Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

Делегаты, лямбда-выражения, события

тема

Вариант 11 (1)

|  |  |  |
| --- | --- | --- |
| Преподаватель |  | А. А. Чикизов |
|  | подпись, дата | инициалы, фамилия |
| Студент КИ18-16б 031831229 |  | В. А. Прекель |
| номер группы, зачетной книжки | подпись, дата | инициалы, фамилия |

Красноярск 2020

# 1 Задание

Информация для всех вариантов

В лабораторной работе требуется определить класс-коллекцию XCollection, содержащий в себе элементы типа X. Для хранения элементов использовать стандартные коллекции.

XCollection изменяется при добавлении, удалении элементов X (событие count change) или изменении одного из элементов (ссылки) в коллекции (событие reference change). В этом случае в соответствующих методах или свойствах класса XCollection вызываются(invoke) специально объявленные делегаты.

Для хранения и вызова в XCollection методов «подписанных» классов (в нашем случаем это будут методы класса Journal) необходимо использовать закрытые делегаты и открытые события. Подписку и отписку методов проводить только через события и их служебные методы (add, remove), никакого прямого доступа к закрытым делегатам быть не должно.

Для сортировки элементов коллекции необходимо использовать метод Sort (лямбда-выражение с полем сортировки). Использование лямбда-выражений для других целей приветствуется.

Определить класс хранения информации о событии XListHandlerEventArgs, производный от класса EventArgs включающий в себя

• открытое автосвойство типа string с названием коллекции, в которой произошло событие;

• открытое автосвойство типа string с информацией о типе изменений в коллекции; (типы изменений назвать самостоятельно)

• открытое автосвойство типа X для хранения ссылки на объект, с которым связаны изменения;

• конструкторы для инициализации класса;

• перегруженную версию метода string ToString() для формирования строки с информацией обо всех полях класса.

Объявить новый тип данных - делегат XListHandler с сигнатурой:

delegate void XListHandler

(object source, XListHandlerEventArgs args);

Определить класс XCollection, который содержит в себе

• закрытое поле типа List<X>;

• автосвойство типа string с названием коллекции;

• метод bool Add (int j, X record) для добавления элемента record в позицию j списка List<X>; если в списке нет элемента с номером j, метод ничего не добавляет и возвращает значение false;

• метод bool Remove (int j) для удаления элемента с номером j из списка List<X>; если в списке нет элемента с номером j, метод возвращает значение false;

• индексатор this типа X (с методами get и set) с целочисленным индексом для доступа к элементу списка List<X> с заданным номером.

Названия событий в XCollection (тип делегата XListHandler)

• onXCountChanged - происходит при добавлении нового элемента в коллекцию или при удалении элемента из коллекции; через объект XListHandlerEventArgs cотбытие передает имя коллекции, строку с информацией о том, что в коллекцию был добавлен новый элемент или из нее был удален элемент, ссылку на добавленный или удаленный элемент X;

• onXReferenceChanged - происходит, когда одной из ссылок, входящих в коллекцию, присваивается новое значение; через объект XListHandlerEventArgs событие передает имя коллекции, строку с информацией о том, что был заменен элемент в коллекции, и ссылку на новый элемент X.

Событие XCountChanged вызывают следующие методы класса XCollection

• AddDefaults(); - заполнение коллекции произвольными заранее заданными данными

• AddX (params X[] ) ; - добавление в коллекцию указанных элементов

• Remove (int j) – удаление j элемента

\*Событие XReferenceChanged вызывается в методе set индексатора, определенного в классе XCollection.

Определить класс Journal, который можно использовать для накопления информации об изменениях в коллекциях типа XCollection. Класс

Journal хранит информацию в списке объектов типа JournalEntry. Каждый элемент списка содержит информацию об отдельном изменении, которое произошло в коллекции.

Класс JournalEntry содержит

• открытое автосвойство типа string с названием коллекции, в которой произошло событие;

• открытое автосвойство типа date с информацией о времени, когда произошло событие;

• открытое автосвойство типа string с информацией о типе изменений в коллекции;

• открытое автосвойство типа string c данными объекта X, с которым связаны изменения в коллекции;

• конструктор для инициализации полей класса;

• перегруженную версию метода string ToString().

Класс Journal содержит

• закрытое поле типа List<JournalEntry>;

• реализует обработчики XCountChanged и XReferenceChanged (для подписки на события в XCollection) которые при вызове добавляют новый элемент JournalEntry в список List<JournalEntry>; для инициализации JournalEntry используется информация из объекта XListHandlerEventArgs, который передается вместе с событием;

• перегруженную версию метода string ToString() для формирования строки с информацией обо всех элементах списка List<JournalEntry>.

В методе Main()

1. Создать две коллекции XCollection.

2. Создать объекта типа Journal и подписать его на события onXCountChanged и onXReferenceChanged всех коллекций XCollection

3. Внести изменения в коллекциях XCollection

• добавить элементы в коллекции;

• удалить некоторые элементы из коллекций;

• присвоить некоторым элементам коллекций новые значения;

• провести сортировку коллекции по разным полям

• провести сортировку журнала по типам операции

4. Вывод программы должен показать работу всех используемых технологий

Варианты заданий

Имя класса и поля класса Х для XCollection взять из лабораторной работы №3

Например для варианта 1 это будет класс Student с полями

□ фамилия и инициалы;

□ номер группы;

□ успеваемость (массив из пяти элементов).

# 2 Исходный код основного алгоритма

Листинг 1 – CSharpLabs.Lab04\CSharpLabs.Lab04.Console\Program.cs

п»їusing System;  
using System.Text;  
  
using CSharpLabs.Lab04.Core;  
  
Console.OutputEncoding = Encoding.UTF8;  
  
StudentCollection collection1 = new("РљРѕР»Р»РµРєС†РёСЏ 1");  
StudentCollection collection2 = new("РљРѕР»Р»РµРєС†РёСЏ 2");  
  
Journal journal = new();  
  
collection1.OnStudentCountChanged += (\_, eventArgs) => journal.StudentCountChanged(eventArgs);  
collection1.OnStudentReferenceChanged += (\_, eventArgs) => journal.StudentReferenceChanged(eventArgs);  
collection2.OnStudentCountChanged += (\_, eventArgs) => journal.StudentCountChanged(eventArgs);  
collection2.OnStudentReferenceChanged += (\_, eventArgs) => journal.StudentReferenceChanged(eventArgs);  
  
  
collection1.AddDefaults();  
collection2.AddDefaults();  
  
collection1.Remove(5);  
collection2.Remove(4);  
collection1.Remove(3);  
collection2.Remove(2);  
collection1.Remove(1);  
  
collection1[0] = collection1[0] with { Name = "РќРѕРІРѕРµ РёРјСЏ" };  
  
collection1.SortViaOrderBy(student => student.Name);  
collection2.SortViaOrderBy(student => student.Group);  
  
journal.SortByType();  
  
Console.WriteLine(journal);

Листинг 2 – CSharpLabs.Lab04\CSharpLabs.Lab04.Core\Journal.cs

using System;  
using System.Collections.Generic;  
  
namespace CSharpLabs.Lab04.Core  
{  
 public class Journal  
 {  
 private readonly List<JournalEntry> List = new();  
  
 public void StudentCountChanged(StudentCollection.StudentListHandlerEventArgs eventArgs)  
 {  
 List.Add(new JournalEntry(eventArgs.CollectionName, DateTimeOffset.Now, eventArgs.Type,  
 eventArgs.Element.ToString()));  
 }  
  
 public void StudentReferenceChanged(StudentCollection.StudentListHandlerEventArgs eventArgs)  
 {  
 List.Add(new JournalEntry(eventArgs.CollectionName, DateTimeOffset.Now, eventArgs.Type,  
 eventArgs.Element.ToString()));  
 }  
  
 public override string ToString() => String.Join(";\n", List);  
  
 public void SortByType()  
 {  
 List.Sort((e1, e2) => String.Compare(e1.Type, e2.Type, StringComparison.Ordinal));  
 }  
 }  
}

Листинг 3 – CSharpLabs.Lab04\CSharpLabs.Lab04.Core\JournalEntry.cs

using System;  
  
namespace CSharpLabs.Lab04.Core  
{  
 public record JournalEntry(string CollectionName, DateTimeOffset Time, string Type, string Data)  
 {  
 public override string ToString() =>  
 $"CollectionName: {CollectionName}, Р’СЂРµРјСЏ: {Time}, РўРёРї: {Type}, Data: {Data}";  
 }  
}

Листинг 4 – CSharpLabs.Lab04\CSharpLabs.Lab04.Core\Student.cs

п»їusing System;  
using System.Collections.Generic;  
  
namespace CSharpLabs.Lab04.Core  
{  
 public record Student(string Name, string Group, IEnumerable<int> Marks)  
 {  
 public override string ToString() =>  
 $"Student: Name: {Name}; Group: {Group}; Marks: {String.Join(", ", Marks)}";  
 }  
}

Листинг 5 – CSharpLabs.Lab04/CSharpLabs.Lab04.Core/StudentCollection.cs

using System;  
using System.Collections;  
using System.Collections.Generic;  
using System.Linq;  
  
namespace CSharpLabs.Lab04.Core  
{  
 public class StudentCollection : IEnumerable<Student>  
 {  
 public delegate void StudentListHandler(object source, StudentListHandlerEventArgs args);  
  
 private List<Student> List = new();  
  
 private StudentListHandler onStudentCountChanged;  
  
 private StudentListHandler onStudentReferenceChanged;  
  
 public StudentCollection(string collectionName) => CollectionName = collectionName;  
  
 public string CollectionName { get; }  
  
 public Student this[int j]  
 {  
 get => List[j];  
 set  
 {  
 onStudentReferenceChanged?.Invoke(this,  
 new StudentListHandlerEventArgs(CollectionName, "Замена", value));  
 List[j] = value;  
 }  
 }  
  
 public IEnumerator<Student> GetEnumerator() => throw new NotImplementedException();  
  
 IEnumerator IEnumerable.GetEnumerator() => GetEnumerator();  
  
 public bool Add(int j, Student record)  
 {  
 if (List.Count <= j)  
 {  
 return false;  
 }  
  
 List.Insert(j, record);  
 return true;  
 }  
  
 public bool Remove(int j)  
 {  
 if (List.Count <= j)  
 {  
 return false;  
 }  
  
 onStudentCountChanged?.Invoke(this, new StudentListHandlerEventArgs(CollectionName, "Удаление", List[j]));  
 List.RemoveAt(j);  
 return true;  
 }  
  
 public void AddDefaults()  
 {  
 var s1 = new Student("Тимофеев М.А.", "КИ18-16б", new[] {1, 2, 3, 4, 5});  
 onStudentReferenceChanged?.Invoke(this, new StudentListHandlerEventArgs(CollectionName, "Добавление", s1));  
 List.Add(s1);  
 var s2 = new Student("Тимофеев М.П.", "КИ18-17б", new[] {2, 2, 3, 4, 5});  
 onStudentReferenceChanged?.Invoke(this, new StudentListHandlerEventArgs(CollectionName, "Добавление", s2));  
 List.Add(s2);  
 var s3 = new Student("Максимов М.П.", "КИ18-15б", new[] {3, 2, 3, 4, 5});  
 onStudentReferenceChanged?.Invoke(this, new StudentListHandlerEventArgs(CollectionName, "Добавление", s3));  
 List.Add(s3);  
 var s4 = new Student("Максимов Т.П.", "КИ18-165б", new[] {4, 2, 3, 4, 5});  
 onStudentReferenceChanged?.Invoke(this, new StudentListHandlerEventArgs(CollectionName, "Добавление", s4));  
 List.Add(s4);  
 var s5 = new Student("Максимов Т.А.", "КИ18-165б", new[] {5, 2, 3, 4, 5});  
 onStudentReferenceChanged?.Invoke(this, new StudentListHandlerEventArgs(CollectionName, "Добавление", s5));  
 List.Add(s5);  
 }  
  
 public void AddStudent(params Student[] students)  
 {  
 foreach (var i in students)  
 {  
 onStudentReferenceChanged?.Invoke(this,  
 new StudentListHandlerEventArgs(CollectionName, "Добавление", i));  
 List.Add(i);  
 }  
 }  
  
 public void SortViaOrderBy<T>(Func<Student, T> f)  
 {  
 List = List.OrderBy(f).ToList();  
 }  
  
 public event StudentListHandler OnStudentCountChanged  
 {  
 add  
 {  
 onStudentCountChanged += value;  
 Console.WriteLine($"{value.Method.Name} добавлен");  
 }  
 remove  
 {  
 onStudentCountChanged -= value;  
 Console.WriteLine($"{value.Method.Name} удален");  
 }  
 }  
  
 public event StudentListHandler OnStudentReferenceChanged  
 {  
 add  
 {  
 onStudentReferenceChanged += value;  
 Console.WriteLine($"{value.Method.Name} добавлен");  
 }  
 remove  
 {  
 onStudentReferenceChanged -= value;  
 Console.WriteLine($"{value.Method.Name} удален");  
 }  
 }  
  
 public record StudentListHandlerEventArgs(string CollectionName, string Type, Student Element);  
 }  
}