Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №2

Сучасні технології Web-застосувань на платформі Microsoft.NET

Модульне тестування. Ознайомлення з засобами та практиками модульного тестування

Варіант 9

Виконала:

студентка групи ІА-11
Шурек Ю.К.

Перевірив:

Бардін В.

Київ 2023

Тема роботи: Модульне тестування. Ознайомлення з засобами та практиками модульного тестування

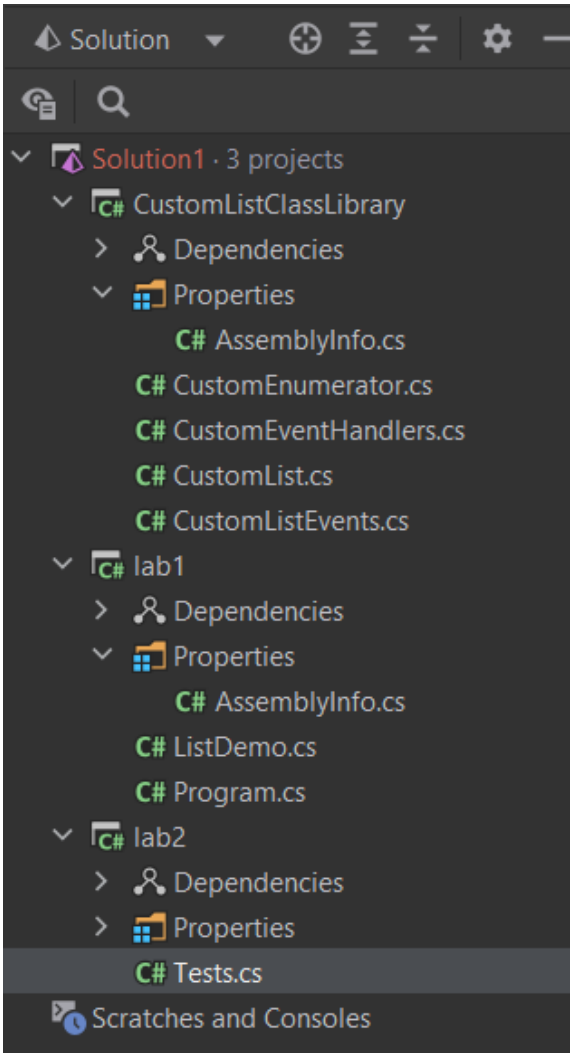
Мета роботи: навчитися створювати модульні тести для вихідного коду розроблювального програмного забезпечення.

Завдання:

1. Додати до проекту власної узагальненої колекції (лаб.1) проект модульних тестів, використовуючи певний фреймворк (Nunit, Xunit, тощо).
2. Розробити модульні тести для функціоналу колекції.
3. Дослідити ступінь покриття модульними тестами вихідного коду колекції, використовуючи, наприклад, засіб AxoCover.

Хід роботи:

Структура проекту:



Tests.cs

```
using System;
using System.Collections.Generic;
using CustomListClassLibrary;
using Xunit;

namespace lab2
{
   public class Tests
   {
        #region Constructor

        [Fact]
        public void Constructor_WithPositiveCapacity_InitializesCustomList()
        {
            // Arrange
            int capacity = 5;

            // Act
            CustomList<int> customList = new CustomList<int>(capacity);

            // Assert
            Assert.Equal(capacity, customList._capacity);
            Assert.Equal(0, customList.Count);
        }

        [Fact]
        public void Constructor_WithZeroCapacity_InitializesCustomList()
        {
            // Arrange
            int capacity = 0;

            // Act
            CustomList<int> customList = new CustomList<int>(capacity);

            // Assert
            Assert.Equal(capacity, customList._capacity);
            Assert.Equal(0, customList.Count);
        }

        [Fact]
        public void Constructor_WithNegativeCapacity_ThrowsException()
        {
            // Arrange
            int capacity = -1;

            // Act & Assert
            Assert.Throws<ArgumentOutOfRangeException>(() => new
CustomList<int>(capacity));
        }

        #endregion

        #region IndexOf
```

```
[Fact]
public void IndexOf_ItemInList_ReturnsIndex()
{
    // Arrange
    CustomList<int> customList = new CustomList<int>();
    customList.Add(1);
    customList.Add(2);

    // Act
    int index = customList.IndexOf(2);

    // Assert
    Assert.Equal(1, index);
}

[Fact]
public void IndexOf_ItemNotInList_ReturnsNegativeOne()
{
    // Arrange
    CustomList<int> customList = new CustomList<int>();
    customList.Add(1);
    customList.Add(2);

    // Act
    int index = customList.IndexOf(3);

    // Assert
    Assert.Equal(-1, index);
}

#endregion

#region Add, Insert

[Fact]
public void Add_ItemToEmptyList_IncreasesSizeAndContainsItem()
{
    // Arrange
    CustomList<int> customList = new CustomList<int>();

    // Act
    customList.Add(42);

    // Assert
    Assert.Equal(1, customList.Count);
    Assert.True(customList.Contains(42));
}

[Fact]
public void Add_ItemToFullList_ResizesAndContainsItem()
{
    // Arrange
    CustomList<int> customList = new CustomList<int>(2);
    customList.Add(1);
    customList.Add(2);
```

```
            // Act
            customList.Add(3);

            // Assert
            Assert.Equal(3, customList.Count);
            Assert.True(customList.Contains(3));
        }

        [Fact]
        public void Insert_ValidIndex_InsertsItem()
        {
            // Arrange
            CustomList<int> customList = new CustomList<int>();
            customList.Add(1);
            customList.Add(3);

            // Act
            customList.Insert(1, 2);

            // Assert
            Assert.Equal(3, customList.Count);
            Assert.Equal(2, customList[1]);
        }

        [Fact]
        public void Insert_InvalidIndex_ThrowsArgumentOutOfRangeException()
        {
            // Arrange
            CustomList<int> customList = new CustomList<int>();

            // Act & Assert
            Assert.Throws<ArgumentOutOfRangeException>(() => customList.Insert(-1,
1));
            Assert.Throws<ArgumentOutOfRangeException>(() => customList.Insert(1,
1));
        }

        [Fact]
        public void Insert_WithResize_ResizesAndInsertsItem()
        {
            // Arrange
            CustomList<int> customList = new CustomList<int>(2);
            customList.Add(1);
            customList.Add(2);

            // Act
            customList.Insert(1, 3);

            // Assert
            Assert.Equal(3, customList.Count);
            Assert.Equal(3, customList[1]);
        }

        #endregion

        #region Remove, RemoveAt
```

```
        [Fact]
        public void Remove_ItemInList_ReturnsTrueAndRemovesItem()
        {
            // Arrange
            CustomList<int> customList = new CustomList<int>();
            customList.Add(1);
            customList.Add(2);

            // Act
            bool removed = customList.Remove(1);

            // Assert
            Assert.True(removed);
            Assert.False(customList.Contains(1));
            Assert.Equal(1, customList.Count);
        }

        [Fact]
        public void Remove_ItemDoesNotExistInList_ThrowsInvalidOperationException()
        {
            // Arrange
            CustomList<int> customList = new CustomList<int>();
            customList.Add(1);
            customList.Add(2);

            // Act & Assert
            Assert.Throws<InvalidOperationException>(() => customList.Remove(3));
        }

        [Fact]
        public void RemoveAt_ValidIndex_RemovesItem()
        {
            // Arrange
            CustomList<int> customList = new CustomList<int>();
            customList.Add(1);
            customList.Add(2);
            customList.Add(3);

            // Act
            customList.RemoveAt(1);

            // Assert
            Assert.Equal(2, customList.Count);
            Assert.False(customList.Contains(2));
        }

        [Fact]
        public void RemoveAt_InvalidIndex_ThrowsArgumentOutOfRangeExcepStion()
        {
            // Arrange
            CustomList<int> customList = new CustomList<int>();

            // Act & Assert
            Assert.Throws<ArgumentOutOfRangeException>(() =>
customList.RemoveAt(0));
```

```
        }

        #endregion

        #region CopyTo

        [Fact]
        public void CopyTo_ValidParameters_CopiesItemsToArray()
        {
            // Arrange
            CustomList<int> customList = new CustomList<int>();
            customList.Add(1);
            customList.Add(2);
            customList.Add(3);
            int[] array = new int[5];

            // Act
            customList.CopyTo(array, 1);

            // Assert
            int[] expected = { 0, 1, 2, 3, 0 };
            Assert.Equal(expected, array);
        }

        [Fact]
        public void CopyTo_NullArray_ThrowsArgumentNullException()
        {
            // Arrange
            CustomList<int> customList = new CustomList<int>();
            customList.Add(1);
            int[] array = null;

            // Act & Assert
            Assert.Throws<ArgumentNullException>(() => customList.CopyTo(array,
0));
        }

        [Fact]
        public void CopyTo_InvalidArrayIndex_ThrowsArgumentOutOfRangeException()
        {
            // Arrange
            CustomList<int> customList = new CustomList<int>();
            customList.Add(1);
            int[] array = new int[5];

            // Act & Assert
            Assert.Throws<ArgumentOutOfRangeException>(() =>
customList.CopyTo(array, -1));
            Assert.Throws<ArgumentOutOfRangeException>(() =>
customList.CopyTo(array, 6));
        }

        [Fact]
        public void CopyTo_NotEnoughSpaceInArray_ThrowsArgumentException()
        {
            // Arrange
```

```
        CustomList<int> customList = new CustomList<int>();
        customList.Add(1);
        customList.Add(2);
        int[] array = new int[1];

        // Act & Assert
        Assert.Throws<ArgumentException>(() => customList.CopyTo(array, 0));
    }

    #endregion

    #region Contains

    [Fact]
    public void Contains_ItemInList_ReturnsTrue()
    {
        // Arrange
        CustomList<int> customList = new CustomList<int>();
        customList.Add(42);

        // Act & Assert
        Assert.True(customList.Contains(42));
    }

    [Fact]
    public void Contains_ItemNotInList_ReturnsFalse()
    {
        // Arrange
        CustomList<int> customList = new CustomList<int>();
        customList.Add(42);

        // Act & Assert
        Assert.False(customList.Contains(24));
    }

    #endregion

    #region Clear

    [Fact]
    public void Clear_EmptyList_ClearsList()
    {
        // Arrange
        CustomList<int> customList = new CustomList<int>();

        // Act
        customList.Clear();

        // Assert
        Assert.Equal(0, customList.Count);
        Assert.False(customList.Contains(42));
    }

    [Fact]
    public void Clear_NonEmptyList_ClearsList()
    {
```

```
    // Arrange
    CustomList<int> customList = new CustomList<int>();
    customList.Add(42);

    // Act
    customList.Clear();

    // Assert
    Assert.Equal(0, customList.Count);
    Assert.False(customList.Contains(42));
}

#endregion

#region Set Element

[Fact]
public void Set_ValidIndex_SetsValue()
{
    // Arrange
    CustomList<int> customList = new CustomList<int>();
    customList.Add(1);

    // Act
    customList[0] = 2;

    // Assert
    Assert.Equal(2, customList[0]);
}

[Fact]
public void Set_InvalidIndex_ThrowsArgumentOutOfRangeException()
{
    // Arrange
    CustomList<int> customList = new CustomList<int>();
    customList.Add(1);

    // Act & Assert
    Assert.Throws<ArgumentOutOfRangeException>(() => customList[-1] = 2);
    Assert.Throws<ArgumentOutOfRangeException>(() => customList[1] = 2);
}

#endregion

#region Enumerator Methods

[Fact]
public void GetEnumerator_EmptyList_ReturnsEmptyEnumerator()
{
    // Arrange
    CustomList<int> customList = new CustomList<int>();

    // Act
    var enumerator = customList.GetEnumerator();

    // Assert
```

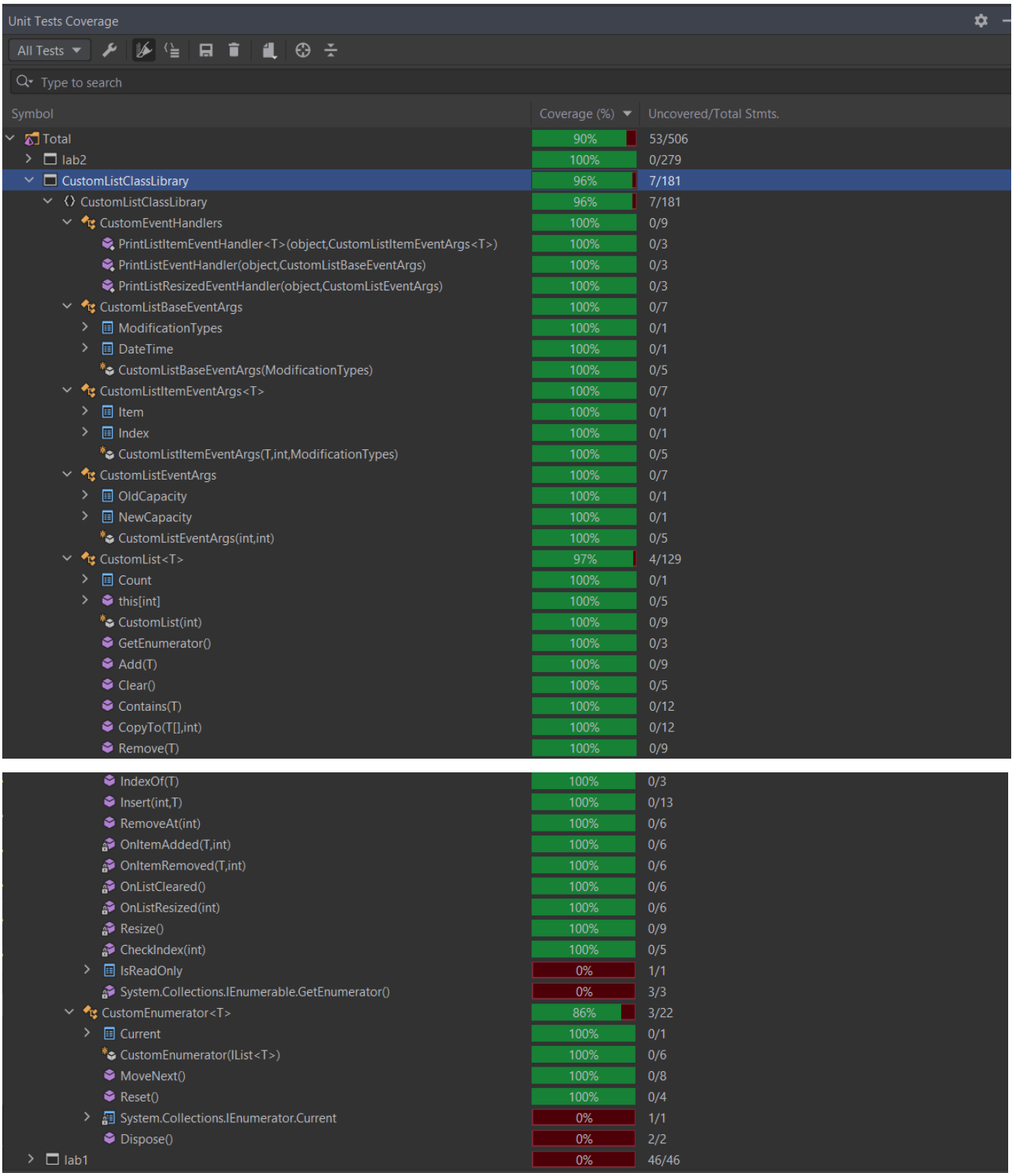
```
            Assert.False(enumerator.MoveNext());
        }

        [Fact]
        public void
GetEnumerator_ListWithItems_ReturnsEnumeratorWithCorrectValues()
        {
            // Arrange
            CustomList<int> customList = new CustomList<int>();
            customList.Add(1);
            customList.Add(2);
            customList.Add(3);

            // Act
            var enumerator = customList.GetEnumerator();

            // Assert
            List<int> items = new List<int>();
            while (enumerator.MoveNext())
            {
                items.Add(enumerator.Current);
            }

            Assert.Equal(new[] { 1, 2, 3 }, items);
        }

        [Fact]
        public void Current_AtStartOfEnumerator_ReturnsDefaultElement()
        {
            // Arrange
            var collection = new CustomList<int>() { 1, 2, 3, 4, 5 };
            var enumerator = collection.GetEnumerator();

            // Act & Assert
            Assert.Equal(default, enumerator.Current);
        }

        [Fact]
        public void Current_AfterResetAndMoveNext_ReturnsFirstElement()
        {
            // Arrange
            var collection = new CustomList<int>() { 1, 2, 3, 4, 5 };
            var enumerator = collection.GetEnumerator();

            enumerator.MoveNext();
            enumerator.MoveNext();

            // Act
            enumerator.Reset();
            enumerator.MoveNext();

            // Assert
            Assert.Equal(1, enumerator.Current);
        }

        #endregion
```

```
#region trigerring events

[Fact]
public void AddItem_TriggerItemAddedEvent()
{
    // Arrange
    var customList = new CustomList<int>();
    var eventRaised = false;
    customList.ItemAdded += (sender, args) => eventRaised = true;

    // Act
    customList.Add(1);

    // Assert
    Assert.True(eventRaised);
}

[Fact]
public void RemoveItem_TriggerItemRemovedEvent()
{
    // Arrange
    var customList = new CustomList<int>();
    customList.Add(1);
    var eventRaised = false;
    customList.ItemRemoved += (sender, args) => eventRaised = true;

    // Act
    customList.Remove(1);

    // Assert
    Assert.True(eventRaised);
}

[Fact]
public void ClearList_TriggerListClearedEvent()
{
    // Arrange
    var customList = new CustomList<int>();
    customList.Add(1);
    var eventRaised = false;
    customList.ListCleared += (sender, args) => eventRaised = true;

    // Act
    customList.Clear();

    // Assert
    Assert.True(eventRaised);
}

[Fact]
public void ResizeList_TriggerListResizedEvent()
{
    // Arrange
    var customList = new CustomList<int>();
    var eventRaised = false;
```

```
            customList.ListResized += (sender, args) => eventRaised = true;

            // Act
            customList.Add(1);
            customList.Add(2);
            customList.Add(3);
            customList.Add(4);
            customList.Add(5);

            // Assert
            Assert.True(eventRaised);
        }

        #endregion

        #region event handlers

        [Fact]
        public void ItemAddedEvent_HandlerLogsCorrectly()
        {
            // Arrange
            var customList = new CustomList<int>();
            customList.ItemAdded += CustomEventHandlers.PrintListItemEventHandler;

            var consoleOutput = new System.IO.StringWriter();
            Console.SetOut(consoleOutput);

            // Act
            customList.Add(1);

            // Assert
            var expectedOutput = "\t\t\t\t\t\t\t\t\tEvent: ItemAdded Item: 1 Index:
0 Date:";
            Assert.Contains(expectedOutput, consoleOutput.ToString());
        }

        [Fact]
        public void ItemRemovedEvent_HandlerLogsCorrectly()
        {
            // Arrange
            var customList = new CustomList<int>();
            customList.ItemRemoved +=
CustomEventHandlers.PrintListItemEventHandler;

            var consoleOutput = new System.IO.StringWriter();
            Console.SetOut(consoleOutput);

            customList.Add(1);

            // Act
            customList.Remove(1);

            // Assert
            var expectedOutput = "\t\t\t\t\t\t\t\t\tEvent: ItemRemoved Item: 1
Index: 0 Date:";
            Assert.Contains(expectedOutput, consoleOutput.ToString());
```

```
        }

        [Fact]
        public void ListClearedEvent_HandlerLogsCorrectly()
        {
            // Arrange
            var customList = new CustomList<int>();
            customList.ListCleared += CustomEventHandlers.PrintListEventHandler;

            var consoleOutput = new System.IO.StringWriter();
            Console.SetOut(consoleOutput);

            customList.Add(1);

            // Act
            customList.Clear();

            // Assert
            var expectedOutput = "\t\t\t\t\t\t\t\t\tEvent: ListCleared Date:";
            Assert.Contains(expectedOutput, consoleOutput.ToString());
        }

        [Fact]
        public void ListResizedEvent_HandlerLogsCorrectly()
        {
            // Arrange
            var customList = new CustomList<int>();
            customList.ListResized +=
CustomEventHandlers.PrintListResizedEventHandler;

            var consoleOutput = new System.IO.StringWriter();
            Console.SetOut(consoleOutput);

            // Act
            customList.Add(1);
            customList.Add(2);
            customList.Add(3);

            // Assert
            var expectedOutput = "\t\t\t\t\t\t\t\t\tEvent: ListResized Old
capacity: 0 New capacity: 4 Date:";
            Assert.Contains(expectedOutput, consoleOutput.ToString());
        }

        #endregion
    }
}
```

Результат роботи:



Висновок: я навчилася створювати модульні тести для вихідного коду розроблювального програмного забезпечення.