**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 №4**

з дисципліни

«Компоненти програмної інженерії. Якість та тестування програмного забезпечення»

на тему

«Провести інтеграційне тестування модульної системи»

Виконав:

Перевірив:

студент групи ІП-93

Бабарикін Ігор Владиславович

Домінський Валентин Олексійович

номер залікової книжки: 9311

номер у списку: 9

Київ 2021

Зміст

[Мета: 3](#_Toc70932308)

[Завдання: 3](#_Toc70932309)

[Хід роботи: 3](#_Toc70932310)

[Початок роботи: 3](#_Toc70932311)

[Приватні поля 4](#_Toc70932312)

[Init 4](#_Toc70932313)

[GetHash 6](#_Toc70932314)

[HashSha2 8](#_Toc70932315)

[Adler32CheckSum 9](#_Toc70932316)

[Тестування: 11](#_Toc70932317)

[Init 11](#_Toc70932318)

[Execution Route 0\_1\_6 11](#_Toc70932319)

[Execution Route 0\_1\_5\_6 13](#_Toc70932320)

[Execution Route 0\_1\_2\_4\_6 15](#_Toc70932321)

[Execution Route 0\_1\_2\_3\_4\_6 16](#_Toc70932322)

[Execution Route 0\_1\_2\_3\_4\_5\_6 17](#_Toc70932323)

[Execution Route 0\_1\_2\_4\_5\_6 18](#_Toc70932324)

[GetHash 19](#_Toc70932325)

[Execution Route 0\_1\_2\_4\_7 19](#_Toc70932326)

[Execution Route 0\_1\_2\_3\_6\_7 20](#_Toc70932327)

[Execution Route 0\_1\_2\_3\_5\_6\_7 21](#_Toc70932328)

[Результати тестування 22](#_Toc70932329)

[Сирцеві коди: 23](#_Toc70932330)

[TestPasswordHashingUtils (тести) 23](#_Toc70932331)

[Висновки: 30](#_Toc70932332)

[Джерела: 30](#_Toc70932333)

Мета:

Провести інтеграційне тестування модульної системи

Завдання:

|  |  |  |  |
| --- | --- | --- | --- |
| N п/п | Nz mod 6 | Input | Output |
| 1 | 0, 3 | [IIG.PasswordHashingUtils (.dll)](https://drive.google.com/drive/u/1/folders/19DgnFmAXtl_Q1qUgVP4xoucQzqLqDzEY)  [IIG.BinaryFlag (project)](https://github.com/IGORGOD/IIG.BinaryFlag) | File ([.dll](https://drive.google.com/drive/u/1/folders/12mKELhYUBLMVmqkItTxemrLNo0w3cQm8))  Database ([.dll](https://drive.google.com/drive/u/1/folders/1znKlrrJsmYBV2wbaFMBwGcqaZekgb6XG) + [.cs](https://drive.google.com/drive/u/1/folders/1D628B-fdROwQxis7muHQ3fQF64i0YEly) + [.bak](https://drive.google.com/drive/u/1/folders/1wRhiEFUiQSOsZ_v8jdD6_Q-ZlLGFPUWV)) |
| 2 | 1, 4 | [IIG.BinaryFlag (.dll)](https://drive.google.com/drive/u/1/folders/1HLsbKf8k27RNVzDhiJkskLX6ypXO5t2u)  [IIG.PasswordHashingUtils (project)](https://github.com/IGORGOD/IIG.PasswordHashingUtils) | File ([.dll](https://drive.google.com/drive/u/1/folders/12mKELhYUBLMVmqkItTxemrLNo0w3cQm8))  Database ([.dll](https://drive.google.com/drive/u/1/folders/1znKlrrJsmYBV2wbaFMBwGcqaZekgb6XG) + [.cs](https://drive.google.com/drive/u/1/folders/1D628B-fdROwQxis7muHQ3fQF64i0YEly) + [.bak](https://drive.google.com/drive/u/1/folders/1wRhiEFUiQSOsZ_v8jdD6_Q-ZlLGFPUWV)) |
| 3 | 2 | [IIG.FileWorker (.dll)](https://drive.google.com/drive/u/1/folders/12mKELhYUBLMVmqkItTxemrLNo0w3cQm8)  [IIG.BinaryFlag (project)](https://github.com/IGORGOD/IIG.BinaryFlag) | Database ([.dll](https://drive.google.com/drive/u/1/folders/1znKlrrJsmYBV2wbaFMBwGcqaZekgb6XG) + [.cs](https://drive.google.com/drive/u/1/folders/1D628B-fdROwQxis7muHQ3fQF64i0YEly) + [.bak](https://drive.google.com/drive/u/1/folders/1wRhiEFUiQSOsZ_v8jdD6_Q-ZlLGFPUWV)) |
| 4 | 5 | [IIG.FileWorker (.dll)](https://drive.google.com/drive/u/1/folders/12mKELhYUBLMVmqkItTxemrLNo0w3cQm8)  [IIG.PasswordHashingUtils (project)](https://github.com/IGORGOD/IIG.PasswordHashingUtils) | Database ([.dll](https://drive.google.com/drive/u/1/folders/1znKlrrJsmYBV2wbaFMBwGcqaZekgb6XG) + [.cs](https://drive.google.com/drive/u/1/folders/1D628B-fdROwQxis7muHQ3fQF64i0YEly) + [.bak](https://drive.google.com/drive/u/1/folders/1wRhiEFUiQSOsZ_v8jdD6_Q-ZlLGFPUWV)) |

Варіант = 9311 mod 6 = 5, отже Мій варіант:

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | 5 | [IIG.FileWorker (.dll)](https://drive.google.com/drive/u/1/folders/12mKELhYUBLMVmqkItTxemrLNo0w3cQm8)  [IIG.PasswordHashingUtils (project)](https://github.com/IGORGOD/IIG.PasswordHashingUtils) | Database ([.dll](https://drive.google.com/drive/u/1/folders/1znKlrrJsmYBV2wbaFMBwGcqaZekgb6XG) + [.cs](https://drive.google.com/drive/u/1/folders/1D628B-fdROwQxis7muHQ3fQF64i0YEly) + [.bak](https://drive.google.com/drive/u/1/folders/1wRhiEFUiQSOsZ_v8jdD6_Q-ZlLGFPUWV)) |

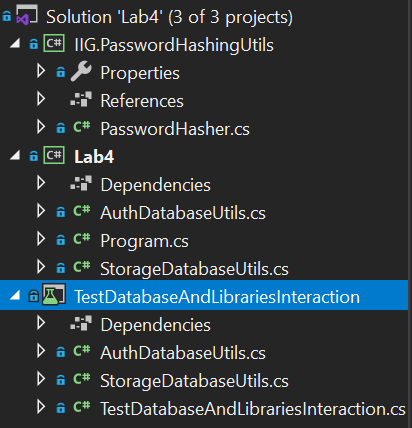
Отже, мій вибір для лабораторної:

* .NET 5
* Бібліотека для тестування xUnit
* Бібліотеки IIG.FileWorker (.dll) та IIG.PasswordHashingUtils (project)

Хід роботи:

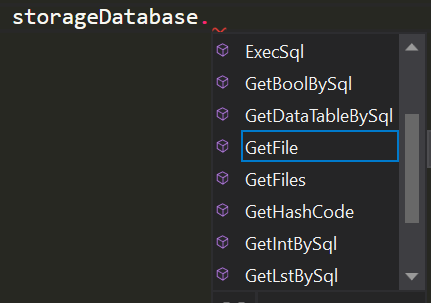
Початок роботи:

Я створив проект “Lab4” на .NET 5, додав xUnit Test Project “TestDatabaseAndLibrariesInteraction” та файли для роботи з базами даних

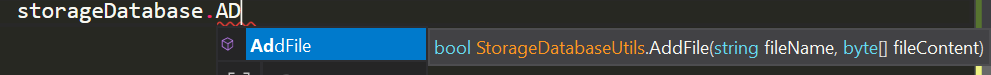


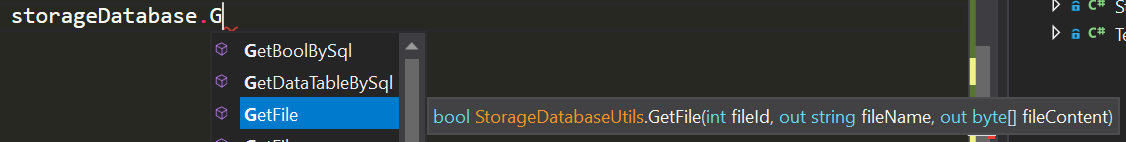
Для початку Нам треба обрати техніку для тестування. Найбільш підходящою, на Мій погляд, є Великий Вибух (“Big Bang” Integration) – всі, або практично всі, розроблені модулі збираються разом у вигляді закінченої системи або її основної частини, а після цього проводиться інтеграційне тестування.

Коли Ми підключили бази даних, можна глянути на методи \*Назва Бази Даних\*Utils, які пропонує Нам Visual Studio. Почати вирішив з бібліотеки FileWorker та БД IIG.CoSWE.StorageDB.

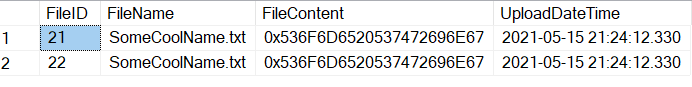


Оскільки Ми тестуємо лише ВЗАЄМОДІЮ, то Мій вибір пав на функції додавання файлів та отримання їх вмісту, тобто AddFile та GetFile



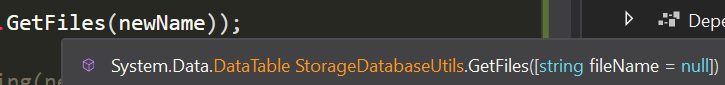


Тут Я почав експериментувати з першим методом, внаслідок чого в БД з’явилося багато файлів кожен з яких мав параметр, який Мною не задавався – FileID. І спочатку Я думав, що це доволі зручно, адже можна діставати вміст файлів за допомогою цього ідентифікатору. При подальших тестах, Я вирішив видалити всі дані з Бази Даних, щоб можна було звертатися до першого елементу і таким чином писати тести. На жаль, тут Я помітив одну проблему – FileID не прийняв значення за замовчанням. Навпаки – він ріс з кожним доданим файлом. У певний момент Моя БД мала такий вигляд:



Єдині два елементи мають значення не, наприклад, (0 та 1) або ж (1 та 2), а (21 й 22). Отже тепер, при кожному запуску тестів, Мені треба буде міняти значення field у тестах…

«Ну добре, але ж ще є метод GetFiles, який приймає лише назву».



Але, на відміну від GetFile, даний метод повертає не масив байтів, а DataTable.

На лекції Я дізнався, що можна дістати з БД максимальне значення параметру FileID, завдяки чому написання тестів продовжилося:



**Тестування**:

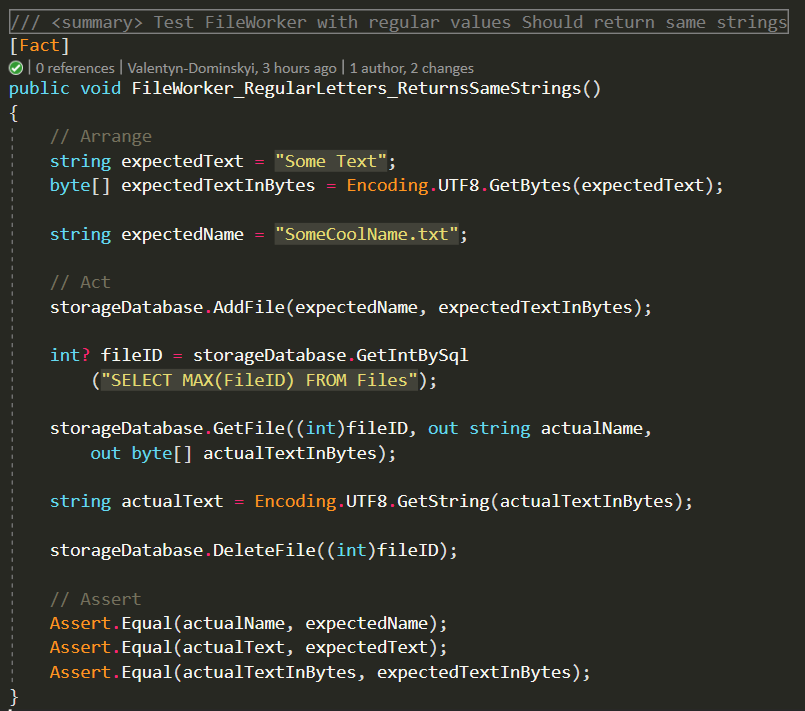
FileWorker

Тести, які стосуються FileWorker мають таку структуру:

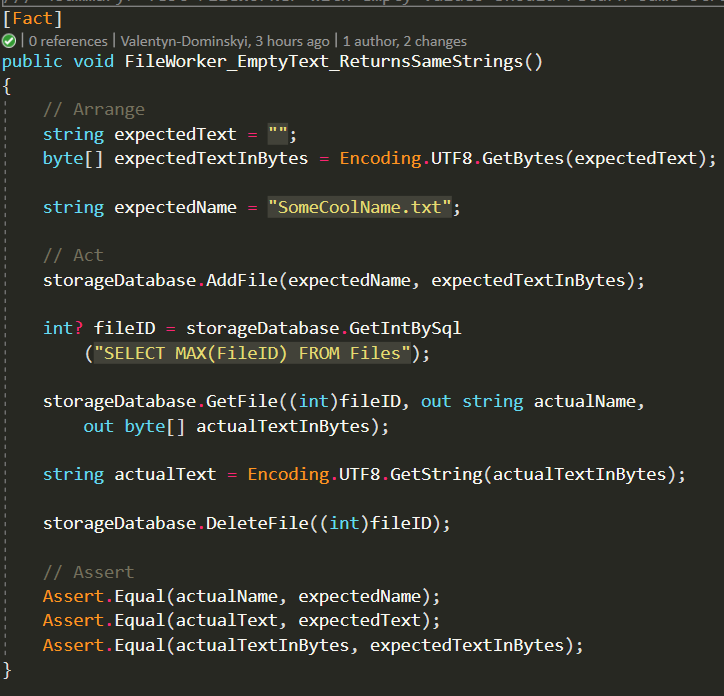
* Задати значення, які будуть використовуватися.
* Додати файл до БД
* Отримати його FileID
* Взяти значення файлу з БД
* Перевірити їх ідентичність

Були перевірені такі кейси:

* Звичайні/повсякденні стрічки, які Ми зазвичай і зустрічаємо в якості назви файлу та його вмісту – значення повертаються успішно



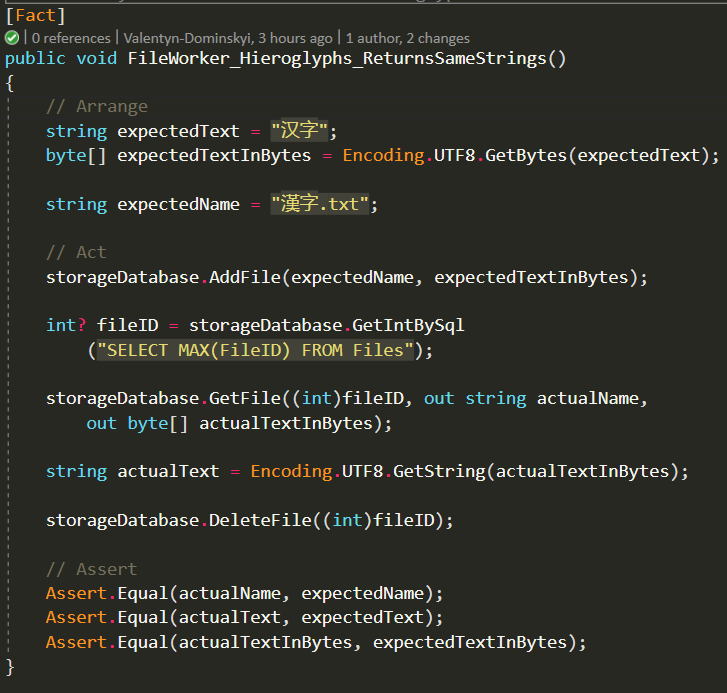
* Порожній вміст файлу - значення повертаються успішно



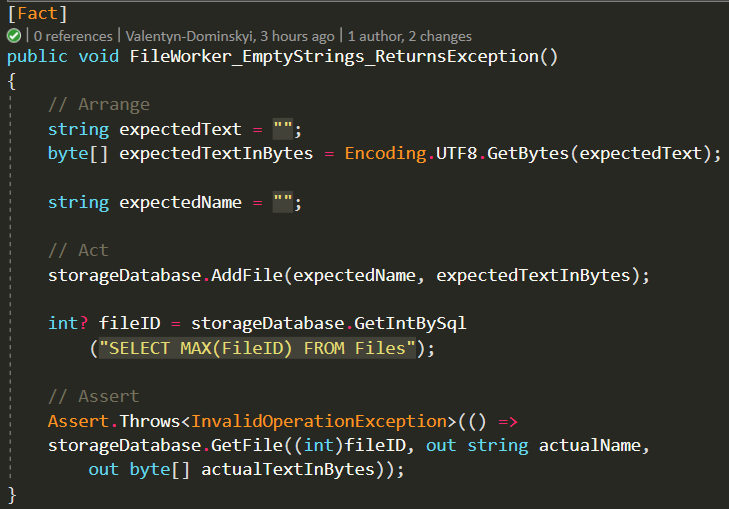
* Емодзі як назва/вміст - значення повертаються успішно



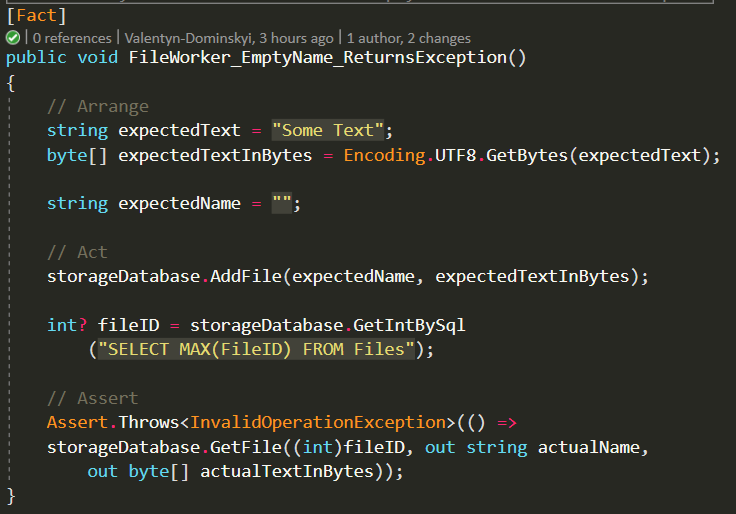
* Ієрогліфи як назва/вміст - значення повертаються успішно



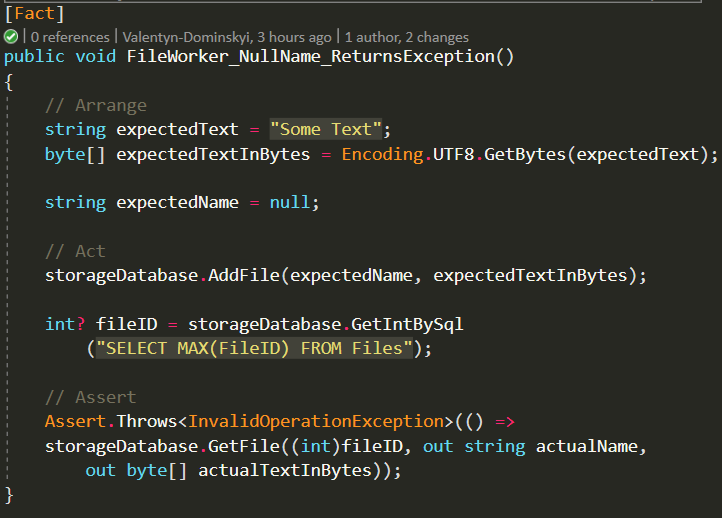
* Усі порожні стрічки - значення не повертаються успішно



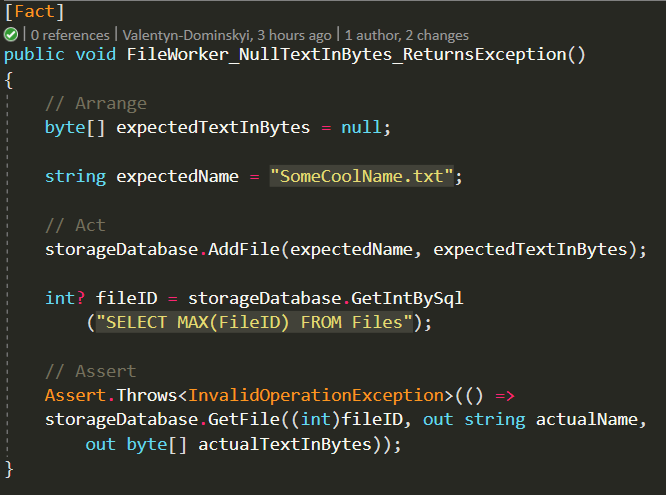
* Порожня назва - значення не повертаються успішно



* Null назва - значення не повертаються успішно



* Null масив байтів - значення не повертаються успішно



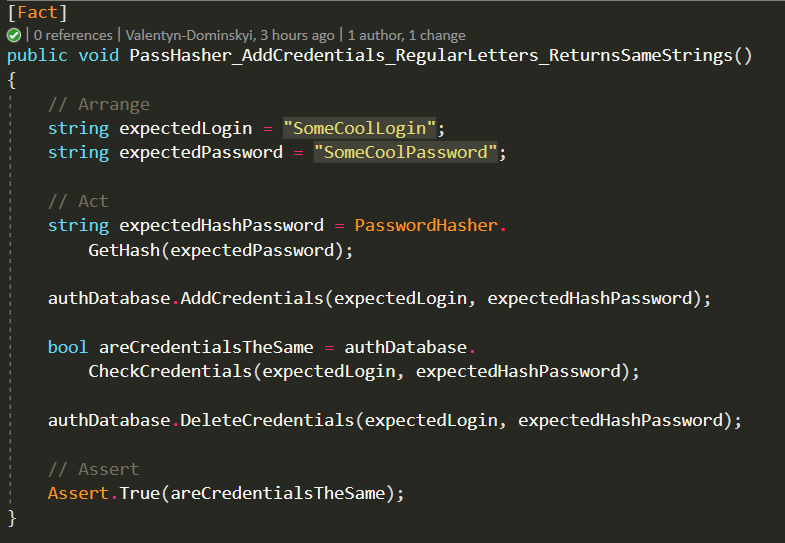
PasswordHasher - AddCredentials

Тести, які стосуються PasswordHasher, а точніше AddCredentials мають таку структуру:

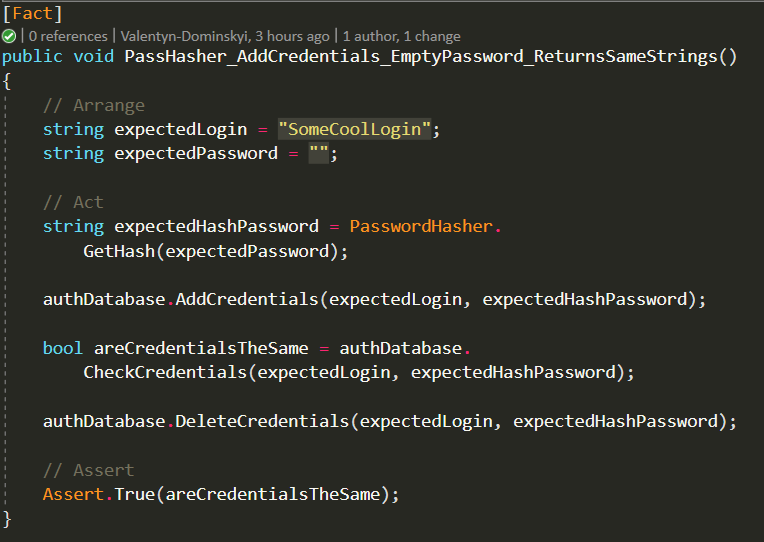
* Задати значення, які будуть використовуватися.
* Отримати хеш
* Додати файл до БД
* Перевірити їх ідентичність за допомогою CheckCredentials

Були перевірені такі кейси:

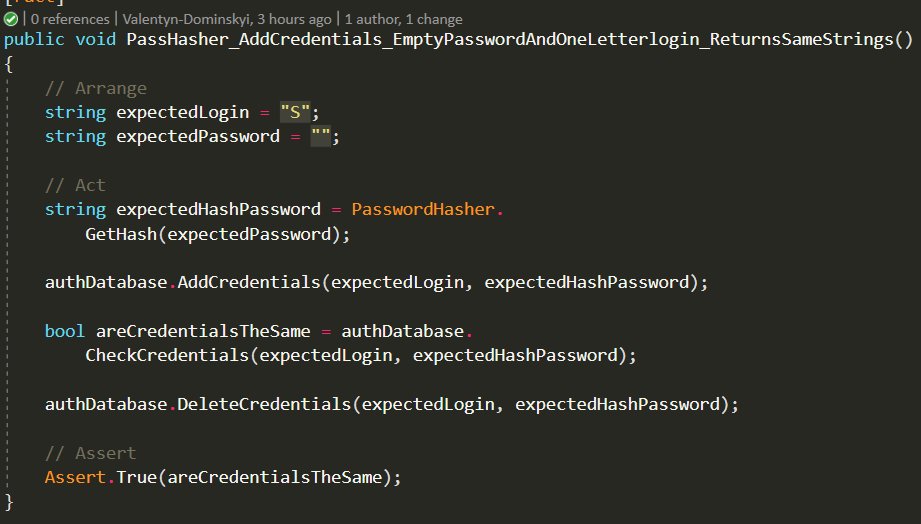
* Звичайні/повсякденні стрічки, які Ми зазвичай і зустрічаємо в якості логіну та паролю– значення збігаються



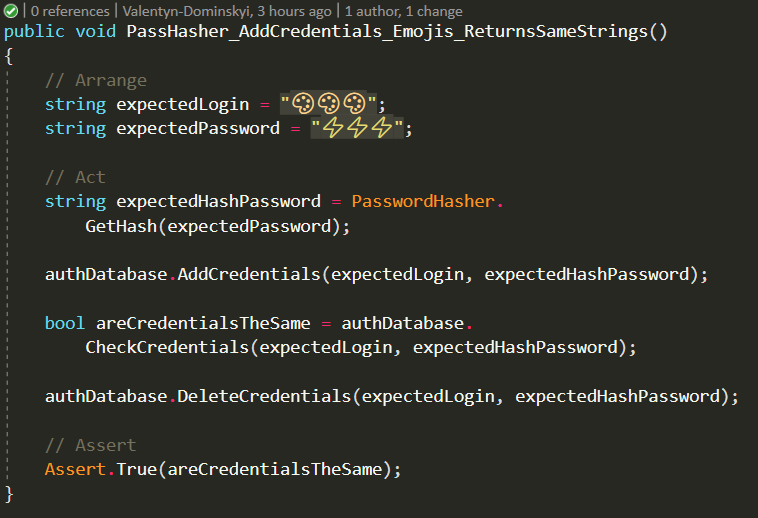
* Порожній пароль - значення збігаються



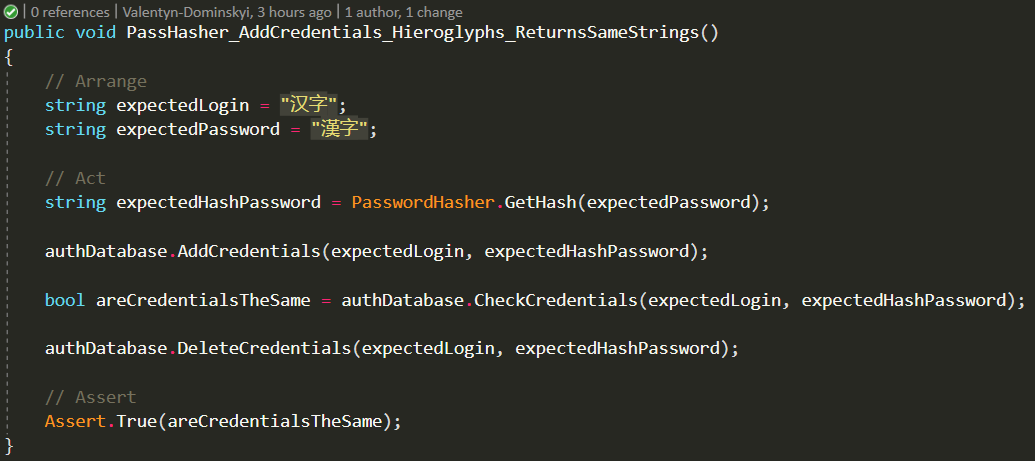
* Порожній пароль та одна літера в логіні - значення збігаються



* Емодзі як логін та пароль - значення збігаються



* Ієрогліфи як логін та пароль значення збігаються



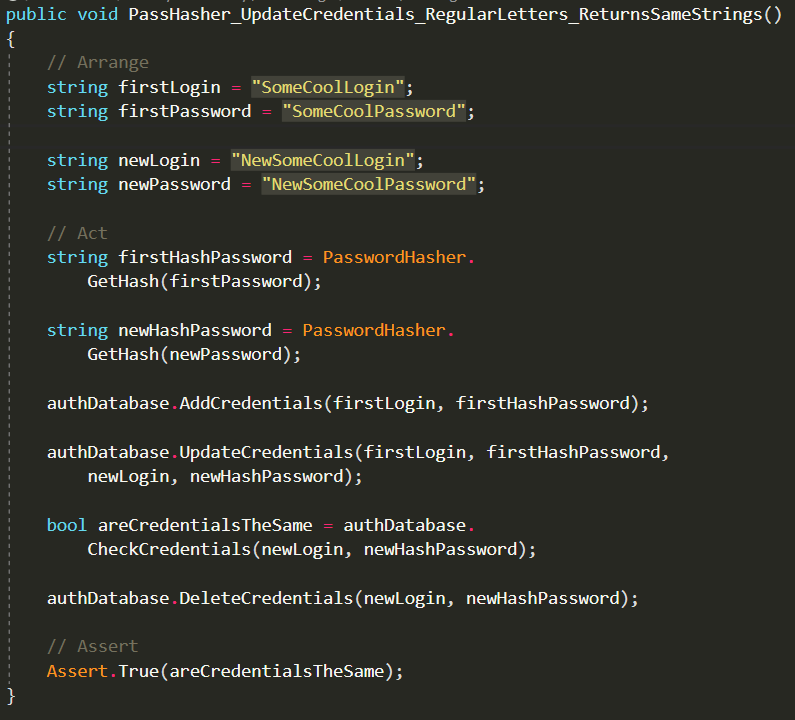
PasswordHasher - UpdateCredentials

Тести, які стосуються PasswordHasher, а точніше UpdateCredentials мають таку структуру:

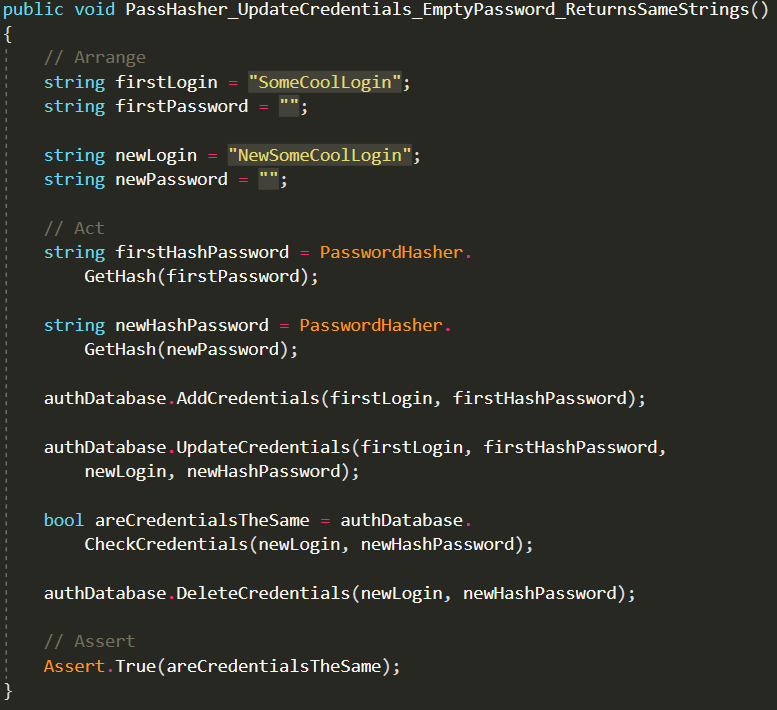
* Задати значення, які будуть використовуватися.
* Отримати хеш
* Додати файл до БД
* Оновити файл у БД
* Перевірити їх ідентичність за допомогою CheckCredentials

Були перевірені такі кейси:

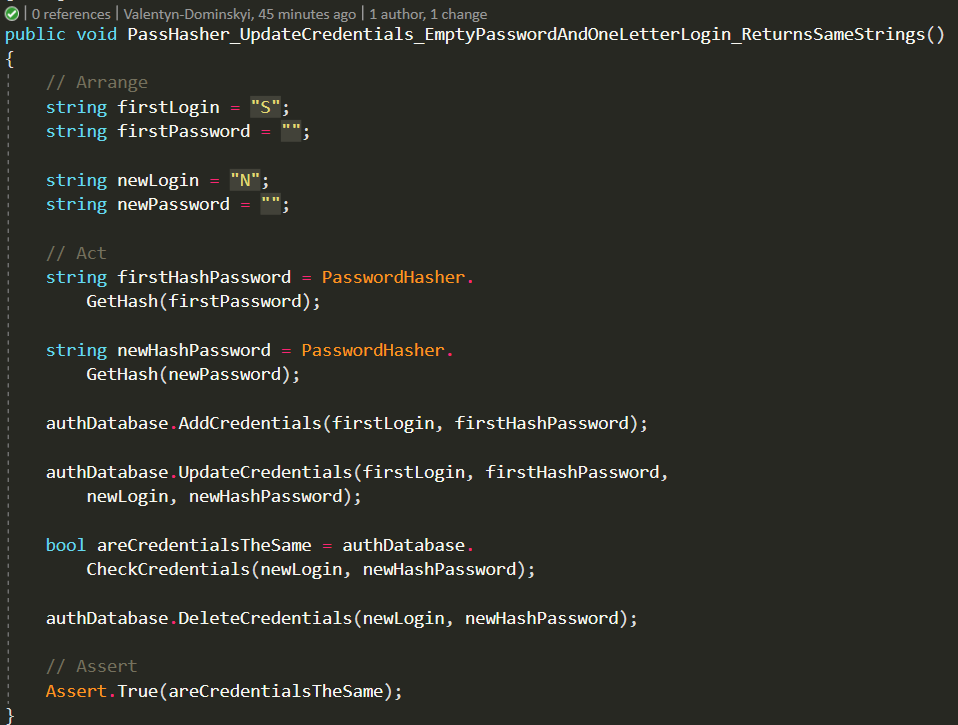
* Звичайні/повсякденні стрічки, які Ми зазвичай і зустрічаємо в якості логіну та паролю– значення збігаються



* Порожній пароль - значення збігаються



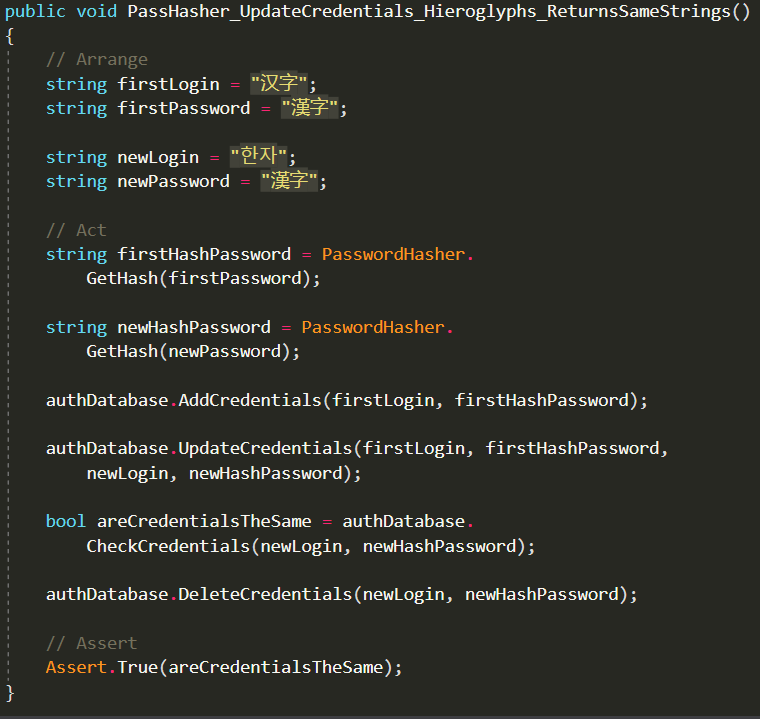
* Порожній пароль та одна літера в логіні - значення збігаються



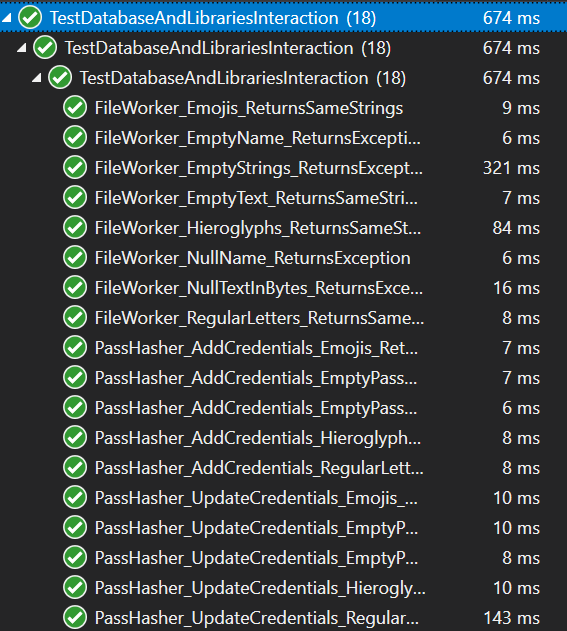
* Емодзі як логін та пароль - значення збігаються



* Ієрогліфи як логін та пароль - значення збігаються



Результати тестування

****

****

Сирцеві коди:

TestDatabaseAndLibrariesInteraction

Висновки:

Виконавши цю лабораторну роботу я познайомився з **White Box тестуванням** у цілому, а також використав таку техніку, як **Тестування потоку виконання** - при тестуванні даним видом тест-кейси створюються таким чином, щоб перевірити правильність виконання *максимально можливої кількість шляхів* виконання ПЗ

Джерела:

* Github - <https://github.com/VsIG-official/Components-Of-Software-Engineering>
* TestPasswordHashingUtils - <https://github.com/VsIG-official/Components-Of-Software-Engineering/blob/master/Labs/Lab3/TestPasswordHashingUtils/TestPasswordHashingUtils.cs>
* Директорія 3-ої лабораторної роботи - <https://github.com/VsIG-official/Components-Of-Software-Engineering/tree/master/Labs/Lab3>
* Офіційна документація - <https://docs.microsoft.com/en-us/dotnet/core/testing/unit-testing-best-practices>
* Лекція по темі White Box Testing - <https://docs.google.com/presentation/d/1UEYTs6OrsO_ssROZNnDG72K5Bc30qL-A/edit#slide=id.p1>