**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

по дисциплине: «Объектно-ориентированное проектирование и программирование»

на тему: **«**Коллекции. Обобщенные типы. Итераторы.»**»**

Выполнил: студент гр. ИТИ-21

Тарканов А. В.

Принял: преподаватель

Башаримов Ю. С.

Гомель 2024

**Цель работы**: понять и применить основные принципы разработки коллекций, включая реализацию интерфейсов, итераторов и обобщений. В результате вы создадите свою собственную коллекцию, которую можно эффективно использовать для хранения и управления данными различных типов.

Разработать коллекцию. Обязательное наличие итератора. Также необходимо реализовать интерфейс *I Collection*. Коллекция обязательно должна хранить любые типы данных (классы и структуры). Все коллекции должны быть реализованы без использования стандартных коллекций.

|  |  |
| --- | --- |
| № | Условие |
| 2. | Односвязный список |

**Ход работы**

На рисунке 1 представлено главное окно приложения.

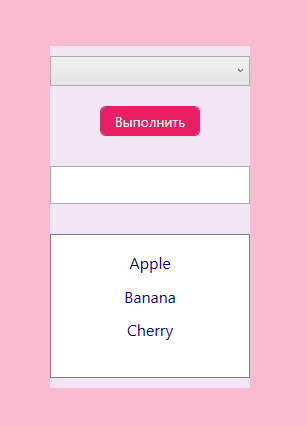


Рисунок 1 – Главное меню

В комбо боксе у нас есть несколько вариантов что можно делать с нашим односвязным списком, это указано на рисунке 2.



Рисунок 2 – Возможности

На рисунке 2 указано, что будет если мы сделаем итерацию в нашем односвязном списке.

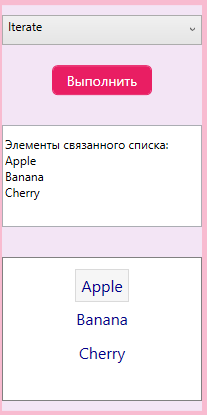


Рисунок 3 – Итерация

На рисунке 4 указана схема работы односвязного списка, где каждый из которых хранит собственно данные и ссылку на следующий узел. В этом случае *Head* – это первый элемент, а *tail* – последний.

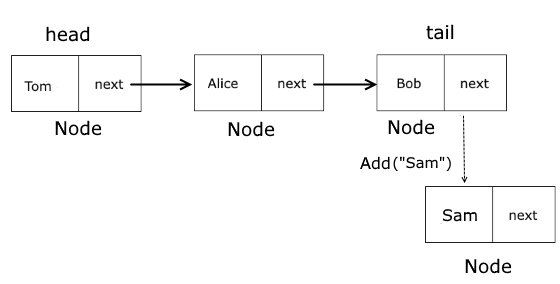


Рисунок 4 – Односвязный список

Также мы можем добавлять и удалять элементы в наш односвязный список, при помощи функций *add* и *remove.* Это показано на рисунке 5.

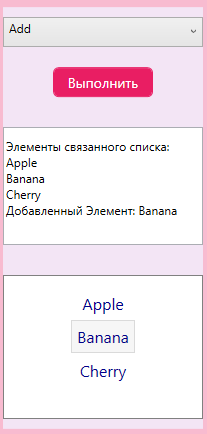


Рисунок 5 – Добавление элементов

Также работает и функция удаления, но, когда мы удаляем элементы и его нет в нашем списке выбивает советующее сообщение о том, что таких элементов нет в списке. Это показано на рисунке 6.

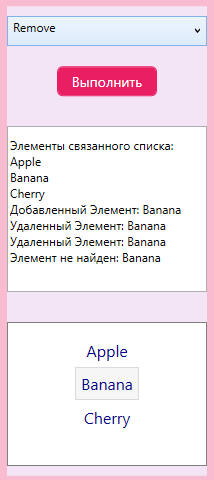


Рисунок 6 – Удаление

При выборе копирования, наш односвязный список копируется в массив и выводится на экран. Это показано на рисунке 7.

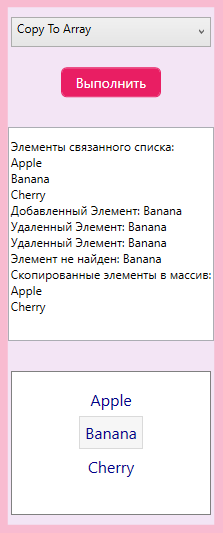


Рисунок 7 – копирование

На рисунке 8 представлены результаты *unit*-тестов.

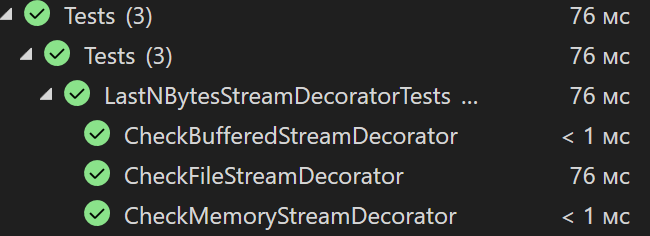


Рисунок 8 – Тестирование

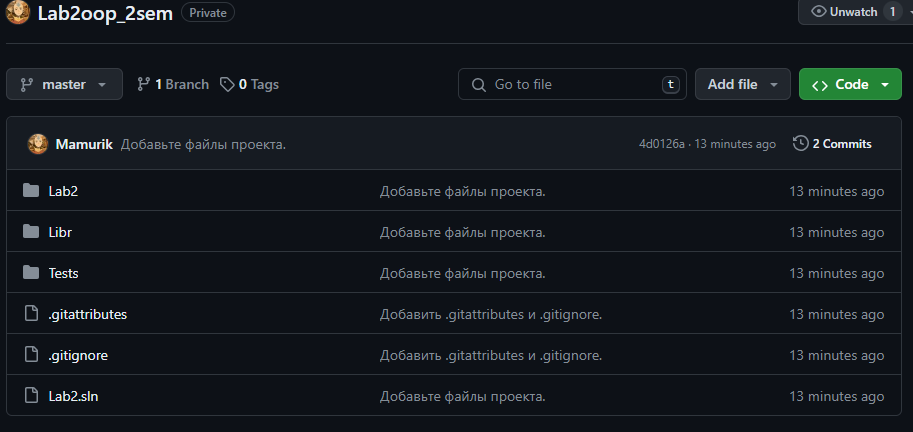


Рисунок 9 – *Github*

В приложении А представлен код программы.

**Вывод:** в результате выполнения лабораторной работы мы углубили понимание принципов разработки коллекций и получили ценный опыт в создании собственных структур данных.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Текст программы**

**MainWindow.xaml**

<Window x:Class="Lab3oop.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:Lab3oop"

mc:Ignorable="d"

Title="MainWindow" WindowStartupLocation="CenterScreen">

<Window.Resources>

<Style TargetType="ComboBox">

<Setter Property="Margin" Value="0 10"/>

<Setter Property="Width" Value="200"/>

<Setter Property="Height" Value="30"/>

</Style>

<Style TargetType="Button">

<Setter Property="Margin" Value="0 10"/>

<Setter Property="Width" Value="100"/>

<Setter Property="Height" Value="30"/>

<Setter Property="Background" Value="#E91E63"/>

<Setter Property="Foreground" Value="White"/>

<Setter Property="FontSize" Value="14"/>

<Setter Property="FontFamily" Value="Segoe UI"/>

<Setter Property="Cursor" Value="Hand"/>

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="Button">

<Border Background="{TemplateBinding Background}" BorderBrush="#E91E63" BorderThickness="1" CornerRadius="5">

<ContentPresenter HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Border>

</ControlTemplate>

</Setter.Value>

</Setter>

<Style.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter Property="Background" Value="#9C27B0"/>

</Trigger>

<Trigger Property="IsPressed" Value="True">

<Setter Property="Background" Value="#673AB7"/>

</Trigger>

</Style.Triggers>

</Style>

<Style TargetType="ScrollViewer">

<Setter Property="Margin" Value="0 10"/>

<Setter Property="MaxWidth" Value="400"/>

<Setter Property="MaxHeight" Value="300"/>

</Style>

<Style TargetType="TextBox">

<Setter Property="IsReadOnly" Value="True"/>

<Setter Property="Margin" Value="0 10"/>

<Setter Property="MaxWidth" Value="400"/>

<Setter Property="MaxHeight" Value="300"/>

</Style>

<Style TargetType="ListBox">

<Setter Property="Margin" Value="0 10"/>

<Setter Property="Width" Value="200"/>

<Setter Property="Height" Value="100"/>

<Setter Property="BorderBrush" Value="Gray"/>

<Setter Property="BorderThickness" Value="1"/>

</Style>

<Style TargetType="ListBoxItem">

<Setter Property="Padding" Value="5"/>

<Setter Property="FontSize" Value="16"/>

<Setter Property="Foreground" Value="DarkBlue"/>

<Setter Property="HorizontalAlignment" Value="Center"/>

<Setter Property="VerticalAlignment" Value="Center"/>

</Style>

<Style TargetType="StackPanel">

<Setter Property="HorizontalAlignment" Value="Center"/>

<Setter Property="VerticalAlignment" Value="Center"/>

</Style>

<Style TargetType="Grid">

<Setter Property="Background" Value="#F8BBD0"/>

</Style>

</Window.Resources>

<Grid>

<StackPanel Background="#F3E5F5">

<ComboBox x:Name="OperationComboBox">

<ComboBoxItem Content="Add"/>

<ComboBoxItem Content="Remove"/>

<ComboBoxItem Content="Iterate"/>

<ComboBoxItem Content="Copy To Array"/>

</ComboBox>

<Button Content="Выполнить" Click="Execute\_Click"/>

<ScrollViewer VerticalScrollBarVisibility="Auto">

<TextBox x:Name="LogTextBox"/>

</ScrollViewer>

<ListBox x:Name="ItemsListBox" Height="144">

<ListBoxItem Content="Apple"/>

<ListBoxItem Content="Banana"/>

<ListBoxItem Content="Cherry"/>

</ListBox>

</StackPanel>

</Grid>

</Window>

**MainWindow.xaml.cs**

using System;

using System.Collections;

using System.Collections.Generic;

using System.Windows;

using System.Windows.Controls;

namespace Lab3oop

{

/// <summary>

/// Узел списка (List node)

/// </summary>

/// <summary>

/// Пример использования коллекции

/// </summary>

public partial class MainWindow : Window

{

private LinkedList<string> linkedList;

public MainWindow()

{

InitializeComponent();

// Создание и заполнение списка

linkedList = new LinkedList<string>

{

"Apple",

"Banana",

"Cherry"

};

}

/// <summary>

/// Добавление элемента в список

/// </summary>

private void AddItem\_Click(object sender, RoutedEventArgs e)

{

string newItem = "New Item";

linkedList.Add(newItem);

AddToLog("Добавленный элемент: " + newItem);

}

/// <summary>

/// Удаление элемента из списка

/// </summary>

private void RemoveItem\_Click(object sender, RoutedEventArgs e)

{

string itemToRemove = "Banana";

bool removed = linkedList.Remove(itemToRemove);

if (removed)

AddToLog("Удаленный Элемент: " + itemToRemove);

else

AddToLog("Элемент не найден:: " + itemToRemove);

}

/// <summary>

/// Итерация по элементам списка

/// </summary>

private void Iterate\_Click(object sender, RoutedEventArgs e)

{

AddToLog("Элементы связанного списка:");

foreach (string item in linkedList)

{

AddToLog(item);

}

}

/// <summary>

/// Выполнение операции в зависимости от выбранного элемента и операции

/// </summary>

private void Execute\_Click(object sender, RoutedEventArgs e)

{

try

{

ListBoxItem selectedItem = (ListBoxItem)ItemsListBox.SelectedItem;

if (selectedItem == null)

{

throw new InvalidOperationException("Не выбран элемент из списка.");

}

string selectedValue = selectedItem.Content.ToString();

ComboBoxItem selectedOperation = (ComboBoxItem)OperationComboBox.SelectedItem;

if (selectedOperation == null)

{

throw new InvalidOperationException("Не выбрана операция из выпадающего списка.");

}

string operation = selectedOperation.Content.ToString();

switch (operation)

{

case "Add":

linkedList.Add(selectedValue);

AddToLog("Добавленный Элемент: " + selectedValue);

break;

case "Remove":

bool removed = linkedList.Remove(selectedValue);

if (removed)

AddToLog("Удаленный Элемент: " + selectedValue);

else

AddToLog("Элемент не найден: " + selectedValue);

break;

case "Iterate":

AddToLog("Элементы связанного списка:");

foreach (string item in linkedList)

{

AddToLog(item);

}

break;

case "Copy To Array":

string[] array = new string[linkedList.Count];

linkedList.CopyTo(array, 0);

AddToLog("Скопированные элементы в массив:");

foreach (string item in array)

{

AddToLog(item);

}

break;

}

}

catch (Exception ex)

{

// Обработка исключения и вывод сообщения об ошибке

AddToLog("Ошибка: " + ex.Message);

}

}

/// <summary>

/// Копирование элементов списка в массив

/// </summary>

private void CopyToArray\_Click(object sender, RoutedEventArgs e)

{

string[] array = new string[linkedList.Count];

linkedList.CopyTo(array, 0);

AddToLog("Скопированные элементы в массив:");

foreach (string item in array)

{

AddToLog(item);

}

}

/// <summary>

/// Отображение сообщения в TextBox

/// </summary>

private void AddToLog(string logMessage)

{

LogTextBox.AppendText(logMessage + "\n");

}

}

**LinkedListNode.cs**

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab3oop

{

/// <summary>

/// Представляет узел в связном списке.

/// </summary>

/// <typeparam name="T">Тип значения, хранящегося в узле.</typeparam>

public class LinkedListNode<T>

{

/// <summary>

/// Получает или задает значение узла.

/// </summary>

public T Value { get; set; }

/// <summary>

/// Получает или задает ссылку на следующий узел в списке.

/// </summary>

public LinkedListNode<T> Next { get; set; }

/// <summary>

/// Инициализирует новый экземпляр класса LinkedListNode с указанным значением.

/// </summary>

/// <param name="value">Значение узла.</param>

public LinkedListNode(T value)

{

Value = value;

Next = null;

}

}

}

**UnitTest1.cs**

using System.IO;

using System.Windows.Controls;

using Libr;

namespace Tests

{

[TestClass]

public class LastNBytesStreamDecoratorTests

{

[TestMethod]

public void CheckFileStreamDecorator()

{

using (var \_fileStream = new FileStream("D:\\Labs\\4sem\\ООПиП\\2\\lab2.txt", FileMode.OpenOrCreate))

{

var \_decoratorFileStream = new LastNBytesStreamDecorator(\_fileStream, 5);

byte[] buffer = { 118, 119, 120, 121, 122 };

int bytesRead = \_decoratorFileStream.Read(buffer, 0, buffer.Length);

byte[] lastNBytes = \_decoratorFileStream.GetLastNBytes();

CollectionAssert.AreEqual(buffer, lastNBytes);

}

}

[TestMethod]

public void CheckMemoryStreamDecorator()

{

using (var memoryStream = new MemoryStream())

{

using (var fileStream = new FileStream("D:\\Labs\\4sem\\ООПиП\\2\\lab2.txt", FileMode.OpenOrCreate))

{

fileStream.CopyTo(memoryStream);

}

memoryStream.Seek(0, SeekOrigin.Begin);

var \_decoratorMemoryStream = new LastNBytesStreamDecorator(memoryStream, 5);

byte[] buffer = { 118, 119, 120, 121, 122 };

int bytesRead = \_decoratorMemoryStream.Read(buffer, 0, buffer.Length);

byte[] lastNBytes = \_decoratorMemoryStream.GetLastNBytes();

CollectionAssert.AreEqual(buffer, lastNBytes);

}

}

[TestMethod]

public void CheckBufferedStreamDecorator()

{

using (var fileStream = new FileStream("D:\\Labs\\4sem\\ООПиП\\2\\lab2.txt", FileMode.OpenOrCreate))

{

using (var bufferStream = new BufferedStream(fileStream))

{

var \_decoratorBufferedStream = new LastNBytesStreamDecorator(bufferStream, 5);

byte[] buffer = { 118, 119, 120, 121, 122 };

int bytesRead = \_decoratorBufferedStream.Read(buffer, 0, buffer.Length);

byte[] lastNBytes = \_decoratorBufferedStream.GetLastNBytes();

CollectionAssert.AreEqual(buffer, lastNBytes);

}

}

}

}

}