Лабораторная работа 1. Вариант 2

Наследование. Интерфейсы. Итераторы и блоки итераторов. Универсальные типы

Темы:

* Переопределение виртуальных методов
* Интерфейсы.
* Интерфейс **System.Collections.IEnumerable** и **System.Collections.Generic.IEnumerable<T>**
* Оператор **foreach**. Итераторы и блоки итераторов.
* Универсальный тип **System.Collections.Generic.List<T>**.

Требования к программе

В программе определить

класс **Person**;

класс **Paper**;

класс **Researcher**, производный от класса **Person**;

класс **Programmer**, производный от класса **Person**;

класс **Department.**

В программе определить перечисление (**enum**) **Subject**,имена элементов которого – названия тем исследований.

Определить интерфейс

**interface IDeepCopy**

**{ object DeepCopy(); }**

Все перечисленые выше классы реализуют интерфейс **IDeepCopy.**

Класс Person

Класс **Person** реализует интерфейс **IDeepCopy** и имеет

* закрытое поле типа массив с двумя элементами типа **string**, в котором хранятся имя и фамилия;
* закрытое поле типа **System.DateTime** для даты рождения.

В классе **Person** определить конструкторы:

* конструктор c параметрами типа **string (имя)**, **string(фамилия)**, **int (число), int (месяц), int (год рождения)** для инициализации всех полей класса; в объявлении метода всем параметрам присвоить значения по умолчанию;
* конструктор c параметрами типа **string (имя)**, **string(фамилия)**, **DateTime (дата рождения)** для инициализации всех полей класса; в объявлении метода всем параметрам присвоить значения по умолчанию.

В классе **Person** определить открытые свойства c методами **get** и **set**:

* свойство типа **string** для доступа к имени;
* свойство типа **string** для доступа к фамилии;
* свойство типа **DateTime** для доступа к полю с датой рождения.

В классе **Person** определить

* перегруженную(**override**) версию виртуального метода **string ToString()** для формирования строки с данными класса;
* виртуальный метод **string ToShortString()**, который возвращает строку, содержащую только фамилию и имя.

Класс Paper

Класс **Paper** реализует интерфейс **IDeepCopy** и имеет два открытых автореализуемых свойства (c методами **get** и **set)**:

* свойство типа **string** для названия публикации;
* свойство типа **int** для числа авторов публикации;
* свойство типа **Subject** с темой исследований, с которой связана публикация.

В классе **Paper** определить

* конструктор c параметрами типа **string, int, Subject** для инициализации данных класса; в объявлении метода всем параметрам присвоить значения по умолчанию;
* перегруженную(**override**) версию виртуального метода **string ToString()** для формирования строки со значениями всех данных класса.

Класс Researcher

Класс **Researcher** является производным от класса **Person,** реализует интерфейс **IDeepCopy** иимеет

* открытое автореализуемое свойство (с методами **get** и **set**) типа **double** для стажа исследователя;
* закрытое поле типа **List<Paper>** со списом статей исследователя;
* конструктор **Researcher (Person p, double d, params Paper[] papers)**;
* открытый метод **AddPapers(params Paper[] papers)** для добавления объекта **Paper** в список статей **List<Paper>**;
* открытый метод **RemovePaperAt(int index)** для удаления элемента с индексом **index** из списка статей **List<Paper>**;
* перегруженную(**override**) версию виртуального метода **string ToString()** для формирования строки с полными данными класса, в том числе содержащую полные данные для каждого элемента списка статей **List<Paper>**;
* перегруженную(**override**) версию виртуального метода **string ToShortString()**, который возвращает строку, содержащую только фамилию, имя, информацию о стаже и числе статей исследователя.

Класс Programmer

Класс **Programmer** является производным от класса **Person** реализует интерфейс **IDeepCopy** иимеет

* закрытое поле типа **List<Subject>** для списка тем исследований, с которыми связана работа программиста;
* конструктор **Programmer (Person p, params Subject[] subjects)**;
* открытый метод **AddSubjects (params Subject[] subjects)** для добавления темы исследованийв список **List<Subject>**;
* открытый метод **RemoveSubjectAt(int index)** для удаления элемента с индексом **index** из списка **List<Subject>**;
* перегруженную(**override**) версию виртуального метода **string ToString()** для формирования строки с полными данными класса, в том числе содержащую список тем исследований **List<Subject>**;
* перегруженную(**override**) версию виртуального метода **string ToShortString()**, который возвращает строку, содержащую только фамилию, имя и информацию о числе тем исследований в списке **List<Subject>**.

Класс Department

Класс **Department** реализует интерфейсы **IDeepCopy** и **IEnumerable<Person>** и имеет

* открытое автореализуемое свойство (с методами **get** и **set**) типа **string** c названием организации;
* открытое автореализуемое свойство (с методами **get** и **set**) типа **DateTime** для года создания организации;
* закрытое поле типа **List<Person>** для списка сотрудников организации;
* открытый метод **AddPersons (params Person[] ps)** для добавления в список **List<Person>** объектов типа **Person** или производных от него;
* открытый метод **AddDefaults()**, в котором в список **List<Person>** добавляются несколько элементов типов **Person**, **Researcher** и **Programmer**; данные добавляемых в список объектов **Person**, **Researcher** и **Programmer** инициализируются в программе(не вводятся с клавиатуры при выполнении программы);
* перегруженную(**override**) версию виртуального метода **string ToString()** для формирования строки с полными данными класса, содержащую в том числе полные данные для каждого элемента списка **List<Person>**;
* метод **string ToShortString()**, который возвращает строку, содержащую только список фамилий и имен сотрудников и информацию о том, кем является соторудник – программистом или исследователем.

В классе **Department**

* для реализации интерфейса **IEnumerable<Person>** определить итератор для перебора всех объектов **Person** из списка **List<Person>**;
* определить именованный итератор для перебора только исследователей (объектов типа **Researcher**)из списка **List<Person>**.

В методе **Main()**

1. Создать объект типа **Department**, вызвать метод **AddDefaults()** класса ивывести данные объекта **Department** сначала с помощью метода **ToString(),** а затем с помощью метода **ToShortString()**.
2. С помощью оператора **foreach** для итератора, определенного в классе **Department**, вывести список всех сотрудников.
3. С помощью оператора **foreach** для именованного итератора, определенного в классе **Department**, вывести список всех исследователей.
4. С помощью метода **DeepCopy()** создать полную копию объекта **Department**. В элементе списка **List<Person>**,который имеет тип **Researcher,** заменить в первом элементе его списка статей значение свойства типа **string** с названием статьи.Вывести копию и исходный объект (исходный объект **Department** должен остаться без изменений).

Замечания

Определенные в некоторых классах базовой библиотеки методы **Clone()** и **Copy()** создают ограниченную (shallow) копию объекта – при копировании объекта копии создаются только для полей структурных типов, для полей ссылочных типов копируются только ссылки. В результате в ограниченной копии объекта поля-ссылки указывают на те же объекты, что и в исходном объекте.

Метод **DeepCopy()** должен создать полные копии всех объектов, ссылки на которые содержат поля типа. После создания полная копия не зависит от исходного объекта - изменение любого поля или свойства исходного объекта не должно приводить к изменению копии.

При реализации метода **DeepCopy()** в классе, который имеет поле типа **System.Collections.Generic.List<T>**, следует иметь в виду, что определенный в классе конструктор **List<T>( IEnumerable<T> collection)** при создании копии коллекции, состоящей из элементов ссылочных типов, копирует только ссылки.

Метод **DeepCopy()** должен создать как копии элементов коллекции **List<T>**, так и полные копии объектов, на которые ссылаются элементы коллекции.

**Исходный код каждого из классов программы должен находиться в отдельном файле, имя которого должно содержать название класса.**

В следующем семестре часть этих типов будут использоваться в лабораторных работах, посвященных пользовательскому интерфейсу.

Срок сдачи лабораторной работы 17 октября