Лабораторная работа №8. Обобщённые коллекции

Титов Павел Сергеевич, 04-1

**[x] Описать интерфейс, включающий 7 методов. Создать класс, реализующие этот интерфейс для объектов типа string (выполнять с использованием класса LinkedList).**

**[x] Написать программу, демонстрирующую работу с экземплярами класса.**

Номера заданий (v – номер варианта, n ‐ количество заданий == 30):

**[x] Метод 1. (v + 7) % n + 1**

**[x] Метод 2. (v + 14) % n + 1**

**[x] Метод 3. (v + 21) % n + 1**

**[x] Метод 4. (v + 28) % n + 1**

**[x] Метод 5. (v + 35) % n + 1**

**[x] Метод 6. (v + 42) % n + 1**

9. void insert\_after(List, Node, int) Вставка нового элемента после заданного

16. void append(List, List) Добавление списка к началу другого (append([1, 2], [3]) -> [1, 2, 3])

23. List reversed(List) Создать развернутую копию списка.

25. void keep(List, int from, int to) Удалить все, кроме элементов с индексами [from, to)

30. int sum(List) Вернуть сумму элементов списка.

**[x] Метод 7. (номер варианта = номер по списку%9)**

- в начало списка L1 добавить все элементы списка L2;   
- из списка L1 удалить все элементы равные последнему элементу списка L2;

**Написать программу проверяющую знание английских слов. В текстовом файле хранятся английские слова и их перевод на русский язык.** Пользователю по очереди предлагается несколько английских слов из файла. Пользователь должен написать правильный перевод этих слов на русский язык. За правильный ответ начисляются баллы. Результаты проверки (имя пользователя, баллы, сколько слов было предложено, дата проверки) записываются в текстовый файл.

Код: Методов для Линкед листа

namespace L8;public interface ILinkedPro{ LinkedListNode<string> AddLast(string value); LinkedListNode<string> InsertAfter(LinkedListNode<string> node, string value); void Append(LinkedList<string> linkedList2); LinkedList<string> Reversed(); void Keep(int from, int to); int Sum();}public class LinkedPro : ILinkedPro{ private LinkedList<string> \_linkedList = new(); public LinkedListNode<string> AddLast(string value) => \_linkedList.AddLast(value); public LinkedListNode<string> InsertAfter(LinkedListNode<string> node, string value) => \_linkedList.AddAfter(node, value); public void Append(LinkedList<string> linkedList2) { foreach (var item in linkedList2) { AddLast(item); } } public LinkedList<string> Reversed() { LinkedList<string> newList = new LinkedList<string>(); LinkedList<string> sList = \_linkedList; while (\_linkedList.Any()) { newList.AddLast(sList.Last()); sList.RemoveLast(); } return newList; } public void Keep(int from, int to) { if ((from > to) || (from < 0) || (to > \_linkedList.Count - 1)) throw new Exception("Выход за границы"); LinkedList<string> newLinked = new LinkedList<string>(\_linkedList.Skip(from).Take(to - from)); for (var recentNode = newLinked.First; recentNode != null; recentNode = recentNode.Next) { \_linkedList.Remove(recentNode); } \_linkedList = newLinked; } public int Sum() { Console.*WriteLine*("Нельзя расчитать сумму для списка строк"); return \_linkedList.Count(); } public LinkedList<string> AddL2toL1Start(LinkedList<string> l1, LinkedList<string> l2) { foreach (var item in l2) { l1.AddFirst(item); } return l1; } public LinkedList<string> RemoveL1elementsByL2Element(LinkedList<string> l1, LinkedList<string> l2) { string removeItem = l2.Last(); foreach (var item in l1) { if (item == removeItem) { l1.Remove(item); } } return l1; }}public class Program{ public static void *Main***()** { Random rnd = new Random(40); LinkedPro lp = new(); var node1 = lp.AddLast("1"); var node2 = lp.AddLast("2"); var node3 = lp.AddLast("3"); IEnumerable<string> enumerable = Enumerable.*Repeat*(0, 5).Select(x => Convert.*ToString*(rnd.Next(1, 10))); lp.Append(new LinkedList<string>(enumerable)); var reversed = lp.Reversed(); lp.InsertAfter(node1, "10"); lp.Keep(1, 2); lp.Sum(); }}

Код для переводчика:

namespace L8;using System.Text.Json;public class Word{ public string En { get; set; } public string Ru { get; set; } public Word(string en, string ru) { En = en; Ru = ru; }}public class WordList{ public List<Word> Words { get; set; } public WordList(List<Word> words) { Words = words; }}public class Player{ public string? Name { get; set; } public int Score { get; set; } public int Words { get; set; } public DateTime TestDate { get; set; } public Player(string? name, int score, int words, DateTime testDate) { Name = name; Score = score; Words = words; TestDate = testDate; }}public class EnglishWords{ public static void ReMain() { Console.*Write*("Введите имя: "); var username = Console.*ReadLine*(); var player = new Player(username, 0, 0, DateTime.Now); Console.*WriteLine*($"Очков: {player.Score}"); using var fsRead = new FileStream("Words.json", FileMode.*OpenOrCreate*); WordList? words = JsonSerializer.*Deserialize*<WordList>(fsRead); foreach (var word in words?.Words!) { Console.*Write*($"Translate <{word.En}>: "); player.Words++; var userWord = Console.*ReadLine*(); if (userWord == word.Ru) player.Score++; } Console.*WriteLine*($"Очков: {player.Score}"); using var fsWrite = new FileStream("EnglishWords.json", FileMode.*OpenOrCreate*); JsonSerializer.*SerializeAsync*<Player>(fsWrite, player); }}

Пример Words.json

{ "Words": [ { "En": "Book", "Ru": "Книга" }, { "En": "Table", "Ru": "Стол" }, { "En": "Door", "Ru": "Дверь" }, { "En": "Pen", "Ru": "Ручка" }, { "En": "Phone", "Ru": "Телефон" } ]}

Пример Выгодных данных:

{ "Name": "Павел", "Score": 5, "Words": 5, "TestDate": "2022-06-09T09:21:15.1916707+05:00"}

Вывод 