Лабораторна робота №2

Тема: Використання методів розширень та узагальнень у C#.

Мета роботи: навчитися використовувати методи розширення та узагальнення у мові програмування C#.

Виділений час: 12 годин (4 години лабораторних робіт та 8 годин самостійної роботи).

2. Реалізувати методи розширення:
   - для класу String:
     - інвертування рядка;
     - підрахунок кількості входжень заданого у параметрі символа у рядок.
   - для одновимірних масивів:
     - метод, що визначає скільки разів зустрічається задане значення у масиві (метод має працювати для одновимірних масивів усіх типів, для реалізації даного методу розширення використайте узагальнення та їх обмеження за допомогою “where”);
     - метод, що повертає новий масив такого ж типу і формує його з унікальних елементів (видаляє повтори);
   - Написати код для демонстрації роботи реалізованих методів розширення.

StringExtentions:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ExtentionsLibrary
{
    public static class StringExtentions
    {
        public static string ReverseString(this string str)
        {
            string revers = "";
            for (int i = str.Length - 1; i >= 0; i--)
                revers += str[i];
            return revers;
        }
```

					ЖИТОМИРСЬКА ПОЛІТЕХНІКА.23.121.13.000 – Лр.1		
Змн.	Арк.	№ докум.	Підпис	Дата			
Розроб.		Лаговський А.В.			Звіт з лабораторної роботи №2	Літ.	Арк. 1 / Аркушів
Перевір.		Морозов А. В.					
Реценз.						ФІКТ, гр. ІПЗ-22-1	
Н. Контр.							
Зав.каф.							

```
        public static int CountOccurrences(this string str, char character)
        {
            int count = 0;
            foreach (char c in str)
            {
                if (c == character)
                    count++;
            }
            return count;
        }
    }
}
```

ArrayExtentions:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading.Tasks;

namespace ExtentionsLibrary
{
    public static class ArrayExtentions
    {
        public static int CountValueOccurrences<T>(this T[] array, T objct)
        {
            int counter = 0;
            foreach (T item in array)
            {
                string item2 = item.ToString();
                string objct2 = objct.ToString();
                if (item2 == objct2)
                {
                    counter++;
                }
            }
            return counter;
        }

        public static T[] ReturnUniqueArray<T>(this T[] array)
        {
            HashSet<T> set = new HashSet<T>(array);
            T[] result = set.ToArray();
            return result;
        }
    }
}
```

```
PT 1.

Before reverse: hello, world
After reverse:
dlrow ,olleh

PT 2.

m's in text: 2

PT 3.

Array elements: 22 1 23 51 123 22
Occurrences of 22 in array: 2

PT 4.

Array elements: hello world seesharp
Occurrences of 'hello' in array: 1

PT 5.

Array elements: 22 1 23 51 123 22
Unique values only:
22 1 23 51 123
```

3. Реалізувати узагальнені класи для:
   - Реалізувати узагальнений клас для зберігання “розширеного словника” (для ключа передбачається два значення).
     ExtendedDictionary<T, U, V>, де T - тип даних ключа, U - тип даних першого значення, V - тип даних другого значення. Передбачити операції:
     - додавання елемента у словник;
     - видалення елемента з словника за заданим ключем;

     - перевірка наявності елемента із заданим ключем;
     - перевірка наявності елемента із заданим значенням (значення1 та значення2);
     - повернення елемента за заданим ключем (реалізувати операцію індексування);
     - властивість, що повертає кількість елементів;
     Представлення елемента словника реалізувати у вигляді окремого класу ExtendedDictionaryElement<T, U, V>, передбачивши властивості для доступу до ключа, першого та другого значення.
     Словник повинен мати можливість використання у циклах foreach: foreach(var elem in array) { … }
   - Написати код для демонстрації роботи з реалізованими узагальненими класами.

ExtendedDictionary:

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ExtensionsLibrary
{
    public class ExtendedDictionary<T, U, V> : IEnumerator, IEnumerable  where T:class
where U: class where V: class
    {
        private Dictionary<T, U> _dictionary1;
        private Dictionary<T, V> _dictionary2;
        private int position = -1;

        public int Count
        {
            get { return _dictionary1.Count; }
        }

        public ExtendedDictionary()
        {
            _dictionary1 = new Dictionary<T, U>();
            _dictionary2 = new Dictionary<T, V>();
        }

        public void AddElements(T key, U val1, V val2)
        {
            _dictionary1.Add(key, val1);
            _dictionary2.Add(key, val2);
        }
```

```
 public void RemoveElemetsByKey(T key)
        {
            _dictionary1.Remove(key);
            _dictionary2.Remove(key);
        }

        public bool CheckIfDictContainsKey(T key)
        {
            return _dictionary1.ContainsKey(key) && _dictionary2.ContainsKey(key);
        }

        public bool CheckKeyContainsValue(U val1, V val2)
        {
            T? keyValue1 = _dictionary1.FirstOrDefault(vl1 => vl1.Value == val1).Key;
            T? keyValue2 = _dictionary2.FirstOrDefault(vl2 => vl2.Value == val2).Key;
            if(keyValue1 == keyValue2)
            {
                return true;
            }
            if(keyValue1 == null && keyValue2 == null || keyValue1 == null | keyValue2 ==
null)
            {
                return false;
            }
            return false;
        }

        public ExtendedDictionaryValues<U, V> this[T key]
        {
            get
            {
                return new ExtendedDictionaryValues<U, V>(_dictionary1[key],
_dictionary2[key]);
            }
        }

        public override string ToString()
        {
            StringBuilder str = new StringBuilder();
            foreach(T key in _dictionary1.Keys)
            {
                str.Append($"key: {key}\nvalue1: {_dictionary1[key]}\nvalue2:
{_dictionary2[key]}\n");
            }
            return str.ToString();
        }
        public object Current
        {
            get
            {
                try
                {
                    KeyValuePair<T, U> keyValue1 = _dictionary1.ElementAt(position);
                    KeyValuePair<T, V> keyValue2 = _dictionary2.ElementAt(position);
                    return new ExtendedDictionaryAll<T, U, V>(keyValue1.Key,
keyValue1.Value, keyValue2.Value);

                }catch(IndexOutOfRangeException)
                {
                    throw new InvalidOperationException();
                }
            }
        }
```

```
        public bool MoveNext()
        {
            position++;
            return (position < _dictionary1.Count);
        }

        public void Reset()
        {
            position = -1;
        }

        public IEnumerator GetEnumerator()
        {
            return (IEnumerator)this;
        }

    }
}
```

ExtendedDictionary2:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ExtensionsLibrary
{
    public class ExtendedDictionary2<T, U, V> where T : class where U : class where V :
class
    {
        public T key { get; set; }
        public U value1 { get; set; }
        public V value2 { get; set; }

        public ExtendedDictionary2(T key, U value1, V value2)
        {
            this.key = key;
            this.value1 = value1;
            this.value2 = value2;
        }

        public override string ToString()
        {
            return $"key: {key}\nvalue1: {value1}\nvalue2: {value2}";
        }
    }
}
```

ExtendedDictionaryValues:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ExtensionsLibrary
{
    public class ExtendedDictionaryValues<U, V> where U : class where V : class
    {
        public U value1 { get; set; }
        public V value2 { get; set; }
```

```
        public ExtendedDictionaryValues(U value1, V value2)
        {
            this.value1 = value1;
            this.value2 = value2;
        }

        public override string ToString()
        {
            return $"value1: {value1}\nvalue2: {value2}\n";
        }
    }
}
```

```
PT 6.

Extended Dictionary:
key: hello
value1: One
value2: apple
key: thursday
value1: Two
value2: peach
key: bye
value1: Three
value2: banana

Does dictionary contain key 'hello'? True
Value for key 'bye': value1: Three
value2: banana


Iterating over the dictionary:
key: hello
value1: One
value2: apple
key: thursday
value1: Two
value2: peach
key: bye
value1: Three
value2: banana
```

ConsoleApp:

```
using ExtensionsLibrary;
using ExtentionsLibrary;
using System;

namespace task2
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("PT 1.\n");
            string txt = "hello, world";
            Console.WriteLine($"Before reverse: {txt}");
            Console.WriteLine("After reverse: ");
            Console.WriteLine($"{txt.ReverseString()}\n");

            Console.WriteLine("PT 2.\n");

            string txt2 = "programming";
            char targetChar = 'm';
            int count = txt2.CountOccurrences(targetChar);
            Console.WriteLine($"{targetChar}'s in text: {count}\n");

            Console.WriteLine("PT 3.\n");

            int[] intArray = { 22, 1, 23, 51, 123, 22 };
            Console.Write("Array elements: ");
            foreach (int i in intArray)
            {
                Console.Write($"{i} ");
            }
            int target = 22;
            Console.WriteLine("\nOccurrences of 22 in array: " +
intArray.CountValueOccurrences(target) + "\n");

            Console.WriteLine("PT 4.\n");

            string[] stringArray = { "hello", "world", "seesharp" };
            Console.Write("Array elements: ");
            foreach (string i in stringArray)
            {
                Console.Write($"{i} ");
            }
            string targetWord = "hello";
            Console.WriteLine("\nOccurrences of 'hello' in array: " +
stringArray.CountValueOccurrences(targetWord) + "\n");

            Console.WriteLine("PT 5.\n");
            Console.Write("Array elements: ");
            foreach (int i in intArray)
            {
                Console.Write($"{i} ");
            }
            Console.WriteLine("\nUnique values only: ");
            int[] uniqueArray = intArray.ReturnUniqueArray();
            foreach (int i in uniqueArray)
            {
                Console.Write($"{i} ");
            }
```

```
            Console.WriteLine("\n\nPT 6.\n");
            var extendedDictionary = new ExtendedDictionary<string, string, string>();
            extendedDictionary.AddElements("hello", "One", "apple");
            extendedDictionary.AddElements("thursday", "Two", "peach");
            extendedDictionary.AddElements("bye", "Three", "banana");

            Console.WriteLine("Extended Dictionary:");
            Console.WriteLine(extendedDictionary);

            Console.WriteLine($"Does dictionary contain key 'hello'?
{extendedDictionary.CheckIfDictContainsKey("hello")}");

            string keyToCheck = "bye";
            Console.WriteLine($"Value for key '{keyToCheck}':
{extendedDictionary[keyToCheck]}");

            Console.WriteLine("\nIterating over the dictionary:");
            foreach (var item in extendedDictionary)
            {
                Console.WriteLine(item);
            }

            Console.ReadKey();
        }
    }
}
```

Висновок: під час виконання даної лабораторної роботи я навчився використовувати методи розширення та узагальнення у мові програмування C#.

GitHub: https://github.com/AdamLahovskyi/DotNetLab2