МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

імені ІГОРЯ СІКОРСЬКОГО»

Лабораторна робота №2

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

на тему: «Модульне тестування. Ознайомлення із засобами та практиками модульного тестування.»

Виконав:

студент гр. ІП-11

Гіжицький Д. О.

Викладач:

Бардін В.

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).

1. Розробити модульні тести для функціоналу колекції.
2. Дослідити ступінь покриття модульними тестами вихідного коду

колекції, використовуючи, наприклад, засіб AxoCover.

**Варіант 9:** ****

**Посилання на репозиторій:**

<https://github.com/doomer2004/dotNetLabs2023term1/tree/main/GenericLab1>

**Лістинг коду:**

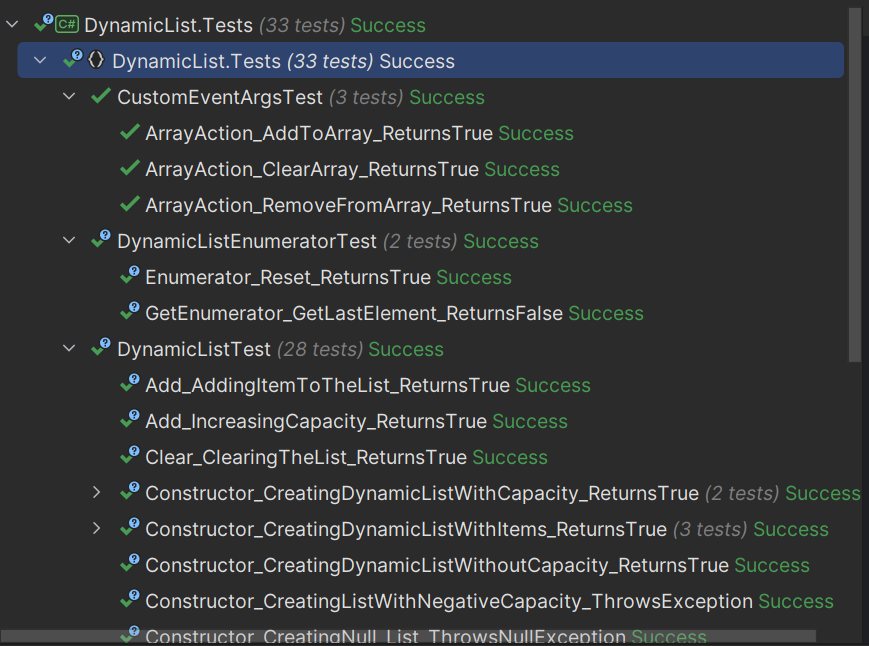
namespace DynamicList.Tests;  
  
public class DynamicListTestBase  
{  
 protected DynamicList<int> \_dynamicList;  
  
 protected DynamicListTestBase()  
 {  
 \_dynamicList = new DynamicList<int>(new int[] {1, 2, 3, 4, 5});  
 }  
}

using DynamicList.CustomEventArgs;  
  
namespace DynamicList.Tests;  
  
public class DynamicListTest : DynamicListTestBase  
{  
   
 #region Constructor  
  
 [Fact]  
 public void Constructor\_CreatingDynamicListWithoutCapacity\_ReturnsTrue()  
 {  
 var collection = new DynamicList<int>();  
  
 Assert.*Empty*(collection);  
 }  
  
 [Fact]  
 public void Constructor\_CreatingListWithNegativeCapacity\_ThrowsException()  
 {  
 var capacity = -1;  
  
 Assert.*Throws*<ArgumentOutOfRangeException>(() => new DynamicList<int>(capacity));  
 }  
  
 [Fact]  
 public void Constructor\_CreatingNull\_List\_ThrowsNullException()  
 {  
 IEnumerable<int> collection = null;  
  
 Assert.*Throws*<ArgumentNullException>(  
 () => collection = new DynamicList<int>(collection));  
 }  
  
 [Theory]  
 [InlineData(10)]  
 [InlineData(0)]  
 public void Constructor\_CreatingDynamicListWithCapacity\_ReturnsTrue(int capacity)  
 {  
 var collection = new DynamicList<int>(capacity);  
  
 Assert.*Empty*(collection);  
 }  
  
 [Theory]  
 [InlineData(new int[] {1, 2, 3, 4, 5})]  
 [InlineData(new int[] {-1, -2, -3})]  
 [InlineData(new int[] {0, 0, 0, 0, 0})]  
 public void Constructor\_CreatingDynamicListWithItems\_ReturnsTrue(IEnumerable<int> items)  
 {  
 var collection = new DynamicList<int>(items);  
   
 Assert.*Equal*(items, collection);  
 }  
 #endregion  
  
 #region Add  
 [Fact]  
 public void Add\_AddingItemToTheList\_ReturnsTrue()  
 {  
 var item = 6;  
   
 \_dynamicList.Add(item);  
   
 Assert.*Contains*(item, \_dynamicList);  
 }  
   
 [Fact]  
 public void Add\_IncreasingCapacity\_ReturnsTrue()  
 {  
 var item = 6;  
 var expectedCapacity = 1;  
 var initialCapacity = \_dynamicList.Count;  
   
 \_dynamicList.Add(item);  
   
 Assert.*Equal*(expectedCapacity, \_dynamicList.Count - initialCapacity);  
   
 }  
 #endregion  
   
 #region Insert  
 [Fact]  
 public void Insert\_AddingItemToTheList\_ReturnsTrue()  
 {  
 var item = 6;  
   
 \_dynamicList.Insert(5, item);  
   
 Assert.*Contains*(item, \_dynamicList);  
 }  
  
 [Fact]  
 public void Insert\_PassingInvalidIndex\_ThrowsException()  
 {  
 var index = 6;  
 var item = 6;  
  
 Assert.*Throws*<InvalidOperationException>(  
 () => \_dynamicList.Insert(index, item));  
 }  
   
 #endregion  
   
 #region Remove  
   
 [Fact]  
 public void Remove\_RemovingItemFromTheList\_ReturnsTrue()  
 {  
 var item = 5;  
   
 \_dynamicList.Remove(item);  
   
 Assert.*DoesNotContain*(item, \_dynamicList);  
 }  
 #endregion  
  
 #region RemoveAt  
  
 [Fact]  
 public void RemoveAt\_RemoveAtInvalidIndex\_ThrowsException()  
 {  
 var index = 6;  
  
 Assert.*Throws*<ArgumentOutOfRangeException>(  
 () => \_dynamicList.RemoveAt(index));  
  
 }  
   
 [Fact]  
 public void RemoveAt\_RemovingItemFromTheList\_ReturnsTrue()  
 {  
 var index = 4;  
   
 \_dynamicList.RemoveAt(index);  
   
 Assert.*DoesNotContain*(\_dynamicList[index], \_dynamicList);  
 }  
  
 #endregion  
  
  
 #region Clear  
  
 [Fact]  
 public void Clear\_ClearingTheList\_ReturnsTrue()  
 {  
 \_dynamicList.Clear();  
   
 Assert.*Empty*(\_dynamicList);  
 }  
  
 #endregion  
  
 #region IndexOf  
  
 [Fact]  
 public void IndexOf\_FindingItemInTheList\_ReturnsTrue()  
 {  
 var item = 5;  
   
 var index = \_dynamicList.IndexOf(item);  
   
 Assert.*Equal*(4, index);  
 }  
  
  
 #endregion  
   
   
 #region Contains  
   
 [Fact]  
 public void Contains\_FindingItemInTheList\_ReturnsTrue()  
 {  
 var item = 5;  
   
 var contains = \_dynamicList.Contains(item);  
   
 Assert.*True*(contains);  
 }  
  
 [Fact]  
 public void Contains\_NotFindingItemInTheList\_ReturnsTrue()  
 {  
 var item = 0;  
   
 Assert.*DoesNotContain*(item, \_dynamicList);  
 }  
 #endregion  
  
 #region CopyTo  
   
 [Fact]  
 public void CopyTo\_CopyingTheList\_ReturnsTrue()  
 {  
 var lst = new int[10];  
   
 \_dynamicList.CopyTo(lst, 0);  
   
 Assert.*Equal*(new int[]{1, 2, 3, 4, 5, 0, 0, 0, 0, 0}, lst);  
 }  
   
 [Fact]  
 public void CopyTo\_PassingInvalidIndex\_ThrowsException()  
 {  
 var lst = new int[10];  
 var index = 6;  
  
 Assert.*Throws*<ArgumentException>(() => \_dynamicList.CopyTo(lst, index));  
 }  
  
 #endregion  
  
  
 #region Count  
  
 [Fact]  
 public void Count\_CountingTheList\_ReturnsTrue()  
 {  
 var expectedCount = 5;  
   
 var count = \_dynamicList.Count;  
   
 Assert.*Equal*(expectedCount, count);  
 }  
   
 #endregion  
  
 #region Set  
  
 [Theory]  
 [InlineData (0, 1)]  
 [InlineData (1, 2)]  
 [InlineData (2, 3)]  
 [InlineData (3, 4)]  
 [InlineData (4, 5)]  
 public void Set\_SettingItemInTheList\_ReturnsTrue(int index, int item)  
 {  
 \_dynamicList[index] = item;  
   
 Assert.*Equal*(item, \_dynamicList[index]);  
 }  
  
 [Fact]  
 public void Set\_PassingInvalidIndex\_ThrowsException()  
 {  
 var index = 6;  
 var item = 6;  
   
 Assert.*Throws*<ArgumentException>(()=> \_dynamicList[index] = item);  
 }  
  
 #endregion  
   
}

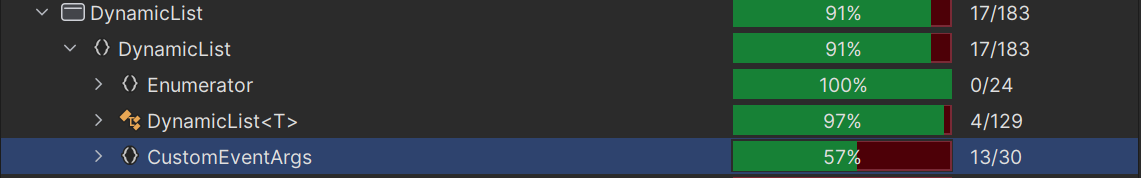
using DynamicList.CustomEventArgs;  
namespace DynamicList.Tests;  
  
  
public class DynamicListEnumeratorTest : DynamicListTestBase  
{  
   
 [Fact]  
 public void GetEnumerator\_GetLastElement\_ReturnsFalse()  
 {  
 var collection = new DynamicList<int>(new int[] {1, 2});  
 var enumerator = collection.GetEnumerator();  
  
 enumerator.MoveNext();  
 enumerator.MoveNext();  
   
 Assert.*False*(enumerator.MoveNext());  
 }  
  
 [Fact]  
 public void Enumerator\_Reset\_ReturnsTrue()  
 {  
 var enumerator = \_dynamicList.GetEnumerator();  
 var expectedItem = 1;  
  
 enumerator.MoveNext();  
 enumerator.MoveNext();  
 enumerator.Reset();  
   
 Assert.*Equal*(expectedItem, enumerator.Current);  
 }  
  
}

using DynamicList.CustomEventArgs;  
  
namespace DynamicList.Tests;  
  
public class CustomEventArgsTest : DynamicListTestBase  
{  
  
 [Fact]  
 public void ArrayAction\_AddToArray\_ReturnsTrue()  
 {  
 var newItem = 6;  
 TestArrayAction(ArrayAction.*Add*, list => \_dynamicList.Insert(5 ,newItem),  
 eventArgs => Assert.*Equal*(eventArgs.Item, newItem));  
 }  
   
 [Fact]  
 public void ArrayAction\_RemoveFromArray\_ReturnsTrue()  
 {  
 var item = 5;  
 TestArrayAction(ArrayAction.*Remove*, list => \_dynamicList.Remove(item),  
 eventArgs => Assert.*Equal*(eventArgs.Item, item));  
 }  
   
 [Fact]  
 public void ArrayAction\_ClearArray\_ReturnsTrue()  
 {  
 TestArrayAction(ArrayAction.*Clear*, list => \_dynamicList.Clear(),  
 eventArgs => Assert.*Empty*(\_dynamicList));  
 }  
   
 private void TestArrayAction(ArrayAction action, Action<DynamicList<int>> arrayAction,  
 Action<NotifyArrayChangedEventArgs<int>> arrayChanged)  
 {  
 NotifyArrayChangedEventArgs<int>? eventArgs = null;  
 \_dynamicList.ArrayChanged+= args => eventArgs = args;  
   
 arrayAction(\_dynamicList);  
   
 Assert.*NotNull*(eventArgs);  
 Assert.*Equal*(eventArgs.Action, action);  
 arrayChanged(eventArgs!);  
 }  
}

**Виконання тестів:**

****

**Покриття тестів:**

****