7

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Практикум №1

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

на тему: «Узагальнені типи (Generic) з підтримкою подій. Колекції»

|  |  |
| --- | --- |
|  | Виконав:  студент 3 курсу  групи ІП-11 ФІОТ  Рябов Ю. І. |

Київ-2023

**Варіант 2**

**Мета лабораторної роботи** – навчитися проектувати та реалізовувати

узагальнені типи, а також типи з підтримкою подій.

**Завдання**:

1. Розробити клас власної узагальненої колекції, використовуючи

стандартні інтерфейси колекцій із бібліотек System.Collections та

System.Collections.Generic. Стандартні колекції при розробці власної

не застосовувати. Для колекції передбачити методи внесення даних

будь-якого типу, видалення, пошуку та ін. (відповідно до типу

колекції).

2. Додати до класу власної узагальненої колекції підтримку подій та

обробку виключних ситуацій.

3. Опис класу колекції та всіх необхідних для роботи з колекцією типів

зберегти у динамічній бібліотеці.

4. Створити консольний додаток, в якому продемонструвати

використання розробленої власної колекції, підписку на події

колекції.

Колекція згідно варіанту – черга.

**Гіт репозиторій лабораторної роботи:**

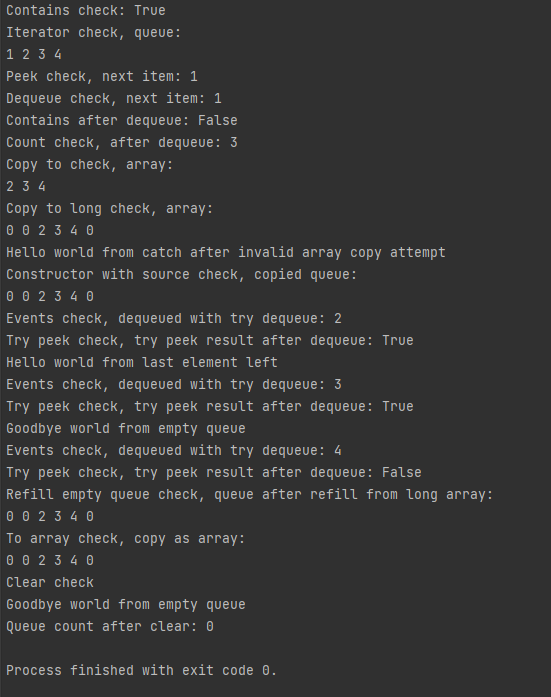
<https://github.com/YuraRiabov/WebNet1Collections>

**Код лабораторної роботи:**

using System.Collections;  
using System.Diagnostics.CodeAnalysis;  
  
namespace Collections;  
  
public class MyQueue<T> : IEnumerable<T>, ICollection  
{  
 private MyQueueNode? \_first;  
 private MyQueueNode? \_last;  
  
 public event Action? LastElementRemoved;  
 public event Action? LastElementLeft;  
  
 public int Count  
 {  
 get  
 {  
 var count = 0;  
 var currentItem = \_first;  
 while (currentItem is not null)  
 {  
 count++;  
 currentItem = currentItem.Next;  
 }  
  
 return count;  
 }  
 }  
 public bool IsSynchronized => false;  
 public object SyncRoot => this;  
  
 public MyQueue()  
 {  
 \_first = null;  
 \_last = null;  
 }  
  
 public MyQueue(IEnumerable<T> source)  
 {  
 foreach (var value in source)  
 {  
 Enqueue(value);  
 }  
 }  
  
 public void Enqueue(T value)  
 {  
 var newNode = new MyQueueNode(value);  
   
 if (\_first is null)  
 {  
 \_first = newNode;  
 return;  
 }  
  
 if (\_last is null)  
 {  
 \_first.Next = newNode;  
 \_last = newNode;  
 return;  
 }  
  
 \_last.Next = newNode;  
 \_last = newNode;  
 }  
  
 public T Dequeue()  
 {  
 if (\_first is null)  
 {  
 throw new InvalidOperationException();  
 }  
  
 return DequeueInternal();  
 }  
  
 public bool TryDequeue([MaybeNullWhen(false)] out T value)  
 {  
 if (\_first is null)  
 {  
 value = default;  
 return false;  
 }  
  
 value = DequeueInternal();  
 return true;  
 }  
  
 public T Peek()  
 {  
 if (\_first is null)  
 {  
 throw new InvalidOperationException();  
 }  
  
 return \_first.Value;  
 }  
  
 public bool TryPeek([MaybeNullWhen(false)] out T value)  
 {  
 if (\_first is null)  
 {  
 value = default;  
 return false;  
 }  
  
 value = \_first.Value;  
 return true;  
 }  
  
 public void Clear()  
 {  
 \_first = null;  
 \_last = null;  
 LastElementRemoved?.Invoke();  
 }  
  
 public bool Contains(T item)  
 {  
 var currentItem = \_first;  
 while (currentItem is not null)  
 {  
 if (currentItem.Value!.Equals(item))  
 {  
 return true;  
 }  
  
 currentItem = currentItem.Next;  
 }  
  
 return false;  
 }  
  
 public T[] ToArray()  
 {  
 var array = new T[Count];  
 CopyToInternal(array, 0);  
 return array;  
 }  
   
 public void CopyTo(T[] array, int index)  
 {  
 if (!IsValidArrayLengthIndexForCopy(array, index))  
 {  
 throw new ArgumentOutOfRangeException();  
 }  
   
 CopyToInternal(array, index);  
 }  
   
 void ICollection.CopyTo(Array array, int index)  
 {  
 if (!IsValidArrayLengthIndexForCopy(array, index))  
 {  
 throw new ArgumentOutOfRangeException();  
 }  
  
 if (!IsValidArrayTypeForCopy(array))  
 {  
 throw new ArgumentException(nameof(array));  
 }  
   
 CopyToInternal(array, index);  
 }  
   
 public IEnumerator<T> GetEnumerator()  
 {  
 var currentNode = \_first;  
 while (currentNode is not null)  
 {  
 yield return currentNode.Value;  
 currentNode = currentNode.Next;  
 }  
 }  
  
 IEnumerator IEnumerable.GetEnumerator()  
 {  
 return GetEnumerator();  
 }  
  
 private T DequeueInternal()  
 {  
 var first = \_first;  
 \_first = \_first!.Next;  
  
 if (\_first is not null && !\_first.HasNext)  
 {  
 \_last = null;  
 LastElementLeft?.Invoke();  
 }  
  
 if (\_first is null)  
 {  
 LastElementRemoved?.Invoke();  
 }  
  
 return first!.Value;  
 }  
  
 private void CopyToInternal(Array array, int index)  
 {  
 var currentIndex = index;  
 var currentItem = \_first;  
 while (currentItem is not null)  
 {  
 array.SetValue(currentItem.Value, currentIndex);  
 currentItem = currentItem.Next;  
 currentIndex++;  
 }  
 }  
  
 private bool IsValidArrayLengthIndexForCopy(Array array, int index)  
 {  
 if (array is null)  
 {  
 return false;  
 }  
  
 if (index < 0 || index >= array.Length)  
 {  
 return false;  
 }  
  
 return array.Length - index >= Count;  
 }  
  
 private bool IsValidArrayTypeForCopy(Array array)  
 {  
 return array.Rank == 1 && array.GetLowerBound(0) == 0;  
 }  
  
 private class MyQueueNode  
 {  
 public T Value { get; }  
 public MyQueueNode? Next { get; set; }  
 public bool HasNext => Next is not null;  
  
 public MyQueueNode(T value)  
 {  
 Value = value;  
 }  
 }  
}

using Collections;  
  
void OutputCollection(IEnumerable<int> collection)  
{  
 foreach (var item in collection)  
 {  
 Console.Write($"{item} ");  
 }  
  
 Console.WriteLine();  
}  
  
void OnLastItemLeft() => Console.WriteLine("Hello world from last element left");  
void OnEmptyQueue() => Console.WriteLine("Goodbye world from empty queue");  
  
var queue = new MyQueue<int>();  
  
queue.Enqueue(1);  
queue.Enqueue(2);  
queue.Enqueue(3);  
queue.Enqueue(4);  
  
Console.WriteLine($"Contains check: {queue.Contains(3)}");  
  
  
Console.WriteLine("Iterator check, queue:");  
OutputCollection(queue);  
  
  
Console.WriteLine($"Peek check, next item: {queue.Peek()}");  
var dequeued = queue.Dequeue();  
Console.WriteLine($"Dequeue check, next item: {dequeued}");  
Console.WriteLine($"Contains after dequeue: {queue.Contains(dequeued)}");  
Console.WriteLine($"Count check, after dequeue: {queue.Count}");  
  
var array = new int[3];  
  
queue.CopyTo(array, 0);  
  
Console.WriteLine("Copy to check, array:");  
OutputCollection(array);  
  
var longArray = new int[6];  
queue.CopyTo(longArray, 2);  
Console.WriteLine("Copy to long check, array:");  
OutputCollection(longArray);  
  
var shortArray = new int[2];  
try  
{  
 queue.CopyTo(shortArray, 0);  
}  
catch(ArgumentOutOfRangeException)  
{  
 Console.WriteLine("Hello world from catch after invalid array copy attempt");  
}  
  
var copiedQueue = new MyQueue<int>(longArray);  
Console.WriteLine($"Constructor with source check, copied queue:");  
OutputCollection(copiedQueue);  
  
queue.LastElementLeft += OnLastItemLeft;  
queue.LastElementRemoved += OnEmptyQueue;  
for (int i = 0; i < 3; i++)  
{  
 queue.TryDequeue(out dequeued);  
 Console.WriteLine($"Events check, dequeued with try dequeue: {dequeued}");  
 Console.WriteLine($"Try peek check, try peek result after dequeue: {queue.TryPeek(out \_)}");  
}  
  
foreach (var item in longArray)  
{  
 queue.Enqueue(item);  
}  
Console.WriteLine("Refill empty queue check, queue after refill from long array:");  
OutputCollection(queue);  
  
var queueAsArray = queue.ToArray();  
Console.WriteLine("To array check, copy as array:");  
OutputCollection(queueAsArray);  
  
Console.WriteLine("Clear check");  
queue.Clear();  
Console.WriteLine($"Queue count after clear: {queue.Count}");

**Скріншот екрану виконання:**



**Висновок:**

Отже, ми освоїли вбудовані колекції, узагальнені типи, обробку виключних ситуацій та подій, створивши власну реалізацію базової колекції – черги.