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

Лабораторна робота №1

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

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

Викладач: Виконав студент:

Бардін В. Лазюта Олексій

групи ІП-15 ФІОТ

Київ-2023

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).

1. Додати до класу власної узагальненої колекції підтримку подій та

обробку виключних ситуацій.

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

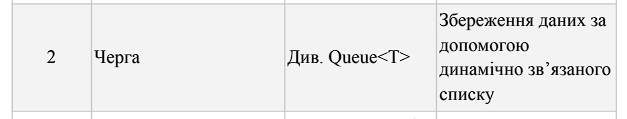
зберегти у динамічній бібліотеці.

1. Створити консольний додаток, в якому продемонструвати

використання розробленої власної колекції, підписку на події

колекції.

Варіант - 2



Код програми:

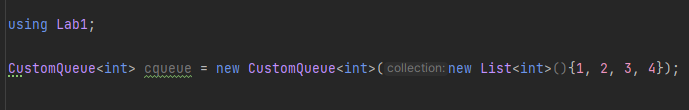
Файл Node.cs:

namespace Lab1;  
  
public class Node<T>  
{  
 public Node<T> next;  
 public T value;  
  
 public Node(T value)  
 {  
 next = null;  
 this.value = value;  
 }  
}

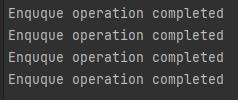
Файл CustomQueue.cs:

using System.Collections;  
  
  
namespace Lab1;  
  
public class CustomQueue <T>: IEnumerable<T> where T: IComparable  
{  
 private int \_size;  
 private Node<T> \_headNode;  
 private Node<T> \_tailNode;  
  
 private event EventHandler? OnEnqueued = (sender, eventArgs)   
 => Console.WriteLine("Enquque operation completed");  
 private event EventHandler? OnDequeued = (sender, eventArgs)   
 => Console.WriteLine("Dequeue operation completed");  
 private event EventHandler? OnCleared = (sender, eventArgs)   
 => Console.WriteLine("Clear operation completed");  
 private event EventHandler? OnCopiedTo = (sender, eventArgs)   
 => Console.WriteLine("Copy operation completed");  
 private event EventHandler? OnReversed = (sender, eventArgs)   
 => Console.WriteLine("Reverse operation completed");  
  
 public int Count => \_size;  
   
 public CustomQueue()  
 {  
 \_headNode = null;  
 \_tailNode = null;  
 \_size= 0;  
 }  
  
 public CustomQueue(IEnumerable<T> collection)  
 {  
 \_headNode = null;  
 \_tailNode = null;  
 \_size = 0;  
 foreach (var item in collection)  
 {  
 Enqueue(item);  
 }  
 }  
   
 IEnumerator<T> IEnumerable<T>.GetEnumerator()  
 {   
 Node<T> currentNode = \_headNode;  
 while (currentNode != null)  
 {  
 yield return currentNode.value;  
 currentNode = currentNode.next;  
 }  
 }  
  
 public IEnumerator GetEnumerator()  
 {  
 return new MyEnumerator(\_headNode);  
 }  
  
 public void CopyTo(T[] array, int index)  
 {  
 Node<T> activeNode = \_headNode;  
 if (array == null)  
 throw new ArgumentNullException("Value can not be null");  
 else if (index < 0 || index > array.Length - 1)  
 throw new ArgumentOutOfRangeException("Index can not be out of range");  
 else if (array.Length - index + 1 < \_size)  
 throw new ArgumentException("Number of the elements in CustomQueue is greater than the Array can contain");  
   
 while (activeNode != null)  
 {  
 array[index] = activeNode.value;  
 activeNode = activeNode.next;  
 index++;  
 }  
 OnCopiedTo?.Invoke(this, new EventArgs());  
 }  
   
 public void Enqueue(T item)  
 {  
 Node<T> node = new Node<T>(item);  
 if (\_headNode == null)  
 {  
 \_headNode = node;  
 \_tailNode = node;  
 }  
  
 else  
 {  
 \_tailNode.next = new Node<T>(item);  
 \_tailNode = \_tailNode.next;  
 }  
   
 \_size++;  
 OnEnqueued?.Invoke(this, new EventArgs());  
 }  
   
 public T Dequeue()  
 {  
 if (\_headNode == null)  
 throw new InvalidOperationException("CustomQueue is empty");  
  
 T headNodeValue = \_headNode.value;  
 \_headNode = \_headNode.next;  
  
 \_size--;  
 OnDequeued?.Invoke(this, new EventArgs());  
 return headNodeValue;  
   
 }  
  
 public void Clear()  
 {  
 \_headNode = null;  
 \_tailNode = null;  
 OnCleared?.Invoke(this, new EventArgs());  
 }  
  
 public bool Contains(T item)  
 {  
 Node<T> activeNode = \_headNode;  
 while (activeNode != null)  
 {  
 if (activeNode.value.Equals(item))  
 return true;  
 activeNode = activeNode.next;  
 }  
  
 return false;  
 }  
  
 public T Peek()  
 {  
 if (\_size == 0)  
 throw new InvalidOperationException("CustomQueue is empty");  
   
 return \_headNode.value;  
 }  
  
 public T[] ToArray()  
 {  
 Node<T> activeNode = \_headNode;  
 T[] array = new T[\_size];  
 for (int i = 0; i < \_size; i++)  
 {  
 array[i] = activeNode.value;  
 activeNode = activeNode.next;  
 }  
  
 return array;  
 }  
  
 public CustomQueue<T> Reverse()  
 {  
 CustomQueue<T> reversed\_queue;  
 IEnumerable<T> tempArray;  
 tempArray = this.ToArray().Reverse();  
 reversed\_queue = new CustomQueue<T>(tempArray);  
 OnReversed?.Invoke(this, new EventArgs());  
 return reversed\_queue;  
 }  
   
 private class MyEnumerator: IEnumerator<T>   
 {  
 private Node<T> \_activeNode;  
 private Node<T> \_headNode;  
 private bool \_beganEnumerating = false;  
  
 public T Current => \_activeNode.value;  
  
 object IEnumerator.Current => Current;  
  
 public MyEnumerator(Node<T> headNode)  
 {  
 \_headNode = headNode;  
 \_activeNode = \_headNode;  
 }  
   
 public bool MoveNext()  
 {  
 if (\_headNode == null)  
 return false;  
   
 else if(!\_beganEnumerating)  
 {  
 \_beganEnumerating = true;  
 \_activeNode = \_headNode;  
 return true;  
 }  
   
 bool hasNext = \_activeNode.next != null;  
 if (!hasNext)  
 {  
 return false;  
 }  
   
 \_activeNode = \_activeNode.next;  
 return hasNext;  
 }  
  
 public void Reset()  
 {  
 \_activeNode = \_headNode;  
 }  
   
 public void Dispose()  
 {  
   
 }  
 }  
   
}

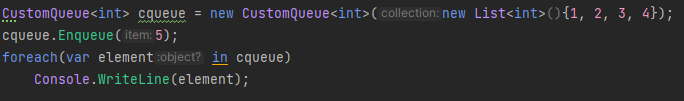
Тестування програми:  
Ініціалізація екзепляру класу за допомогою конструктора:



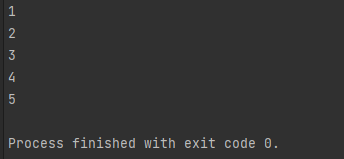
Результат:



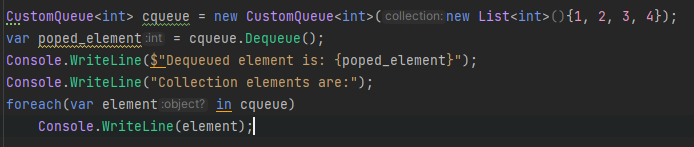
Тестування методу Enqueue:



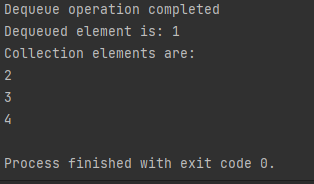
Результат:



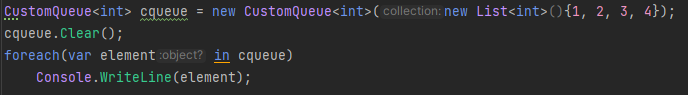
Тестування методу Dequeue:



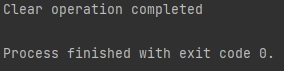
Результат:



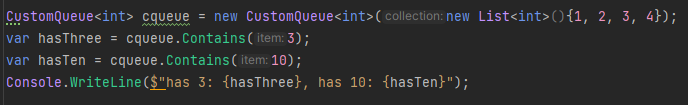
Тестування методу Clear:



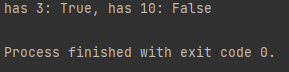
Результат:



Тестування методу Contains:



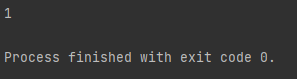
Результат:



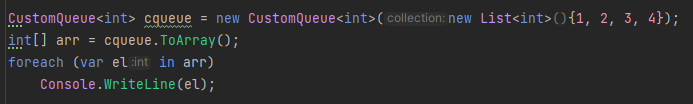
Тестування методу Peek:



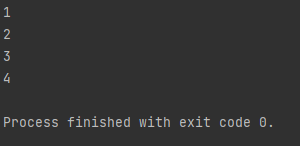
Результат:



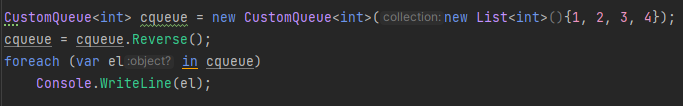
Тестування методу ToArray:



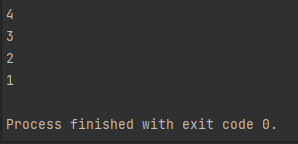
Результат:



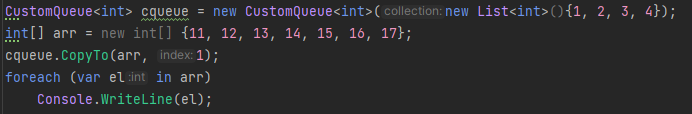
Тестування методу Reverse:



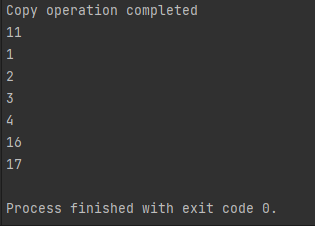
Результат:



Тестування методу CopyTo:



Результат:



Висновок:

У ході виконання даної лабораторної роботи я запрограмував власну колекцію типу черга(Queue) використовуючи узагальнені типи. Розглянув основні методи даної колекції та імплементував їх у власній колекції. Проаналізував інтерфейси від яких наслідуються колекції, та реалізував їх у власній колекції. Створив події та ініціював їх у відповідних методах.

Загалом поглибив знання про колекції та покращив практичні навички їх створення.