**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 №3**

з дисципліни

«Компоненти програмної інженерії. Якість та тестування програмного забезпечення»

на тему

«Unit тестування з використанням методів White Box Testing»

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](#_Toc70859593)

[Завдання: 3](#_Toc70859594)

[Хід роботи: 3](#_Toc70859595)

[Початок роботи: 3](#_Toc70859596)

[Приватні поля 4](#_Toc70859597)

[Init 4](#_Toc70859598)

[GetHash 6](#_Toc70859599)

[HashSha2 8](#_Toc70859600)

[Adler32CheckSum 9](#_Toc70859601)

[Тестування: 11](#_Toc70859602)

[Init 11](#_Toc70859603)

[Результати тестування 11](#_Toc70859604)

[Сирцеві коди: 12](#_Toc70859605)

[TestFileWorkingUtils (тести) 12](#_Toc70859606)

[Висновки: 12](#_Toc70859607)

[Джерела: 12](#_Toc70859608)

Мета:

Написати Unit тести з використанням методів White Box Testing

Завдання:

|  |  |  |
| --- | --- | --- |
| N п/п | 9311 mod 6 | Library |
| 1 | 0, 2, 3 | IIG.BinaryFlag |
| 2 | 1, 4, 5 | IIG.PasswordHashingUtils |

Варіант = 9311 mod 6 = 5, отже провести тестування IIG.PasswordHashingUtils

Отже, мій вибір для лабораторної:

* .NET 5
* Бібліотека для тестування xUnit
* Бібліотека IIG.PasswordHashingUtils

Хід роботи:

Початок роботи:

Я створив проект “Lab3” на .NET 5, додав xUnit Test Project “TestPasswordHashingUtils” та бібліотеку IIG.PasswordHashingUtils:



Оскільки у даній лабораторній Ми використовуємо BBT, то можемо переглянути її вміст та зробити відповідні графи для подальшого використовування їх при написанні тестів.

Давайте пройдемося по коду та спробуємо його проаналізувати:

Приватні поля



Як видно з даного шматку, Ми маємо два приватних статичних поля з певними значеннями за замовченням:

* \_modAdler представляє з себе модуль найбільшого простого числа, меншого, ніж 216 , який буде використовуватися при хешуванні паролю
* \_salt використовується для створення певного хешу Нашого паролю

Init



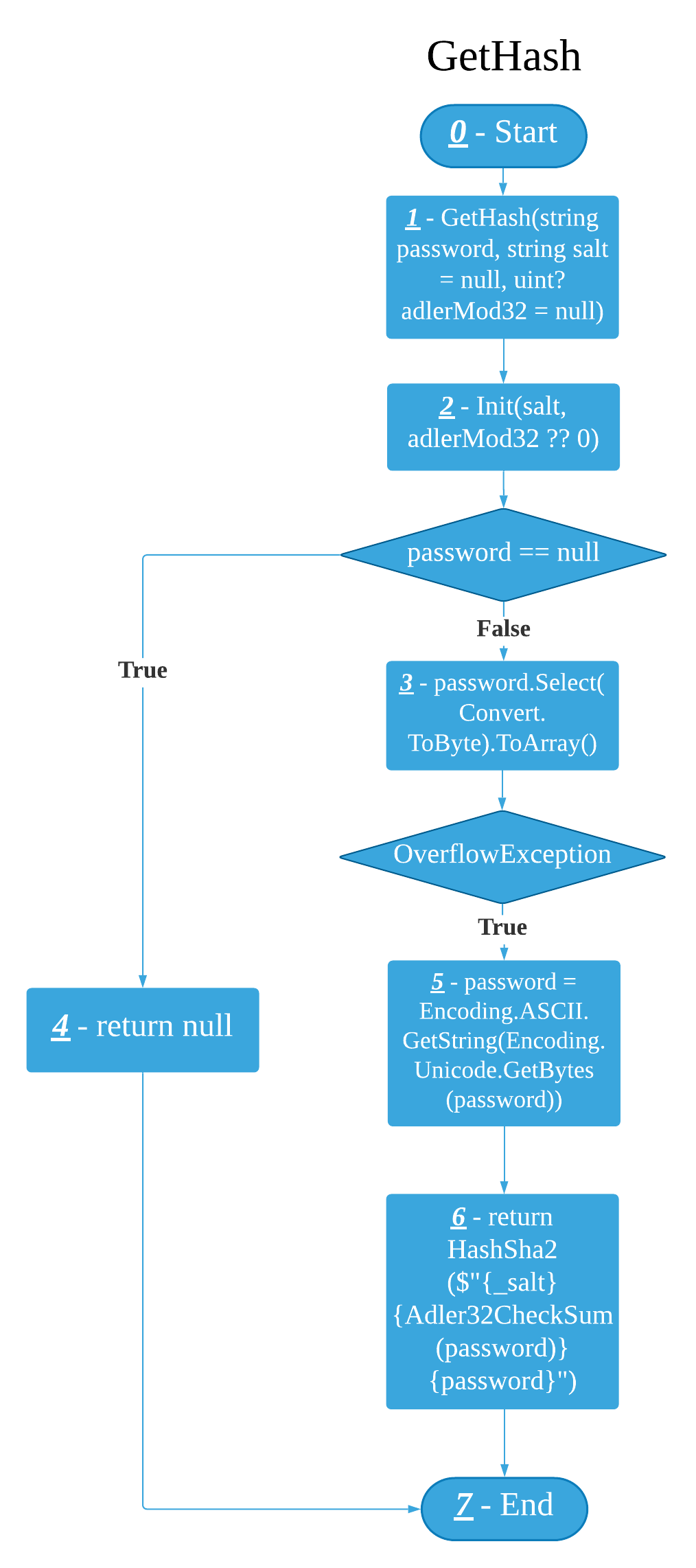
Дана функція перезаписує значення приватних полів, які були вище, з певними перевірками на нові, які приходять як параметри.



GetHash



Цей метод повертає певний хеш для паролю, який Ми передали. Сюди ж можна вписати salt та adlerMod, якщо Нам потрібні якісь конкретні значення.



Далі вже йдуть функції, які є приватними, отже тестувати їх напряму – неможливо

HashSha2



У цьому шматку коду приймається певна стрічка, яка приймає участь у створенні SHA2 хешу.



Adler32CheckSum



І останній метод Adler32CheckSum. Як видно з XML-коментаря дана функція приймає текст, індекс та довжину Adler32CheckSum, а повертає Adler32CheckSum у вигляді стрічки.



Отже Ми можемо дійти до висновку, що напряму тестувати можна лише дві функції: Init та GetHash.

**Тестування**:

Init

Execution Route 0\_1\_6

Почати тестування Я вирішив з методу Init. Але сталася проблема: Я не можу перевірити чи змінилися приватні поля без іншого методу – GetHash. Тобто, якщо повертається один й той же хеш до та після виклику Init, то Ми можемо дійти до висновку, що \_modAdler та \_salt залишилися тими самими. Давайте спробуємо так і зробити:





І тест дійсно пройшов! Тепер зробимо те саме, але з іншими значенням, які підуть по цьому ж маршруту:



Також були спроби передати від’ємне значення AdlerMod, але оскільки у бібліотеці використовується uint, який не може приймати негативні значення, то Visual Studio видавало помилку.

Execution Route 0\_1\_6

А тут Я зрозумів одну річ… усі тести будуть використовувати спільні параметри PasswordHasher’у. Тобто, якщо у тесті, який запускається раніше, йде зміна, наприклад, \_modAdler32, то вона буде помітна в УСІХ подальших тестах, навіть якщо вони знаходяться у різних класах. Через це Я доволі довгий час не міг зрозуміти, чому падає тест. Вирішилось це за допомогою додаткової функції SetDefaultValues, яка викликається на початку кожного тесту.



Результати тестування



Сирцеві коди:

TestFileWorkingUtils (тести)

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

Висновки:

Виконавши цю лабораторну роботу я познайомився з White Box тестуванням у цілому, а також використав таку техніку, як Тестування потоку виконання

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)