**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 І. В.

Завальнюк Максим Євгенович

Залікова книжка: 9312

Номер у списку: 10

Київ 2021

## Мета

Написати Unit тести з використанням методів Black Box Testing.

## Основні кроки виконання

1. Вибрати бібліотеку для тестування за допомогою остачі від ділення номеру залікової книжки на 3.
2. Створити проект тестування, в який підключити бібліотеку.
3. Написати юніт-тести притримуючись методів Black Box Testing.
4. Підготувати звіт про виконану роботу, який буде містити використані методи Black Box Testing, сирцеві коди юніт-тестів та/або посилання на GitHub де розміщено проект і результати тестування.

## Виконання роботи

Для початку я обрахував свій варіант – 9312 mod 3 = 0. Отже, бібліотека, яку необхідно мені тестувати – PasswordHasher.

Буду я працювати із платформою **.NET 5** та бібліотекою для тестування **xUnit**.

Далі я створив проект Lab2(основний) та TestLabsXUnit і Attempts(«пісочниці»). До усіх проектів я під’єднав файл **.dll** бібліотеки, що тестую.

Оскільки це BlackBox тестування, то я нічого спочатку не знав про методи та функції, які є у бібліотеці. За допомогою .xml файлу я зрозумів, які методи взагалі є в бібліотеці. Пізніше програмно, побачивши перелік публічних методів, зрозумів, що буду тестувати PasswordHasher.Init() та PasswordHasher.GetHash(). Звідси я міг визначити скільки аргументів вони приймають, яких типів та чи обов’язкові.

Я створив два класи, щоб тестувати дві функції. Почнемо із PasswordHasher.Init()

public class TestHashingUtils\_Initialization

    {

        /// <summary>

        /// Test init default proccess with parameters

        /// </summary>

        [Fact]

        public void FullParams\_Initialization\_NotNull()

        {

            PasswordHasher.Init("hanzo", 14);

            const string password = PasswordHasher.GetHash("genji");

            Assert.NotNull(password);

        }

        /// <summary>

        /// Test init default proccess with cyrillic parameters

        /// </summary>

        [Fact]

        public void CyrillicParams\_Initialization\_NotNull()

        {

            PasswordHasher.Init("Хандзо", 14);

            const string password = PasswordHasher.GetHash("genji");

            Assert.NotNull(password);

        }

        /// <summary>

        /// Test init default proccess with special parameters

        /// </summary>

        [Fact]

        public void SpecialParams\_Initialization\_NotNull()

        {

            PasswordHasher.Init("\n\r", 14);

            const string password = PasswordHasher.GetHash("genji");

            Assert.NotNull(password);

        }

        /// <summary>

        /// Test init default proccess with emojies in parameters

        /// </summary>

        [Fact]

        public void EmojieParams\_Initialization\_NotNull()

        {

            PasswordHasher.Init("😀😀😀😀", 14);

            const string password = PasswordHasher.GetHash("genji");

            Assert.NotNull(password);

        }

        /// <summary>

        /// Test init default proccess with hieroglyphies in parameters

        /// </summary>

        [Fact]

        public void HieroglyphiesParams\_Initialization\_NotNull()

        {

            PasswordHasher.Init("汉字漢字", 14);

            const string password = PasswordHasher.GetHash("genji");

            Assert.NotNull(password);

        }

        /// <summary>

        /// Test init default proccess with random adler in parameters

        /// </summary>

        [Fact]

        public void RandomAlert\_Initialization\_NotNull()

        {

            Random adler = new Random();

            PasswordHasher.Init("hanzo", (uint)adler.Next());

            const string password = PasswordHasher.GetHash("genji");

            Assert.NotNull(password);

        }

        /// <summary>

        /// Test init default proccess with blank parameters

        /// </summary>

        [Fact]

        public void BlankParams\_Initialization\_NotNull()

        {

            PasswordHasher.Init("", 0);

            const string password = PasswordHasher.GetHash("genji");

            Assert.NotNull(password);

        }

        /// <summary>

        /// Test init proccess in GetHash method

        /// </summary>

        [Fact]

        public void DirectParams\_Initialization\_NotNull()

        {

            const string password = PasswordHasher.GetHash("genji", "hanzo", 14);

            Assert.NotNull(password);

        }

        /// <summary>

        /// Test init proccess in method and outside

        /// </summary>

        [Fact]

        public void CompareDifference\_Initialization\_NotNull()

        {

            const string first\_password = PasswordHasher.GetHash("genji");

            Assert.NotNull(first\_password);

            PasswordHasher.Init("widowmaker", 14);

            const string second\_password = PasswordHasher.GetHash("genji");

            Assert.NotNull(second\_password);

            Assert.NotEqual(first\_password, second\_password);

        }

    }

Я спробував аргументи різного типу передати у цю функції та затестувати їх поведінку. Сама «ініціалізація» впливає на отримання хешу, тому й перевіряв чи зашифрований пароль не є пустотою.

Наступний клас – перевірка функції отримання хешу

public class TestHashingUtils\_Hashing

    {

        /// <summary>

        /// Test default hashing proccess without parameters

        /// </summary>

        [Fact]

        public void WithoutParams\_Hashing\_NotNull()

        {

            const string password = PasswordHasher.GetHash("genji");

            Assert.NotNull(password);

        }

        /// <summary>

        /// Test default hashing proccess with parameters

        /// </summary>

        [Fact]

        public void WithParams\_Hashing\_NotNull()

        {

            const string password = PasswordHasher.GetHash("genji", "hanzo", 14);

            Assert.NotNull(password);

        }

        /// <summary>

        /// Test default hashing proccess with blank parameters

        /// </summary>

        [Fact]

        public void BlankParams\_Hashing\_NotNull()

        {

            const string password = PasswordHasher.GetHash("", "", 14);

            Assert.NotNull(password);

        }

        /// <summary>

        /// Test default hashing proccess with cyrilic parameters

        /// </summary>

        [Fact]

        public void CyrillicParams\_Hashing\_NotNull()

        {

            const string password = PasswordHasher.GetHash("Мей", "Ангел", 14);

            Assert.NotNull(password);

        }

        /// <summary>

        /// Test default hashing proccess with hieroglyphies parameters

        /// </summary>

        [Fact]

        public void HieroglyphiesParams\_Hashing\_NotNull()

        {

            const string password = PasswordHasher.GetHash("汉字漢字", "字漢字字字字字字", 14);

            Assert.NotNull(password);

        }

        /// <summary>

        /// Test default hashing proccess with special parameters

        /// </summary>

        [Fact]

        public void SpecialParams\_Hashing\_NotNull()

        {

            const string password = PasswordHasher.GetHash("\n/n", "\r/r", 14);

            Assert.NotNull(password);

        }

        /// <summary>

        /// Test default hashing proccess with emojies parameters

        /// </summary>

        [Fact]

        public void EmojieParams\_Hashing\_NotNull()

        {

            const string password = PasswordHasher.GetHash("😄 😁 😆 😅", "😄 😁 😆 😅", 14);

            Assert.NotNull(password);

        }

        /// <summary>

        /// Test default hashing proccess with random adler

        /// </summary>

        [Fact]

        public void RandomAdler\_Hashing\_NotNull()

        {

            Random adler = new Random();

            const string password = PasswordHasher.GetHash("genji", "hanzo", (uint)adler.Next());

            Assert.NotNull(password);

        }

        /// <summary>

        /// Test hashing proccess with null parameters in init

        /// </summary>

        [Fact]

        public void NullParams\_Hashing\_NotNull()

        {

            const string password = PasswordHasher.GetHash("genji", null, null);

            Assert.NotNull(password);

        }

        /// <summary>

        /// Test hashing proccess with previous init and blank params in next hashing

        /// </summary>

        [Fact]

        public void PrevoiusInitZeroParams\_Hashing\_NotNull()

        {

            const string password\_with\_params = PasswordHasher.GetHash("genji", "hanzo", 13);

            Assert.NotNull(password\_with\_params);

            const string password\_without\_params = PasswordHasher.GetHash("genji");

            Assert.NotNull(password\_without\_params);

            Assert.Equal(password\_with\_params, password\_without\_params);

        }

        /// <summary>

        /// Test hashing proccess with same parameter after another init

        /// </summary>

        [Fact]

        public void PrevoiusInitOneParam\_Hashing\_NotNull()

        {

            const string password\_with\_params = PasswordHasher.GetHash("genji", "hanzo", 13);

            const string password\_without\_params = PasswordHasher.GetHash("genji");

            const string password\_with\_one\_param = PasswordHasher.GetHash("genji", "hanzo");

            Assert.Equal(password\_with\_params, password\_with\_one\_param);

            Assert.Equal(password\_without\_params, password\_with\_one\_param);

        }

        /// <summary>

        /// Test hashing proccess with different parameter after another init

        /// </summary>

        [Fact]

        public void PrevoiusInitAnotherParam\_Hashing\_NotNull()

        {

            const string password\_without\_params = PasswordHasher.GetHash("genji");

            const string password\_with\_one\_param = PasswordHasher.GetHash("genji", "tracer");

            const string password\_without\_params\_1 = PasswordHasher.GetHash("genji");

            Assert.Equal(password\_without\_params\_1, password\_with\_one\_param);

            Assert.NotEqual(password\_without\_params\_1, password\_without\_params);

        }

    }

Так само як і в попередньому випадку, я пробував різного типу аргументи. З такого тестування я зрозумів, що якщо у функцію передаються другий та третій аргументи, то виконується ініціалізація. Хоча ці аргументи не є обов’язковими.

Я спробував автоматизувати тестування, тому використав GitHub Actions, де ви можете і побачити результати виконання тестів. Посилання в джерелах.

## Висновок

Під час виконання роботи я познайомився із технологією BlackBox тестування. Хоч це було для мене вперше і незвично, було цікаво. Можна сказати, що це було дослідження бібліотеки всліпу. Я спробував написати тести, які спадали на думку та перевірити усю працездатність бібліотеки. Також я познайомився з техніками написання тестів.

## Джерела

1. [Папка](https://github.com/mezgoodle/TestLabs-kpi/tree/master/Lab2) з програмою.
2. [Результати](https://github.com/mezgoodle/TestLabs-kpi/actions/workflows/dotnet.yml) CI.