Лабораторная работа 3. Вариант 1

Требования к программе

В программе определить

- класс **Person** (из лабораторной работы 2);
- класс **Researcher**, производный от класса **Person** (из лабораторной работы 2);
- класс **Programmer**, производный от класса **Person**;
- класс **Team**;
- интерфейс **interface IDeepCopy { object DeepCopy(); }**

Класс Person (из лабораторной работы 2)

Класс **Person** реализует интерфейс **IDeepCopy.**

В классе **Person** определить открытые автореализуемые свойства (с методами **get** и **set**)

- типа **string[]** для имени и фамилии;
- типа **System.DateTime** для даты рождения.

В классе **Person** определить

- конструктор с параметрами типа **string**, **string**, **DateTime** для инициализации данных класса;
- перегруженную(**override**) версию виртуального метода **string ToString()** для формирования строки с данными класса.

В классе **Person**

- переопределить (**override**) виртуальный метод **bool Equals (object obj);**
- определить операции **==** и **!=** ;
- переопределить виртуальный метод **int GetHashCode().**

Операции **==** и **!=** и метод **bool Equals (object obj)** надо переопределить так, чтобы объекты считались равными, если равны все данные объектов. Для класса **Person** это означает, что равны имя, фамилия и дата рождения.

Класс Researcher (из лабораторной работы 2)

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

В классе **Researcher** определить открытые автореализуемые свойства (с методами **get** и **set**)

- типа **string** для темы исследований;
- типа **int** для числа публикаций.

В классе **Researcher** определить:

- конструктор для инициализации всех данных класса;
- перегруженную(**override**) версию виртуального метода **string ToString()** для формирования строки с данными класса.

Класс Programmer

Класс **Programmer** является производным от класса **Person,** реализует интерфейс **IDeepCopy** и имеет

- открытое автореализуемое свойство типа **double** для информации о стаже работы;
- открытое автореализуемое свойство типа **string** для темы исследований;
- конструктор для инициализации полей класса;
- перегруженную(**override**) версию виртуального метода **string ToString()** для формирования строки с данными класса.

Класс Team

Класс **Team** реализует интерфейс **IDeepCopy**.
В классе **Team** определить

- открытое автореализуемое свойство типа **string** для названия группы;
- открытое автореализуемое свойство типа **List<Person>** для списка участников;
- конструктор с параметром типа **string** для инициализации данных класса;
- открытый метод **AddPerson (params Person[] persons)** для добавления в список **List<Person>** новых элементов типа **Person**, **Researcher** и **Programmer**; метод проверяет, что в списке **List<Person>** нет элемента с такими же данными базового подобъекта **Person** как и у элемента, который добавляется;
- открытый метод **AddDefaults()**, который добавляет несколько элементов **Person, Programmer, Reseacher** в список **List<Person>**;
- открытый метод **bool IsProgrammer (Person ps),** возвращающий значение **true**, если динамический тип **ps** это **Programmer, false** в противном случае;
- перегруженную(**override**) версию виртуального метода **string ToString()** для формирования строки со значениями всех данных класса, в том числе с данными каждого элемента списка **List<Person>**;

В пространстве имен **System** определен универсальный делегат

public delegate bool Predicate<T> (T obj);

В классе **Team** определить именованный итератор с параметром **IEnumerable<Person> Subset (Predicate<Person> Filter),** возвращающий последовательность элементов, которые удовлетворяют условию, реализованному в методе-предикате **Filter**.

В методе **Main()**

1. Создать объект типа **Team**, вызвать метод **Add_Defaults()** и вывести данные объекта **Team.**
2. С помощью метода **DeepCopy()** сделать копию объекта, изменить данные одного из элеменов в списке **List<Person>** копии и вывести копию, чтобы проверить, что изменение данных копии не изменяет данные исходного объекта.
3. **С** помощью оператора **foreach** для именованного итератора **Subset** вывести все элементы типа **Programmer** из списка **List<Person>** класса **Team.**
4. В классе **Program** определить метод **bool HasPublication (Person ps)**, который возвращает значение **true**, если динамический тип **ps** это **Researcher** и число публикаций больше 0, **false** в противном случае. С помощью оператора **foreach** для именованного итератора **Subset** вывести все элементы типа **Researcher** из списка **List<Person>** класса **Team** с числом публикаций больше 0.

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