Міністерство освіти і науки України КПІ ім. Ігоря Сікорського

Факультет Інформатики та Обчислювальної Техніки

# ЗВІТ

**до лабораторної роботи №2 з модуля**

**«Сучасні технології розробки WEB-застосувань на платформі Microsoft.NET»**

Перевірив:
Викладач кафедри ІСТ
ФІОТ
Бардін В.

Виконав:
Боровков Іван
гр. ІП-11

2023

**Модульне тестування. Ознайомлення з засобами та практиками модульного тестування**

**Мета лабораторної роботи** – навчитися створювати модульні тести для вихідного коду розроблювального програмного забезпечення.

**Завдання:**

1. Додати до проекту власної узагальненої колекції (застосувати виконану лабораторну роботу No1) 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.

Код програми

**Test.cs**

```
using System.Collections;
using MyQueue;

namespace MyQueueTest;

public class Tests
{
    private readonly int[] _valuesToQueue = { 1, 2, 3, 4, 5, 6 };
    private MyQueue<int> _emptyQueue = null!;
    private MyQueue<int> _filledQueue = null!;
    private IEnumerator _enumerator = null!;

    [SetUp]
    public void Setup()
    {
        _emptyQueue = new MyQueue<int>();
        _filledQueue = new MyQueue<int>(_valuesToQueue);
        _enumerator = _filledQueue.GetEnumerator();
    }

    [Test]
    public void Ctor_Empty_ReturnsEmptyQueue()
    {
```

```
    Assert.That(_emptyQueue, Has.Count.EqualTo(0));
}

[Test]
public void Ctor_Enumerable_ReturnsQueueFromEnumerable()
{
    Assert.That(_filledQueue, Has.Count.EqualTo(6));
    Assert.That(_filledQueue.Dequeue(), Is.EqualTo(1));
}

[Test]
public void Enqueue_SingleNumber_ReturnsCorrectQueueSize()
{
    _emptyQueue.Enqueue(1);

    Assert.That(_emptyQueue, Has.Count.EqualTo(1));
}

[Test]
public void Enqueue_SequenceOfNumbers_ReturnsCorrectQueueSize()
{
    foreach (var item in _valuesToQueue)
    {
        _emptyQueue.Enqueue(item);
    }

    Assert.That(_emptyQueue, Has.Count.EqualTo(6));
}

[Test]
public void Enqueue_SingleString_ReturnsCorrectQueueSize()
{
    var queue = new MyQueue<string>();
    queue.Enqueue("1");

    Assert.That(queue, Has.Count.EqualTo(1));
}

[Test]
public void Dequeue_FilledQueue_DequeueFirstItem()
```

```
{
    Assert.That(_filledQueue.Dequeue(), Is.EqualTo(1));
    Assert.That(_filledQueue, Has.Count.EqualTo(5));
}

[Test]
public void Dequeue_FilledQueue_DequeueFiveItem()
{
    for (int i = 0; i < 6; i++)
    {
        Assert.That(_filledQueue.Dequeue(), Is.EqualTo(_valuesToQueue[i]));

    }
    Assert.That(_filledQueue, Has.Count.EqualTo(0));
}

[Test]
public void Dequeue_EmptyQueue_ThrowsInvalidOperationException()
{
    Assert.Throws<InvalidOperationException>(() => _emptyQueue.Dequeue());
}

[Test]
public void Peek_FilledQueue_ReturnsFirstItem()
{
    Assert.That(_filledQueue.Peek(), Is.EqualTo(1));
    Assert.That(_filledQueue, Has.Count.EqualTo(6));
}

[Test]
public void Peek_EmptyQueue_ThrowsInvalidOperationException()
{
    Assert.Throws<InvalidOperationException>(() => _emptyQueue.Peek());
}

[Test]
public void Clear_EmptyQueue_QueueIsEmpty()
{
    _emptyQueue.Clear();
    Assert.That(_emptyQueue, Has.Count.EqualTo(0));
```

```
    Assert.Throws<InvalidOperationException>(() => _emptyQueue.Peek());
}

[Test]
public void Clear_FilledQueue_QueueIsEmpty()
{
    _filledQueue.Clear();
    Assert.That(_emptyQueue, Has.Count.EqualTo(0));
    Assert.Throws<InvalidOperationException>(() => _emptyQueue.Peek());
}

[Test]
[TestCase(1)]
[TestCase(2)]
[TestCase(3)]
public void Contains_EmptyQueue_ReturnsFalse(int value)
{
    Assert.That(_emptyQueue.Contains(value), Is.EqualTo(false));
}

[Test]
[TestCase(1)]
[TestCase(2)]
[TestCase(3)]
public void Contains_FilledQueue_ReturnsTrue(int value)
{
    Assert.That(_filledQueue.Contains(value), Is.EqualTo(true));
}

[Test]
[TestCase(-3)]
[TestCase(10)]
[TestCase(10000000)]
public void Contains_FilledQueue_ReturnsFalse(int value)
{
    Assert.That(_filledQueue.Contains(value), Is.EqualTo(false));
}

[Test]
public void CopyTo_EmptyQueueFromZeroIndex_SequenceIsEqual()
```

```
{
    var basicArray = new[] { 1, 2, 3, 4 };
    var expectedArray = new[] { 1, 2, 3, 4 };
    _emptyQueue.CopyTo(basicArray, 0);
    CollectionAssert.AreEqual(basicArray, expectedArray);
}

[Test]
public void CopyTo_EmptyQueueFromFirstIndex_SequenceIsEqual()
{
    var basicArray = new[] { 1, 2, 3, 4 };
    var expectedArray = new[] { 1, 2, 3, 4 };
    _emptyQueue.CopyTo(basicArray, 1);
    CollectionAssert.AreEqual(basicArray, expectedArray);
}

[Test]
public void CopyTo_FilledQueueFromZeroIndex_SequenceIsEqual()
{
    var arr = new int[6];
    _filledQueue.CopyTo(arr, 0);

    CollectionAssert.AreEqual(arr, _valuesToQueue);
}

[Test]
public void CopyTo_FilledQueueFromFirstIndex_SequenceIsEqual()
{
    var arr = new[] { 1, 2, 3, 4, 5, 6, 7 };
    var expectedArr = new[] { 1, 1, 2, 3, 4, 5, 6 };
    _filledQueue.CopyTo(arr, 1);

    CollectionAssert.AreEqual(arr, expectedArr);
}

[Test]
public void CopyTo_NullSequence_ThrowsArgumentNullException()
{
    Assert.Throws<ArgumentNullException>(() => _filledQueue.CopyTo(null!, 0));
}
```

```
    [Test]
    public void CopyTo_IncorrectIndex_ThrowsArgumentOutOfRangeException()
    {
        var arr = new int[6];

        Assert.Throws<ArgumentOutOfRangeException>(() =>
_filledQueue.CopyTo(arr, -1));
        Assert.Throws<ArgumentOutOfRangeException>(() =>
_filledQueue.CopyTo(arr, 6));
    }

    [Test]
    public void CopyTo_FilledQueueIsGreaterThanArr_ThrowsArgumentException()
    {
        var arr = new int[5];

        Assert.Throws<ArgumentException>(() => _filledQueue.CopyTo(arr, 0));
    }

    [Test]
    public void
CopyTo_FilledQueueIsGreaterThanArrFromIndex_ThrowsArgumentException()
    {
        var arr = new int[6];

        Assert.Throws<ArgumentException>(() => _filledQueue.CopyTo(arr, 1));
    }

    [Test]
    public void ToArray_EmptyQueue_IsEqualToEmptyArr()
    {

        Assert.That(_emptyQueue.ToArray(), Is.EqualTo(Array.Empty<int>()));
    }

    [Test]
    public void ToArray_FilledQueue_IsEqualToBasicSequence()
    {
        Assert.That(_filledQueue.ToArray(), Is.EqualTo(_valuesToQueue));
```

```
}

[Test]
public void GetEnumerator_NonGenericIEnumerableQueue_ReturnFirstValue()
{
   IEnumerable queue = new MyQueue<int>(_valuesToQueue);
   IEnumerator enumerator = queue.GetEnumerator();
   enumerator.MoveNext();
   Assert.That(enumerator.Current, Is.EqualTo(1));
}

private IEnumerable AsWeakEnumerable<T>(IEnumerable<T> sequence)
{
   foreach (object o in sequence)
   {
      yield return o;
   }
}

[Test]
public void Enumerator_FilledQueue_IsEqualToBasicSequence()
{
   IEnumerable weak = AsWeakEnumerable(_filledQueue);

   var sequence = weak.Cast<int>().Take(6).ToArray();
   CollectionAssert.AreEqual(sequence, _valuesToQueue);
}

[Test]
public void LinqCount_EmptyQueue_IsEqualToZero()
{
   var i = _emptyQueue.Count();

   Assert.That(i, Is.EqualTo(0));
}

[Test]
public void Enumerator_MoveNext_CorrectEnumeration_IsFalseAtTheEnd()
{
   for (int i = 0; i < 6; i++)
```

```
        {
            Assert.That(_enumerator.MoveNext(), Is.True);
        }

        Assert.That(_enumerator.MoveNext(), Is.False);
    }

    [Test]
    public void Enumerator_MoveNext_CorrectEnumeration_IsFalseAfterTheEnd()
    {
        for (int i = 0; i < 7; i++)
        {
            _enumerator.MoveNext();
        }

        Assert.That(_enumerator.MoveNext(), Is.False);
    }

    [Test]
    public void Enumerator_MoveNext_WithEmptyQueue_IsFalse()
    {
        _enumerator = _emptyQueue.GetEnumerator();
        Assert.That(_enumerator.MoveNext(), Is.False);
    }

    [Test]
    public void
Enumerator_Current_EnumerationDoesNotStarted_ThrowsInvalidOperationExceptio
n()
    {
        Assert.Throws<InvalidOperationException>(() =>
        {
            var i = _enumerator.Current;
        });
    }

    [Test]
    public void Enumerator_Current_EnumerationCorrect_EveryValueIsEqual()
    {
        int i = 0;
```

```
        while (_enumerator.MoveNext())
        {
            Assert.That(_enumerator.Current, Is.EqualTo(_valuesToQueue[i++]));
        }
    }

    [Test]
    public void
Enumerator_Current_EnumerationFinished_ThrowsInvalidOperationException()
    {
        while (_enumerator.MoveNext())
        {
        }

        Assert.Throws<InvalidOperationException>(() =>
        {
            var i = _enumerator.Current;
        });
    }

    [Test]
    public void Enumerator_Reset_EnumerationFinished_MoveNextIsTrue()
    {
        while (_enumerator.MoveNext())
        {
        }

        _enumerator.Reset();

        Assert.That(_enumerator.MoveNext(), Is.True);
    }

    [Test]
    public void SyncRoot_FilledQueue_IsEqualToFilledQueue()
    {
        Assert.That(_filledQueue.SyncRoot, Is.EqualTo(_filledQueue));
    }

    [Test]
    public void IsSynchronized_FilledQueue_IsAlwaysTrue()
```

```
    {
        Assert.That(_filledQueue.IsSynchronized, Is.False);
    }
}
```

**Висновок:** в ході роботи ми навчилися створювати модульні тести для вихідного коду розроблювального програмного забезпечення.

.