Текст программы в лабораторной работе №13

Класс Program

using System;

using LibraryLaba10;

using LibraryLaba12;

namespace laba13

{

class Program

{

static void Main(string[] args)

{

MyNewCollection c1 = new MyNewCollection("Первая коллекция");

c1.NameCollection = "Первая коллекция";

MyNewCollection c2 = new MyNewCollection("Вторая коллекция");

c2.NameCollection = "Вторая коллекция";

Journal first = new Journal("первый");

Journal second = new Journal("второй");

c1.CollectionCountChanged += first.CollectionCountChanged;

c1.CollectionReferenceChanged += first.CollectionReferenceChanged;

c1.CollectionReferenceChanged += second.CollectionReferenceChanged;

c2.CollectionReferenceChanged += second.CollectionReferenceChanged;

c1.Add(new Challenge("Предмет1\_1", 10, 0));

//LibraryLaba12.MyCollection<Challenge>.PrintList(c1);

c1.Add(new Challenge("Предмет1\_2", 9, 1));

//LibraryLaba12.MyCollection<Challenge>.PrintList(c1);

c1.Add(new Challenge("Предмет1\_3", 8, 2));

//LibraryLaba12.MyCollection<Challenge>.PrintList(c1);

c1.AddDefault();

//LibraryLaba12.MyCollection<Challenge>.PrintList(c1);

c1.AddDefault();

//LibraryLaba12.MyCollection<Challenge>.PrintList(c1);

c1.AddDefault();

//LibraryLaba12.MyCollection<Challenge>.PrintList(c1);

c2.Add(new Challenge("Предмет2\_1", 1, 0));

//LibraryLaba12.MyCollection<Challenge>.PrintList(c2);

c2.Add(new Challenge("Предмет2\_2", 2, 1));

//LibraryLaba12.MyCollection<Challenge>.PrintList(c2);

c2.Add(new Challenge("Предмет2\_3", 3, 2));

//LibraryLaba12.MyCollection<Challenge>.PrintList(c2);

c1.Remove(1); //в скобке индекс начиная с 0

//LibraryLaba12.MyCollection<Challenge>.PrintList(c1);

c2.Remove(1);

//LibraryLaba12.MyCollection<Challenge>.PrintList(c2);

c1[0] = new MyCollection<Challenge>("Измененный предмет1", 0, 0);

//LibraryLaba12.MyCollection<Challenge>.PrintList(c1);

c2[1] = new MyCollection<Challenge>("Измененный предмет2", 2, 1);

//LibraryLaba12.MyCollection<Challenge>.PrintList(c2);

first.Show();

//LibraryLaba12.MyCollection<Challenge>.PrintList(c1);

second.Show();

//LibraryLaba12.MyCollection<Challenge>.PrintList(c2);

Console.ReadLine();

}

}

}

Класс MyNewCollection

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using LibraryLaba10;

using LibraryLaba12;

namespace laba13

{

public delegate void CollectionHandler(object source, CollectionHandlerEventArgs args); //объявление делегата

public class MyNewCollection : LibraryLaba12.MyCollection<Challenge>

{

public event CollectionHandler CollectionCountChanged; //событие после добаления или удаления объектов

public event CollectionHandler CollectionReferenceChanged; //событие при изменении ссылок

//свойство - имя коллекции

public string NameCollection

{

get;

set;

}

//конструктор новой коллекции с заданием ее имени

public MyNewCollection(string Name)

{

NameCollection = Name;

this.data = null;

this.next = null;

this.Count = 0;

}

//добавление в коллекцию заданного испытания

public void Add(Challenge obj)

{

if (this.Count == 0)

{

this.data = obj;

this.Count = this.Count + 1;

//вызов обработчика события при изменении коллекции

this.OnCollectionCountChaged(this, new CollectionHandlerEventArgs("добавление", this[0]));

}

else

{

this[Count - 1].next = new MyCollection<Challenge>(obj.Name, obj.Grade, obj.ID);

this.Count = this.Count + 1;

//вызов обработчика события при изменении коллекции

this.OnCollectionCountChaged(this, new CollectionHandlerEventArgs("добавление", this[Count - 1]));

}

}

//добаление в коллекцию случайного элемента

public void AddDefault()

{

Challenge c = (Challenge)new Challenge().Init();

if (this.Count == 0)

{

this.data = c;

this.Count = this.Count + 1;

//вызов обработчика события при изменении коллекции

this.OnCollectionCountChaged(this, new CollectionHandlerEventArgs("добавление", this[0]));

}

else

{

this[Count - 1].next = new MyCollection<Challenge>(c.Name, c.Grade, c.ID);

this.Count = this.Count + 1;

//вызов обработчика события при изменении коллекции

this.OnCollectionCountChaged(this, new CollectionHandlerEventArgs("добавление", this[Count - 1]));

}

}

//удаление элемента из коллекции по номеру элемента

public bool Remove(int j)

{

if (j > 0 && j < this.Count - 1)

{

//вызов обработчика события при изменении коллекции

this.OnCollectionCountChaged(this, new CollectionHandlerEventArgs("удаление", this[j]));

this[j - 1].next = this[j + 1];

this.Count--;

return true;

}

else

{

if (j == 0)

{

//вызов обработчика события при изменении коллекции

this.OnCollectionCountChaged(this, new CollectionHandlerEventArgs("удаление", this[j]));

MyCollection<Challenge> a = DeletePoint(this);

this.data = a.data;

this.next = a.next;

this.Count--;

return true;

}

else

{

if (j == this.Count - 1)

{

//вызов обработчика события при изменении коллекции

this.OnCollectionCountChaged(this, new CollectionHandlerEventArgs("удаление", this[j]));

this[Count - 2].next = null;

this.Count--;

return true;

}

else

return false;

}

}

}

public MyCollection<Challenge> this[int index]

{

get

{

MyCollection<Challenge> r = this;

if (index <= Count - 1)

{

for (int i = 0; i < index; i++)

{

r = r.next;

}

return r;

}

else

{

if (index == 0)

{

for (int i = 0; i <= index; i++)

{

r = r.next;

}

return r;

}

else return this;

}

}

set

{

if (index == 0)

{

MyCollection<Challenge> r = this;

for (int i = 0; i <= index; i++)

{

r = r.next;

}

r = value;

//вызов обработчика события при изменении ссылки

this.OnCollectionReferenceChanged(this, new CollectionHandlerEventArgs("изменение", r));

}

else

if (index <= Count - 1)

{

MyCollection<Challenge> r = this;

for (int i = 0; i < index; i++)

{

r = r.next;

}

r = value;

//вызов обработчика события при изменении ссылки

this.OnCollectionReferenceChanged(this, new CollectionHandlerEventArgs("изменение", r));

}

}

}

//обработчик события, возникающего при добавлении/удалении элементов коллекции

public void OnCollectionCountChaged(object source, CollectionHandlerEventArgs args)

{

//if (CollectionReferenceChanged != null)

//CollectionReferenceChanged(source, args);

//упрощенная запись

CollectionCountChanged?.Invoke(source, args);

}

//обработчик события, возникающего при изменении ссылок

public void OnCollectionReferenceChanged(object source, CollectionHandlerEventArgs args)

{

//if (CollectionReferenceChanged != null)

//CollectionReferenceChanged(source, args);

//упрощенная запись

CollectionReferenceChanged?.Invoke(source, args);

}

}

}

Класс CollectionHandlerEventArgs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using LibraryLaba10;

using LibraryLaba12;

namespace laba13

{

public class CollectionHandlerEventArgs : System.EventArgs

{

public string Change

{

get;

set;

}

public MyCollection<Challenge> Obj

{

get;

set;

}

public CollectionHandlerEventArgs(string u, MyCollection<Challenge> s)

{

Obj = new MyCollection<Challenge>(s.data.Name, s.data.Grade, s.data.ID);

Change = u;

}

}

}

Классы Journal и JournalEntry

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using LibraryLaba10;

using LibraryLaba12;

namespace laba13

{

public class Journal

{

JournalEntry[] arr;

string NameJournal;

public Journal(string a)

{

arr = new JournalEntry[0];

NameJournal = a;

}

public void CollectionCountChanged(object source, CollectionHandlerEventArgs args)

{

JournalEntry[] tmp = new JournalEntry[arr.Length + 1];

for (int i = 0; i <= arr.Length - 1; i++)

{

tmp[i] = arr[i];

}

tmp[tmp.Length - 1] = new JournalEntry(source, args);

arr = new JournalEntry[tmp.Length];

for (int i = 0; i <= arr.Length - 1; i++)

{

arr[i] = tmp[i];

}

}

public void CollectionReferenceChanged(object source, CollectionHandlerEventArgs args)

{

JournalEntry[] tmp = new JournalEntry[arr.Length + 1];

for (int i = 0; i <= arr.Length - 1; i++)

{

tmp[i] = arr[i];

}

tmp[tmp.Length - 1] = new JournalEntry(source, args);

arr = new JournalEntry[tmp.Length];

for (int i = 0; i <= arr.Length - 1; i++)

{

arr[i] = tmp[i];

}

}

//вывод элементов журналов

public void Show()

{

for (int i = 0; i < arr.Length; i++)

{

Console.WriteLine($"Журнал {NameJournal}, элемент: {i + 1}. Коллекция: {arr[i].NameCollection}; действие: {arr[i].Change}; Испытание: {arr[i].Data.data.Name}, Оценка: {arr[i].Data.data.Grade}\n");

}

}

//индексатор

public JournalEntry this[int index]

{

get

{

if (index < arr.Length)

return arr[index];

else

{

Console.WriteLine("Индекс вне массива!");

Console.WriteLine("Последний элемент массива: ");

return arr[arr.Length - 1];

}

}

set

{

if (index < arr.Length)

arr[index] = value;

else

Console.WriteLine("Индекс вне массива!");

}

}

}

public class JournalEntry

{

public string NameCollection { get; set; }

public string Change { get; set; }

public MyCollection<Challenge> Data { get; set; }

public JournalEntry(object source, CollectionHandlerEventArgs args)

{

NameCollection = ((MyNewCollection)source).NameCollection;

Change = args.Change;

Data = new MyCollection<Challenge>(args.Obj.data.Name, args.Obj.data.Grade, args.Obj.data.ID);

}

}

}

Код для UnitTests

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using System.Collections.Generic;

using System.Collections;

using laba13;

using LibraryLaba10;

using LibraryLaba12;

namespace testsfor13

{

[TestClass]

public class UnitTest1

{

[TestMethod]

public void TestMethod1()

{

JournalEntry j = new JournalEntry("gfg", "del", "efr");

Assert.AreEqual(j.Name, "gfg");

}

[TestMethod]

public void TestMethod2()

{

JournalEntry j = new JournalEntry("gfg", "del", "efr");

Assert.AreEqual(j.Change, "del");

}

[TestMethod]

public void TestMethod3()

{

JournalEntry j = new JournalEntry("gfg", "del", "efr");

Assert.AreEqual(j.Obj, "efr");

}

[TestMethod]

public void TestMethod4()

{

Journal j = new Journal();

Assert.AreEqual(0, j.journal.Count);

}

[TestMethod]

public void TestMethod5()

{

Journal j = new Journal();

j.Add(new JournalEntry("","",""));

Assert.AreEqual(1, j.journal.Count);

}

[TestMethod]

public void TestMethod6()

{

Journal j = new Journal();

j.Add(new JournalEntry("", "", ""));

j.Add(new JournalEntry("", "", ""));

j.Add(new JournalEntry("", "", ""));

Assert.AreEqual(3, j.journal.Count);

}

[TestMethod]

public void TestMethod7()

{

Journal j = new Journal();

CollectionHandlerEventArgs e = new CollectionHandlerEventArgs("", "", new GenPoint<Organisation>(new Organisation("jj",1,1,1)));

j.CollectionCountChanged(new MyNewCollection<Organisation>(), e);

Assert.AreEqual(1, j.journal.Count);

}

[TestMethod]

public void TestMethod8()

{

Journal j = new Journal();

CollectionHandlerEventArgs e = new CollectionHandlerEventArgs("", "", new GenPoint<Organisation>(new Organisation("jj", 1, 1, 1)));

j.CollectionCountChanged(new MyNewCollection<Organisation>(), e);

j.CollectionCountChanged(new MyNewCollection<Organisation>(), e);

Assert.AreEqual(2, j.journal.Count);

}

[TestMethod]

public void TestMethod9()

{

Journal j = new Journal();

CollectionHandlerEventArgs e = new CollectionHandlerEventArgs("", "", new GenPoint<Organisation>(new Organisation("jj", 1, 1, 1)));

j.CollectionReferenceChanged(new MyNewCollection<Organisation>(), e);

Assert.AreEqual(1, j.journal.Count);

}

[TestMethod]

public void TestMethod10()

{

Journal j = new Journal();

CollectionHandlerEventArgs e = new CollectionHandlerEventArgs("", "", new GenPoint<Organisation>(new Organisation("jj", 1, 1, 1)));

j.CollectionReferenceChanged(new MyNewCollection<Organisation>(), e);

j.CollectionReferenceChanged(new MyNewCollection<Organisation>(), e);

Assert.AreEqual(2, j.journal.Count);

}

[TestMethod]

public void TestMethod11()

{

MyNewCollection<Organisation> m = new MyNewCollection<Organisation>(5, 5, "");

Journal j = new Journal();

m.CollectionReferenceChanged += new CollectionHandler(j.CollectionReferenceChanged);

m[1] = new GenPoint<Organisation>();

Assert.AreEqual(1, j.journal.Count);

}

[TestMethod]

public void TestMethod12()

{

MyNewCollection<Organisation> m = new MyNewCollection<Organisation>(5, 5, "");

Journal j = new Journal();

m.CollectionCountChanged += new CollectionHandler(j.CollectionCountChanged);

m.Remove(0);

Assert.AreEqual(1, j.journal.Count);

}

[TestMethod]

public void TestMetho1()

{

CollectionHandlerEventArgs m = new CollectionHandlerEventArgs("", "", new GenPoint<Organisation>());

Assert.AreEqual("", m.ChangeCollection);

}

[TestMethod]

public void TestMetho2()

{

CollectionHandlerEventArgs m = new CollectionHandlerEventArgs("", "", new GenPoint<Organisation>());

Assert.AreEqual("", m.NameCollection);

}

[TestMethod]

public void TestMetho3()

{

CollectionHandlerEventArgs m = new CollectionHandlerEventArgs("", "", new GenPoint<Organisation>());

GenPoint<Organisation> org = new GenPoint<Organisation>();

Assert.AreEqual(org.Next, m.Obj.Next);

}

[TestMethod]

public void TestMeth4()

{

MyNewCollection<Organisation> mc = new MyNewCollection<Organisation>(5,5,"One");

mc.AddDefault();

Assert.AreEqual(mc.Count, 6);

}

[TestMethod]

public void TestMeth5()

{

MyNewCollection<Organisation> mc = new MyNewCollection<Organisation>(5, 5, "One");

int count = 0;

foreach (GenPoint<Organisation> e in mc)

count++;

Assert.AreEqual(count, 5);

}

[TestMethod]

public void TestMeth6()

{

MyNewCollection<Organisation> mc = new MyNewCollection<Organisation>(5, 5, "One");

mc[1] = new GenPoint<Organisation>(new Organisation("",1,1,1));

Assert.AreEqual(1, mc[1].Data.idNumber.id);

}

[TestMethod]

public void TestMeth7()

{

MyNewCollection<Organisation> mc = new MyNewCollection<Organisation>(5, 5, "One");

mc.Add\_Some(1, 2);

Assert.AreEqual(6,mc.Count);

}

[TestMethod]

public void TestMeth8()

{

MyNewCollection<Organisation> mc = new MyNewCollection<Organisation>(5, 5, "One");

mc.Del\_Some(1, 2);

Assert.AreEqual(4, mc.Count);

}

}

}