

**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:

студент групи ПІ-93

Завальнюк Максим Євгенович

Залікова книжка: 9312

Номер у списку: 10

Перевірив:

Бабарикін І. В.

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 –  $9312 \bmod 3 = 0$ . 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 emojis 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("汉字汉字", 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 process in GetHash method
    /// </summary>
    [Fact]
    public void DirectParams_Initialization_NotNull()
    {
        const string password = PasswordHasher.GetHash("genji", "hanzo", 14);
        Assert.NotNull(password);
    }

    /// <summary>
    /// Test init process 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 process without parameters
    /// </summary>
    [Fact]
    public void WithoutParams_Hashing_NotNull()
    {
        const string password = PasswordHasher.GetHash("genji");
        Assert.NotNull(password);
    }

    /// <summary>
    /// Test default hashing process with parameters
    /// </summary>

```

```

[Fact]
public void WithParams_Hashing_NotNull()
{
    const string password = PasswordHasher.GetHash("genji", "hanzo", 14);
    Assert.NotNull(password);
}

/// <summary>
/// Test default hashing process with blank parameters
/// </summary>
[Fact]
public void BlankParams_Hashing_NotNull()
{
    const string password = PasswordHasher.GetHash("", "", 14);
    Assert.NotNull(password);
}

/// <summary>
/// Test default hashing process with cyrillic parameters
/// </summary>
[Fact]
public void CyrillicParams_Hashing_NotNull()
{
    const string password = PasswordHasher.GetHash("Мей", "Ангел", 14);
    Assert.NotNull(password);
}

/// <summary>
/// Test default hashing process with hieroglyphies parameters
/// </summary>
[Fact]
public void HieroglyphiesParams_Hashing_NotNull()
{
    const string password = PasswordHasher.GetHash("汉字汉字", "字汉字字字字字字", 14);
    Assert.NotNull(password);
}

/// <summary>
/// Test default hashing process 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 process with emojis 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", (uin
t)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 has
    /// </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("genj
i");
    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](#) з програмою.
2. [Результати](#) СІ.