 4**)]**

**public** void ReadLinesFromConstructor\_None\_ReturnsStringFromFileXML

**(**string stringFromFile**,** int indexOfString**)**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles\TempFileXML.xml"**;**

FileWorker worker **=** **new(**FULL\_PATH**);**

string expected **=** worker**.**ReadLines**()[**indexOfString**];**

// Act

string actual **=** stringFromFile**;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

/// <summary>

/// Test public method ReadAll using class

/// Check if we get correct array of strings from xml file

/// </summary>

**[**Theory**]**

**[**InlineData**(**"<?xml version=\"1.0\"?>"**,** 0**)]**

**[**InlineData**(**"<Text>"**,** 1**)]**

**[**InlineData**(**" <FirstText>Some text for lab2 XML</FirstText>"**,** 2**)]**

**[**InlineData**(**" <SecondText>Second string XML</SecondText>"**,** 3**)]**

**[**InlineData**(**"</Text>"**,** 4**)]**

**public** void ReadLinesFromClass\_Path\_ReturnsStringFromFileXMl

**(**string stringFromFile**,** int indexOfString**)**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles\TempFileXML.xml"**;**

string expected **=** FileWorker**.**ReadLines**(**FULL\_PATH**)[**indexOfString**];**

// Act

string actual **=** stringFromFile**;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

#endregion ReadLinesXML

#region ReadLinesCSV

/// <summary>

/// Test public method ReadLines using constructor

/// Check if we get correct array of strings from csv file

/// </summary>

**[**Theory**]**

**[**InlineData**(**"Some text for lab2,CSV"**,** 0**)]**

**[**InlineData**(**"Second string,CSV"**,** 1**)]**

**public** void ReadLinesFromConstructor\_None\_ReturnsStringFromFileCSV

**(**string stringFromFile**,** int indexOfString**)**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles\TempFileCSV.csv"**;**

FileWorker worker **=** **new(**FULL\_PATH**);**

string expected **=** worker**.**ReadLines**()[**indexOfString**];**

// Act

string actual **=** stringFromFile**;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

/// <summary>

/// Test public method ReadAll using class

/// Check if we get correct array of strings from csv file

/// </summary>

**[**Theory**]**

**[**InlineData**(**"Some text for lab2,CSV"**,** 0**)]**

**[**InlineData**(**"Second string,CSV"**,** 1**)]**

**public** void ReadLinesFromClass\_Path\_ReturnsStringFromFileCSV

**(**string stringFromFile**,** int indexOfString**)**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles\TempFileCSV.csv"**;**

string expected **=** FileWorker**.**ReadLines**(**FULL\_PATH**)[**indexOfString**];**

// Act

string actual **=** stringFromFile**;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

#endregion ReadLinesCSV

#endregion ReadLines

#region TryWrite

#region TryWriteTXT

/// <summary>

/// Test public method TryWrite using constructor

/// Check if we create txt file

/// </summary>

**[**Fact**]**

**public** void TryWriteTXTFromConstructor\_None\_ReturnsTrue**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\TestTryWriteFileConstructorTXT.txt"**;**

const string STRING\_TO\_FILE **=** "Text from test Constructor TXT"**;**

FileWorker worker **=** **new(**FULL\_PATH**);**

bool expected **=** worker**.**TryWrite**(**STRING\_TO\_FILE**);**

// Act

bool actual **=** **true;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

/// <summary>

/// Test public method TryWrite using class

/// Check if we create txt file

/// </summary>

**[**Fact**]**

**public** void TryWriteTXTFromClass\_Path\_ReturnsTrue**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\TestTryWriteFileClassTXT.txt"**;**

const string STRING\_TO\_FILE **=** "Text from test Class TXT"**;**

bool expected **=** FileWorker**.**TryWrite**(**STRING\_TO\_FILE**,** FULL\_PATH**);**

// Act

bool actual **=** **true;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

#endregion TryWriteTXT

#region TryWriteMD

/// <summary>

/// Test public method TryWrite using constructor

/// Check if we create txt file

/// </summary>

**[**Fact**]**

**public** void TryWriteMDFromConstructor\_None\_ReturnsTrue**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\TestTryWriteFileConstructorMD.md"**;**

const string STRING\_TO\_FILE **=** "Text from test Constructor MD"**;**

FileWorker worker **=** **new(**FULL\_PATH**);**

bool expected **=** worker**.**TryWrite**(**STRING\_TO\_FILE**);**

// Act

bool actual **=** **true;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

/// <summary>

/// Test public method TryWrite using class

/// Check if we create md file

/// </summary>

**[**Fact**]**

**public** void TryWriteMDFromClass\_Path\_ReturnsTrue**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\TestTryWriteFileClassMD.md"**;**

const string STRING\_TO\_FILE **=** "Text from test Class MD"**;**

bool expected **=** FileWorker**.**TryWrite**(**STRING\_TO\_FILE**,** FULL\_PATH**);**

// Act

bool actual **=** **true;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

#endregion TryWriteMD

#region TryWriteXML

/// <summary>

/// Test public method TryWrite using constructor

/// Check if we create xml file

/// </summary>

**[**Fact**]**

**public** void TryWriteXMLFromConstructor\_None\_ReturnsTrue**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\TestTryWriteFileConstructorXML.xml"**;**

const string STRING\_TO\_FILE **=** "Text from test Constructor XML"**;**

FileWorker worker **=** **new(**FULL\_PATH**);**

bool expected **=** worker**.**TryWrite**(**STRING\_TO\_FILE**);**

// Act

bool actual **=** **true;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

/// <summary>

/// Test public method TryWrite using class

/// Check if we create xml file

/// </summary>

**[**Fact**]**

**public** void TryWriteXMLFromClass\_Path\_ReturnsTrue**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\TestTryWriteFileClassXML.xml"**;**

const string STRING\_TO\_FILE **=** "Text from test Class XML"**;**

bool expected **=** FileWorker**.**TryWrite**(**STRING\_TO\_FILE**,** FULL\_PATH**);**

// Act

bool actual **=** **true;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

#endregion TryWriteXML

#region TryWriteCSV

/// <summary>

/// Test public method TryWrite using constructor

/// Check if we create csv file

/// </summary>

**[**Fact**]**

**public** void TryWriteCSVFromConstructor\_None\_ReturnsTrue**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\TestTryWriteFileConstructorCSV.csv"**;**

const string STRING\_TO\_FILE **=** "Text from test Constructor CSV"**;**

FileWorker worker **=** **new(**FULL\_PATH**);**

bool expected **=** worker**.**TryWrite**(**STRING\_TO\_FILE**);**

// Act

bool actual **=** **true;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

/// <summary>

/// Test public method TryWrite using class

/// Check if we create csv file

/// </summary>

**[**Fact**]**

**public** void TryWriteCSVFromClass\_Path\_ReturnsTrue**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\TestTryWriteFileClassCSV.csv"**;**

const string STRING\_TO\_FILE **=** "Text from test Class CSV"**;**

bool expected **=** FileWorker**.**TryWrite**(**STRING\_TO\_FILE**,** FULL\_PATH**);**

// Act

bool actual **=** **true;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

#endregion TryWriteCSV

#region TryWriteZeroTries

/// <summary>

/// Test public method TryWrite using constructor

/// Check if we can't create txt file with Zero tries

/// </summary>

**[**Fact**]**

**public** void TryWriteTXTFromConstructorZeroTries\_None\_ReturnsFalse**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\TestTryWriteFileConstructorZeroTriesTXT.txt"**;**

const string STRING\_TO\_FILE **=** "Text from test " **+**

"Zero Tries Constructor TXT"**;**

FileWorker worker **=** **new(**FULL\_PATH**);**

bool expected **=** worker**.**TryWrite**(**STRING\_TO\_FILE**,** 0**);**

// Act

bool actual **=** **false;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

/// <summary>

/// Test public method TryWrite using class

/// Check if we can't create txt file with Zero tries

/// </summary>

**[**Fact**]**

**public** void TryWriteTXTFromClassZeroTries\_Path\_ReturnsFalse**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\TestTryWriteFileClassZeroTriesTXT.txt"**;**

const string STRING\_TO\_FILE **=** "Text from test " **+**

"Zero Tries Class TXT"**;**

bool expected **=** FileWorker**.**TryWrite**(**STRING\_TO\_FILE**,** FULL\_PATH**,** 0**);**

// Act

bool actual **=** **false;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

#endregion TryWriteZeroTries

#region TryWriteNegativeTries

/// <summary>

/// Test public method TryWrite using constructor

/// Check if we can't create txt file with Negative tries

/// </summary>

**[**Fact**]**

**public** void TryWriteTXTFromConstructorNegativeTries\_None\_ReturnsFalse**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\TestTryWriteFileConstructorNegativeTriesTXT.txt"**;**

const string STRING\_TO\_FILE **=** "Text from test " **+**

"Negative Tries Constructor TXT"**;**

FileWorker worker **=** **new(**FULL\_PATH**);**

bool expected **=** worker**.**TryWrite**(**STRING\_TO\_FILE**,** **-**1**);**

// Act

bool actual **=** **false;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

/// <summary>

/// Test public method TryWrite using class

/// Check if we can't create txt file with Negative tries

/// </summary>

**[**Fact**]**

**public** void TryWriteTXTFromClassNegativeTries\_Path\_ReturnsFalse**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\TestTryWriteFileClassNegativeTriesTXT.txt"**;**

const string STRING\_TO\_FILE **=** "Text from test " **+**

"Negative Tries Class TXT"**;**

bool expected **=** FileWorker**.**TryWrite**(**STRING\_TO\_FILE**,** FULL\_PATH**,** **-**1**);**

// Act

bool actual **=** **false;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

#endregion TryWriteZeroTries

#region TryWriteManyTries

/// <summary>

/// Test public method TryWrite using constructor

/// Check if we can create txt file with Many tries

/// </summary>

**[**Fact**]**

**public** void TryWriteTXTFromConstructorManyTries\_None\_ReturnsTrue**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\TestTryWriteFileConstructorManyTriesTXT.txt"**;**

const string STRING\_TO\_FILE **=** "Text from test " **+**

"Many Tries Constructor TXT"**;**

FileWorker worker **=** **new(**FULL\_PATH**);**

bool expected **=** worker**.**TryWrite**(**STRING\_TO\_FILE**,** 10**);**

// Act

bool actual **=** **true;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

/// <summary>

/// Test public method TryWrite using class

/// Check if we can create txt file with Many tries

/// </summary>

**[**Fact**]**

**public** void TryWriteTXTFromClassManyTries\_Path\_ReturnsTrue**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\TestTryWriteFileClassManyTriesTXT.txt"**;**

const string STRING\_TO\_FILE **=** "Text from test " **+**

"Many Tries Class TXT"**;**

bool expected **=** FileWorker**.**TryWrite**(**STRING\_TO\_FILE**,** FULL\_PATH**,** 10**);**

// Act

bool actual **=** **true;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

#endregion TryWriteManyTries

#endregion TryWrite

#region Write

#region WriteTXT

/// <summary>

/// Test public method Write using constructor

/// Check if we create txt file

/// </summary>

**[**Fact**]**

**public** void WriteTXTFromConstructor\_None\_ReturnsTrue**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\TestWriteFileConstructorTXT.txt"**;**

const string STRING\_TO\_FILE **=** "Text from test Constructor TXT"**;**

FileWorker worker **=** **new(**FULL\_PATH**);**

bool expected **=** worker**.**TryWrite**(**STRING\_TO\_FILE**);**

// Act

bool actual **=** **true;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

/// <summary>

/// Test public method Write using class

/// Check if we create txt file

/// </summary>

**[**Fact**]**

**public** void WriteTXTFromClass\_Path\_ReturnsTrue**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\TestWriteFileClassTXT.txt"**;**

const string STRING\_TO\_FILE **=** "Text from test Class TXT"**;**

bool expected **=** FileWorker**.**TryWrite**(**STRING\_TO\_FILE**,** FULL\_PATH**);**

// Act

bool actual **=** **true;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

#endregion WriteTXT

#region WriteMD

/// <summary>

/// Test public method Write using constructor

/// Check if we create md file

/// </summary>

**[**Fact**]**

**public** void WriteMDFromConstructor\_None\_ReturnsTrue**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\TestWriteFileConstructorMD.md"**;**

const string STRING\_TO\_FILE **=** "Text from test Constructor MD"**;**

FileWorker worker **=** **new(**FULL\_PATH**);**

bool expected **=** worker**.**TryWrite**(**STRING\_TO\_FILE**);**

// Act

bool actual **=** **true;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

/// <summary>

/// Test public method Write using class

/// Check if we create md file

/// </summary>

**[**Fact**]**

**public** void WriteMDFromClass\_Path\_ReturnsTrue**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\TestWriteFileClassMD.md"**;**

const string STRING\_TO\_FILE **=** "Text from test Class MD"**;**

bool expected **=** FileWorker**.**TryWrite**(**STRING\_TO\_FILE**,** FULL\_PATH**);**

// Act

bool actual **=** **true;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

#endregion WriteMD

#region WriteXML

/// <summary>

/// Test public method Write using constructor

/// Check if we create xml file

/// </summary>

**[**Fact**]**

**public** void WriteXMLFromConstructor\_None\_ReturnsTrue**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\TestWriteFileConstructorXML.xml"**;**

const string STRING\_TO\_FILE **=** "Text from test Constructor XML"**;**

FileWorker worker **=** **new(**FULL\_PATH**);**

bool expected **=** worker**.**TryWrite**(**STRING\_TO\_FILE**);**

// Act

bool actual **=** **true;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

/// <summary>

/// Test public method Write using class

/// Check if we create xml file

/// </summary>

**[**Fact**]**

**public** void WriteXMLFromClass\_Path\_ReturnsTrue**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\TestWriteFileClassXML.xml"**;**

const string STRING\_TO\_FILE **=** "Text from test Class XML"**;**

bool expected **=** FileWorker**.**TryWrite**(**STRING\_TO\_FILE**,** FULL\_PATH**);**

// Act

bool actual **=** **true;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

#endregion WriteXML

#region WriteCSV

/// <summary>

/// Test public method Write using constructor

/// Check if we create csv file

/// </summary>

**[**Fact**]**

**public** void WriteCSVFromConstructor\_None\_ReturnsTrue**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\TestWriteFileConstructorCSV.csv"**;**

const string STRING\_TO\_FILE **=** "Text from test Constructor CSV"**;**

FileWorker worker **=** **new(**FULL\_PATH**);**

bool expected **=** worker**.**TryWrite**(**STRING\_TO\_FILE**);**

// Act

bool actual **=** **true;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

/// <summary>

/// Test public method Write using class

/// Check if we create csv file

/// </summary>

**[**Fact**]**

**public** void WriteCSVFromClass\_Path\_ReturnsTrue**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\TestWriteFileClassCSV.csv"**;**

const string STRING\_TO\_FILE **=** "Text from test Class CSV"**;**

bool expected **=** FileWorker**.**TryWrite**(**STRING\_TO\_FILE**,** FULL\_PATH**);**

// Act

bool actual **=** **true;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

#endregion WriteCSV

#endregion Write

#region IsPathValid

/// <summary>

/// Test public method IsPathValid

/// Check if the path is valid

/// </summary>

**[**Fact**]**

**public** void IsPathValid\_Path\_ReturnsTrue**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\IsPathValidTrue.txt"**;**

bool expected **=** FileWorker**.**IsPathValid**(**FULL\_PATH**);**

// Act

bool actual **=** **true;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

/// <summary>

/// Test public method IsPathValid

/// Check if the path is not valid

/// </summary>

**[**Fact**]**

**public** void IsPathValid\_Path\_ReturnsFalse**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\IsPathValidFalse.txt"**;**

bool expected **=** FileWorker**.**IsPathValid**(**FULL\_PATH**);**

// Act

bool actual **=** **true;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

#endregion IsPathValid

#region MkDir

/// <summary>

/// Test public method MkDir

/// Check if the path is created

/// </summary>

**[**Fact**]**

**public** void MkDirLatin\_Path\_ReturnsPathString**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\NoDirCreate"**;**

string expected **=** FileWorker**.**MkDir**(**FULL\_PATH**);**

// Act

string actual **=** FULL\_PATH**;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

/// <summary>

/// Test public method MkDir

/// Check if the path with cyrillic is created

/// </summary>

**[**Fact**]**

**public** void MkDirCyrillic\_Path\_ReturnsPathString**()**

**{**

// Arrange

const string FULL\_PATH **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\НемаДиректоріїCreate"**;**

string expected **=** FileWorker**.**MkDir**(**FULL\_PATH**);**

// Act

string actual **=** FULL\_PATH**;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

#endregion IsPathValid

#region TryCopy

/// <summary>

/// Test public method TryCopy Latin

/// Check if file can be created without rewriting

/// </summary>

**[**Fact**]**

**public** void TryCopyLatinNoRewrite\_Path\_ReturnsTrue**()**

**{**

// Arrange

const string FULL\_PATH\_FROM **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\CopyFileTestFromTrue1.txt"**;**

const string FULL\_PATH\_TO **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\CopyDir\CopyFileTestToTrue1.txt"**;**

bool expected **=** FileWorker**.**TryCopy**(**FULL\_PATH\_FROM**,**

FULL\_PATH\_TO**,** **false);**

// Act

bool actual **=** **true;**

File**.**Delete**(**FULL\_PATH\_TO**);**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

/// <summary>

/// Test public method TryCopy Latin

/// Check if file can be created with rewriting

/// </summary>

**[**Fact**]**

**public** void TryCopyLatinYesRewrite\_Path\_ReturnsTrue**()**

**{**

// Arrange

const string FULL\_PATH\_FROM **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\CopyFileTestFromTrue2.txt"**;**

const string FULL\_PATH\_TO **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\CopyDir\CopyFileTestToTrue2.txt"**;**

bool expected **=** FileWorker**.**TryCopy**(**FULL\_PATH\_FROM**,**

FULL\_PATH\_TO**,** **true);**

// Act

bool actual **=** **true;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

/// <summary>

/// Test public method TryCopy Cyrillic

/// Check if file can be created without rewriting

/// </summary>

**[**Fact**]**

**public** void TryCopyCyrillicNoRewrite\_Path\_ReturnsTrue**()**

**{**

// Arrange

const string FULL\_PATH\_FROM **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\ФайлДляКопіюванняTestFromTrue3.txt"**;**

const string FULL\_PATH\_TO **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\CopyDir\ФайлДляКопіюванняTestToTrue3.txt"**;**

bool expected **=** FileWorker**.**TryCopy**(**FULL\_PATH\_FROM**,**

FULL\_PATH\_TO**,** **false);**

// Act

bool actual **=** **true;**

File**.**Delete**(**FULL\_PATH\_TO**);**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

/// <summary>

/// Test public method TryCopy Cyrillic

/// Check if file can be created with rewriting

/// </summary>

**[**Fact**]**

**public** void TryCopyCyrillicYesRewrite\_Path\_ReturnsTrue**()**

**{**

// Arrange

const string FULL\_PATH\_FROM **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\ФайлДляКопіюванняTestFromTrue4.txt"**;**

const string FULL\_PATH\_TO **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\CopyDir\ФайлДляКопіюванняTestToTrue4.txt"**;**

bool expected **=** FileWorker**.**TryCopy**(**FULL\_PATH\_FROM**,**

FULL\_PATH\_TO**,** **true);**

// Act

bool actual **=** **true;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

/// <summary>

/// Test public method TryCopy Latin

/// Check if file can not be created without rewriting

/// </summary>

**[**Fact**]**

**public** void TryCopyLatinNoRewrite\_Path\_ReturnsFalse**()**

**{**

// Arrange

const string FULL\_PATH\_FROM **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\SomeRandomFile.txt"**;**

const string FULL\_PATH\_TO **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\CopyDir\SomeRandomFile.txt"**;**

bool expected **=** FileWorker**.**TryCopy**(**FULL\_PATH\_FROM**,**

FULL\_PATH\_TO**,** **false);**

// Act

bool actual **=** **false;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

/// <summary>

/// Test public method TryCopy Cyrillic

/// Check if file can not be created with rewriting

/// </summary>

**[**Fact**]**

**public** void TryCopyCyrillicYesRewrite\_Path\_ReturnsFalse**()**

**{**

// Arrange

const string FULL\_PATH\_FROM **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\ВипадковийФайл.txt"**;**

const string FULL\_PATH\_TO **=** @"D:\ForStudy\Components-Of-" **+**

@"Software-Engineering\Labs\Lab2\Lab2\TestFiles" **+**

@"\CopyDir\ВипадковийФайл.txt"**;**

bool expected **=** FileWorker**.**TryCopy**(**FULL\_PATH\_FROM**,**

FULL\_PATH\_TO**,** **true);**

// Act

bool actual **=** **false;**

// Assert

Assert**.**Equal**(**actual**,** expected**);**

**}**

#endregion TryCopy

**}**

**}**

Висновки:

Виконавши цю лабораторну роботу я познайомився з Black Box Testing:

* Тестування, як функціональне, так і нефункціональне, не передбачає знань внутрішньої будови компонента або системи
* Тест-дизайн, базований на техніці чорного ящику – процедура для написання або вибору тест-кейсів на базі аналізу функціональної або нефункціональної специфікації компонента або системи без знання про її внутрішній устрій.

Таким чином, ми не маючи уяви про внутрішню структуру та устрій системи маємо концентруватися на тому, ***що*** програма робить, а не на тому, ***як*** вона це робить.

А щоб зрозуміти, які методи та поля існують, Ми повинні були використовувати \*.xml документ та підказки Visual Studio.

Щодо технік тестування, які **були** використані в *більшості* тестів:

1. Оскільки вся бібліотека працює з файловою системою (створення, редагування, зчитування файлів, перевірка шляхів), то перевірити її роботу за допомогою *статичної* техніки буде складно. Отже треба використовувати **динамічний** варіант.
2. Використовуючи BBT було складно зрозуміти, як можна зламати ті чи інші тести, тому що Ми не знаємо, як саме написаний код. Через це для даної ситуації можна дотримуватися **Test To Pass**, тобто писати малі тести, не намагаючись знайти баг (хоча такі спроби були, але не призвели до їх виявлення)

Техніки, які **не** **були** використані:

1. **Класи еквівалентності та граничні значення**. Оскільки дана бібліотека не використовує якісь конкретні числа або ж змінні, то максимум як можна поділити код на класи – це вірні значення та невірні (тобто, наприклад, існує шлях чи ні. Або ж чи дійсно є певна стрічка у файлі). Через цю ж причину визначити граничні значення просто неможливо
2. **STD**. При роботі з бібліотекою Я не помітив певних станів або ж переходів. Дана техніка може бути корисною, якщо Ми тестуємо велику систему, де є кілька можливих варіантів кінцевих результатів
3. **Статична техніка**. Як Я і казав у першому пункті використаних технік, дана бібліотека працює з файловою системою, тому перевірити її роботу даним способом буде складно
4. **Аналіз доменної області**. Дана техніка неможлива у цій роботі, оскільки базується на класах еквівалентності та граничних значеннях, яких у Нас немає

Джерела:

* Github - <https://github.com/VsIG-official/Components-Of-Software-Engineering>
* Program.cs - <https://github.com/VsIG-official/Components-Of-Software-Engineering/blob/master/Labs/Lab2/Lab2/Program.cs>
* TestFileWorkingUtils.cs - <https://github.com/VsIG-official/Components-Of-Software-Engineering/blob/master/Labs/Lab2/TestFileWorkingUtils/TestFileWorkingUtils.cs>
* Офіційна документація - [https://docs.microsoft.com/en-us/dotnet/core/testing/unit-testing-best-practices#characteristics-of-a-good-unit-test](https://docs.microsoft.com/en-us/dotnet/core/testing/unit-testing-best-practices%23characteristics-of-a-good-unit-test)
* Лекція по темі Black Box Testing - [https://docs.google.com/presentation/d/1zDBgNUV73ja\_yShIY-\_WpTGkktol7DOE/edit#slide=id.p18](https://docs.google.com/presentation/d/1zDBgNUV73ja_yShIY-_WpTGkktol7DOE/edit%23slide=id.p18%20)