Задание:

**Вариант 3**

Определить новую версию класса **ResearchTeamCollection** из лабораторной работы 3, которая с помощью событий сообщает об изменениях в коллекции.

Для событий определить делегат **TeamListHandler** с сигнатурой:

void TeamListHandler

(object source, TeamListHandlerEventArgs args);

Класс **TeamListHandlerEventArgs**, производный от класса System.EventArgs, содержит

* открытое автореализуемое свойство типа string с названием коллекции, в которой произошло событие;
* открытое автореализуемое свойство типа string с информацией о типе изменений в коллекции;
* открытое автореализуемое свойство типа int с номером элемента, который был добавлен или заменен;
* конструкторы для инициализации класса;
* перегруженную версию метода string ToString() для формирования строки с информацией обо всех полях класса.

В новую версию класса **ResearchTeamCollection** добавить

* открытое автореализуемое свойство типа string с названием коллекции;
* метод void InsertAt (int j, ResearchTeam rt), который вставляет элемент rt в список List<ResearchTeam> перед элементом с номером j; если в списке нет элемента с номером j, метод добавляет элемент в конец списка;
* индексатор типа ResearchTeam (с методами get и set) с целочисленным индексом для доступа к элементу списка List<ResearchTeam> с заданным номером.

В новую версию класса **ResearchTeamCollection** добавить два события типа TeamListHandler

* **ResearchTeamAdded**, которое происходит при добавлении элемента в конец списка List<ResearchTeam>; cобытие передает через объект TeamListHandlerEventArgs имя коллекции, строку с информацией о том, что в коллекцию был добавлен элемент, и номер добавленного элемента в списке List<ResearchTeam>;
* **ResearchTeamInserted**, которое происходит, когда новый элемент вставляется перед одним из элементов списка List<ResearchTeam>; событие передает через объект TeamListHandlerEventArgs имя коллекции, строку с информацией о том, что в коллекцию был вставлен элемент, и номер нового элемента.

Событие ResearchTeamAdded бросают методы класса ResearchTeamCollection

* AddDefaults();
* AddResearchTeams ( params ResearchTeam []);
* InsertAt (int j, ResearchTeam rt), если элемента с номером j нет в списке.

Событие ResearchTeamInserted бросает метод InsertAt (int j, ResearchTeam rt), если элемент с номером j есть в списке List<ResearchTeam>.

Определить класс **TeamsJournal** для накопления информации об изменениях в коллекциях типа ResearchTeamCollection. В классе TeamsJournal информация хранится в списке из элементов типа **TeamsJournalEntry,** каждый элемент списка содержит информацию об отдельном изменении в коллекции ResearchTeamCollection.

Класс **TeamsJournalEntry** содержит

* открытое автореализуемое свойство типа string с названием коллекции, в которой произошло событие;
* открытое автореализуемое свойство типа string с информацией о том, какое событие произошло в коллекции;
* номер нового элемента;
* конструктор для инициализации полей класса;
* перегруженную версию метода string ToString().

Класс **TeamsJournal** содержит

* закрытое поле List<TeamsJournalEntry> для списка изменений;
* обработчик событий ResearchTeamAdded и ResearchTeamInserted. Обработчик использует информацию, которая передается ему через объект TeamListHandlerEventArgs, создает элемент TeamsJournalEntry и добавляет его в список List<TeamsJournalEntry>.
* перегруженную версию метода string ToString() для формирования строки с информацией обо всех элементах списка List<TeamsJournalEntry>.

В методе **Main()**

1. Создать две коллекции ResearchTeamCollection.
2. Создать два объекта типа TeamsJournal, один объект TeamsJournal подписать на события ResearchTeamAdded и ResearchTeamInserted из первой коллекции ResearchTeamCollection, другой объект TeamsJournal подписать на события ResearchTeamInserted из обеих коллекций ResearchTeamCollection.
3. Внести изменения в коллекции ResearchTeamCollection

* добавить элементы в коллекции;
* с помощью метода InsertAt (int j, ResearchTeam rt) перед элементом с номером j, который есть в коллекции, вставить новый элемент;
* вызвать метод InsertAt (int j, ResearchTeam rt) с номером j, которого нет в коллекции.

1. Вывести данные обоих объектов TeamsJournal.

Код:

**ResearchTeamCollection**

class ResearchTeamCollection : IEnumerable

{

private List<ResearchTeam> SomeResearchTeams = new List<ResearchTeam>();// акрытое поле типа System.Collections.Generic.List<ResearchTeam>;

//открытое автореализуемое свойство типа string с названием коллекции

public delegate void TeamListHandler(object source, TeamListHandlerEventArgs args);

public string CollectionName { get; set; }

//метод void InsertAt (int j, ResearchTeam rt), который вставляет элемент rt в список List<ResearchTeam> перед элементом с номером j; если в списке нет элемента с номером j, метод добавляет элемент в конец списка

public void InsertAt(int j, ResearchTeam rt)

{

if (SomeResearchTeams.Count < j)

{

SomeResearchTeams[SomeResearchTeams.Count - 1] = rt;

ResearchTeamAdded(SomeResearchTeams[SomeResearchTeams.Count - 1], new TeamListHandlerEventArgs("Коллекция в InsertAt", "Last element added", SomeResearchTeams.Count - 1));

}

for (int i = 0; i < SomeResearchTeams.Count; i++)

{

if (i - 1 == j)

{

SomeResearchTeams.Insert(i, rt);

ResearchTeamInserted(SomeResearchTeams[i], new TeamListHandlerEventArgs("Коллекция в InsertAt", "Element was added", j));

}

}

}

//индексатор типа ResearchTeam (с методами get и set) с целочисленным индексом для доступа к элементу списка List<ResearchTeam> с заданным номером

public ResearchTeam this[int index]

{

get { return SomeResearchTeams[index]; }

set { SomeResearchTeams[index] = value; }

}

public event TeamListHandler ResearchTeamAdded;

public event TeamListHandler ResearchTeamInserted;

public void AddDefaults()

{

for (int i = 0; i < 3; i++)

{

SomeResearchTeams.Add(new ResearchTeam());

}

ResearchTeamAdded(SomeResearchTeams[SomeResearchTeams.Count - 1], new TeamListHandlerEventArgs("Коллекция в AddDefaults", "Last element added", 2));

}

public void AddResearchTeams(params ResearchTeam[] AdditionalTeams)

{

SomeResearchTeams.AddRange(AdditionalTeams);

ResearchTeamAdded(SomeResearchTeams[SomeResearchTeams.Count - 1], new TeamListHandlerEventArgs("Коллекция в AddResearchTeams", "Last element added", SomeResearchTeams.Count - 1));

}

public int minNum

{

get

{

if (SomeResearchTeams.Count == 0) return -1;

return SomeResearchTeams.Min((e) => e.RegistrationNumber);

}

}

//перегруженную версию виртуального метода string ToString() для формирования строки c информацией обо всех элементах списка List<ResearchTeam>, которая содержит значения всех полей, список

public override string ToString()

{

string ResTeamString = "";

foreach (ResearchTeam team in SomeResearchTeams)

{

ResTeamString += team.ToString();

}

return ResTeamString;

}

//по номеру регистрации с использованием интерфейса IComparable, реализованного в классе Team;

public void ToSortByRegistrNumber()

{

SomeResearchTeams.Sort((x, y) => x.RegistrationNumber.CompareTo(y.RegistrationNumber));

}

//по названию темы исследований с использованием интерфейса IComparer<ResearchTeam>, реализованного в классе ResearchTeam;

public void SortByString()

{

SomeResearchTeams.Sort();

}

//по числу публикаций с использованием интерфейса IComparer<ResearchTeam>, реализованного во вспомогательном классе.

public void SortByPublications()

{

ResearchTeamComparer comp = new ResearchTeamComparer();

SomeResearchTeams.Sort(comp);

}

// свойство типа int (только с методом get), возвращающее минимальное значение номера регистрации для элементов списка List<ResearchTeam>;

public int MinRegNumber

{

get

{

if (SomeResearchTeams.Count == 0)

{

return 0;

}

return SomeResearchTeams.Min(teams => teams.RegistrationNumber);

}

}

//свойство типа IEnumerable<ResearchTeam> (только с методом get), возвращающее подмножество элементов списка List<ResearchTeam> с продолжительностью исследований TimeFrame.TwoYears; для формирования подмножества использовать метод Where класса System.Linq.Enumerable;

public IEnumerable<ResearchTeam> TwoYearsLong

{

get

{

IEnumerable<ResearchTeam> TwoTearsL = SomeResearchTeams.Where(time => time.ResearchDuration == TimeFrame.TwoYears);

return TwoTearsL;

}

}

//• метод List<ResearchTeam> NGroup(int value), который возвращает список, в который входят элементы ResearchTeam из списка List<ResearchTeam> с заданным числом участников исследования; для формирования списка использовать методы Group и ToList класса System.Linq.Enumerable.

public IEnumerator GetEnumerator()

{

for (int i = 0; i < SomeResearchTeams.Count; i++)

{

yield return SomeResearchTeams[i];

}

}

}

}

**TeamListHandlerEventArgs**

class TeamListHandlerEventArgs : EventArgs

{

//открытое автореализуемое свойство типа string с названием коллекции, в которой произошло событие

public string EventedCollectionName { get; set; }

//открытое автореализуемое свойство типа string с информацией о типе изменений в коллекции

public string EventInfo { get; set; }

//открытое автореализуемое свойство типа int с номером элемента, который был добавлен или заменен

public int ElementNumber { get; set; }

//• конструкторы для инициализации класса

public TeamListHandlerEventArgs(string eventedCollectionName, string eventInfo, int elementNumber)

{

EventedCollectionName = eventedCollectionName;

EventInfo = eventInfo;

ElementNumber = elementNumber;

}

//• конструкторы для инициализации класса

public TeamListHandlerEventArgs() : this("SomCollection", "Some changes", 0) { }

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

public override string ToString()

{

return $"Name of collection that throw event: {EventedCollectionName}\nEvent Info:\n\t{EventInfo}\nNumber of collection element, where event was thrown: {ElementNumber}";

}

}

**TeamsJournal**

class TeamsJournal

{

//закрытое поле List<TeamsJournalEntry> для списка изменений

private List<TeamsJournalEntry> ListOfChanges = new List<TeamsJournalEntry>();

public void TeamEventHandler(object source, TeamListHandlerEventArgs args)

{

ListOfChanges.Add(new TeamsJournalEntry(args.EventedCollectionName, args.EventInfo, args.ElementNumber));

}

//перегруженную версию метода string ToString() для формирования строки с информацией обо всех элементах списка List<TeamsJournalEntry>

public override string ToString()

{

string str = "";

foreach (TeamsJournalEntry en in ListOfChanges)

{

str += en.ToString() + "\n";

}

return str;

}

}

}

**TeamsJournalEntry**

class TeamsJournalEntry

{

//открытое автореализуемое свойство типа string с названием коллекции, в которой произошло событие

public string EventedCollectionName { get; set; }

//открытое автореализуемое свойство типа string с информацией о том, какое событие произошло в коллекции

public string EventInfo { get; set; }

//номер нового элемента

public int ElementNumber { get; set; }

//конструктор для инициализации полей класса

public TeamsJournalEntry(string Name, string Ev, int numOfEl)

{

EventedCollectionName = Name;

EventInfo = Ev;

ElementNumber = numOfEl;

}

//перегруженную версию метода string ToString()

public override string ToString()

{

return string.Format("Название коллекции: {0} \n Событие: {1} \n Кол-во элементов {2} \n", this.EventedCollectionName, this.EventInfo, this.ElementNumber);

}

}

**Main**

static void Main(string[] args)

{

//Console.BackgroundColor = ConsoleColor.White;

//Console.ForegroundColor = ConsoleColor.Green;

//1. Создать две коллекции ResearchTeamCollection

ResearchTeamCollection collection1 = new ResearchTeamCollection();

ResearchTeamCollection collection2 = new ResearchTeamCollection();

//2. Создать два объекта типа TeamsJournal, один объект TeamsJournal

//подписать на события ResearchTeamAdded и ResearchTeamInserted из

//первой коллекции ResearchTeamCollection, другой объект TeamsJournal

//подписать на события ResearchTeamInserted из обеих коллекций ResearchTeamCollection

TeamsJournal journal1 = new TeamsJournal();

TeamsJournal journal2 = new TeamsJournal();

collection1.ResearchTeamAdded += journal1.TeamEventHandler;

collection1.ResearchTeamInserted += journal1.TeamEventHandler;

collection1.ResearchTeamAdded += journal2.TeamEventHandler;

collection1.ResearchTeamInserted += journal2.TeamEventHandler;

collection2.ResearchTeamAdded += journal2.TeamEventHandler;

collection2.ResearchTeamInserted += journal2.TeamEventHandler;

collection1.AddDefaults();

collection1.AddResearchTeams(new ResearchTeam("Chaos", "IRE", 1, TimeFrame.Year));

collection2.AddDefaults();

//с помощью метода InsertAt (int j, ResearchTeam rt) перед элементом с номером j, который есть в коллекции, вставить новый элемент

collection2.InsertAt(1, new ResearchTeam("Another", "IRE", 1, TimeFrame.Year));

//вызвать метод InsertAt (int j, ResearchTeam rt) с номером j, которого нет в коллекции

collection2.InsertAt(6, new ResearchTeam("Another", "IRE", 1, TimeFrame.Year));

Console.WriteLine(journal1.ToString());

Console.WriteLine();

Console.WriteLine(journal2.ToString());

Console.ReadLine();

Результат:

