# Лабораторная работа 1

## Классы, свойства, индексаторы.

**Требования к программе, общие для всех вариантов**

Определить класс **Person**, который имеет

* закрытое поле типа string, в котором хранится имя;
* закрытое поле типа string, в котором хранится фамилия;
* закрытое поле типа System.DateTime для даты рождения. В классе **Person** определить конструкторы:
* конструктор c тремя параметрами типа string, string, DateTime для

инициализации всех полей класса;

* конструктор без параметров, инициализирующий все поля класса некоторыми значениями по умолчанию.

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

* свойство типа string для доступа к полю с именем;
* свойство типа string для доступа к полю с фамилией;
* свойство типа DateTime для доступа к полю с датой рождения;
* свойство типа int c методами get и set для получения информации(get) и изменения (set) года рождения в закрытом поле типа DateTime, в котором хранится дата рождения.

В классе **Person** определить

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

## Вариант 1. Требования к программе

Определить тип **Education** - перечисление(enum) со значениями Specialist, Вachelor, SecondEducation.

Определить класс **Exam**, который имеет три открытых автореализуемых свойства, доступных для чтения и записи:

* свойство типа string, в котором хранится название предмета;
* свойство типа int, в котором хранится оценка;
* свойство типа System.DateTime для даты экзамена.

В классе **Exam** определить:

* конструктор с параметрами типа string, int и DateTime для инициализации всех свойств класса;
* конструктор без параметров, инициализирующий все свойства класса некоторыми значениями по умолчанию;
* перегруженную(override) версию виртуального метода string ToString() для формирования строки со значениями всех свойств класса.

Определить класс **Student**, который имеет

* закрытое поле типа Person, в котором хранятся данные студента;
* закрытое поле типа Education для информации о форме обучения;
* закрытое поле типа int для номера группы;
* закрытое поле типа Exam \*+ для информации об экзаменах, которые сдал студент.

В классе **Student** определить конструкторы:

* конструктор c параметрами типа Person, Education, int для инициализации соответствующих полей класса;
* конструктор без параметров, инициализирующий поля класса значениями по умолчанию.

В классе **Student** определить свойства c методами get и set:

* свойство типа Person для доступа к полю с данными студента;
* свойство типа Education для доступа к полю с формой обучения;
* свойство типа int для доступа к полю с номером группы;
* свойство типа Exam \*+ для доступа к полю со списком экзаменов.

В классе **Student** определить

* свойство типа double ( только с методом get), в котором вычисляется средний балл как среднее значение оценок в списке сданных экзаменов;
* индексатор булевского типа (только с методом get) с одним параметром типа Education; значение индексатора равно true, если значение поля с

формой обучения студента совпадает со значением индекса, и false в противном случае;

* метод void AddExams ( params Exam [] ) для добавления элементов в список экзаменов;
* перегруженную версию виртуального метода string ToString() для формирования строки со значениями всех полей класса, включая список экзаменов;
* виртуальный метод string ToShortString(), который формирует строку со значениями всех полей класса без списка экзаменов, но со значением среднего балла.

В методе **Main()**

* 1. Создать один объект типа Student, преобразовать данные в текстовый вид с помощью метода ToShortString() и вывести данные.
  2. Вывести значения индексатора для значений индекса Education.Specialist, Education.Bachelor и Education.SecondEducation.
  3. Присвоить значения всем определенным в типе Student свойствам, преобразовать данные в текстовый вид с помощью метода ToString() и вывести данные.
  4. C помощью метода AddExams( params Exam\*+ ) добавить элементы в список экзаменов и вывести данные объекта Student, используя метод ToString().
  5. Сравнить время выполнения операций с элементами одномерного, двумерного прямоугольного и двумерного ступенчатого массивов с одинаковым числом элементов типа Exam.

## Вариант 2. Требования к программе

Определить тип **Frequency -** перечисление(enum) со значениями Weekly, Monthly, Yearly.

Определить класс **Article**, который имеет три открытых автореализуемых свойства, доступных для чтения и записи:

* свойство типа Person, в котором хранятся данные автора статьи;
* свойство типа string для названия статьи;
* свойство типа double для рейтинга статьи.

В классе **Article** определить:

* конструктор c параметрами типа Person, string, double для инициализации всех свойств класса;
* конструктор без параметров, инициализирующий все свойства класса некоторыми значениями по умолчанию;
* перегруженную(override) версию виртуального метода string ToString() для формирования строки со значениями всех свойств класса.

Определить класс **Magazine**, который имеет

* закрытое поле типа string c названием журнала;
* закрытое поле типа Frequency с информацией о периодичности выхода журнала;
* закрытое поле типа DateTime c датой выхода журнала;
* закрытое поле типа int с тиражом журнала;
* закрытое поле типа Article\*+ со списком статей в журнале.

В классе **Magazine** определить конструкторы:

* конструктор с параметрами типа string, Frequency, DateTime, int для инициализации соответствующих полей класса;
* конструктор без параметров, инициализирующий поля класса значениями по умолчанию.

В классе **Magazine** определить свойства c методами get и set:

* свойство типа string для доступа к полю с названием журнала;
* свойство типа Frequency для доступа к полю с информацией о периодичности выхода журнала;
* свойство типа DateTime для доступа к полю c датой выхода журнала;
* свойство типа int для доступа к полю с тиражом журнала;
* свойство типа Article\*+ для доступа к полю со списком статей.

В классе **Magazine** определить

* свойство типа double ( только с методом get), в котором вычисляется среднее значение рейтинга в списке статей;
* индексатор булевского типа (только с методом get) с одним параметром типа Frequency; значение индексатора равно true, если значение поля типа Frequency совпадает со значением индекса, и false в противном случае;
* метод void AddArticles (params Article[]) для добавления элементов в список статей в журнале;
* перегруженную версию виртуального метода string ToString() для формирования строки со значениями всех полей класса, включая список статей;
* виртуальный метод string ToShortString(), который формирует строку со значениями всех полей класса без списка статей, но со значением среднего рейтинга статей.

В методе **Main()**

1. Создать один объект типа Magazine, преобразовать данные в текстовый вид с помощью метода ToShortString() и вывести данные.
2. Вывести значения индексатора для значений индекса Frequency.Weekly, Frequency.Monthly и Frequency.Yearly.
3. Присвоить значения всем определенным в типе Magazine свойствам, преобразовать данные в текстовый вид с помощью метода ToString() и вывести данные.
4. C помощью метода AddArticles( params Article\*+ ) добавить элементы в список статей и вывести данные объекта Magazine, используя метод ToString().
5. Сравнить время выполнения операций с элементами одномерного, двумерного прямоугольного и двумерного ступенчатого массивов с одинаковым числом элементов типа Article.

## Вариант 3. Требования к программе

Определить тип **TimeFrame** - перечисление(enum) со значениями Year, TwoYears, Long.

Определить класс **Paper**, который имеет три открытых автореализуемых свойства, доступных для чтения и записи:

* свойство типа string, в котором хранится название публикации;
* свойство типа Person для автора публикации;
* свойство типа DateTime c датой публикации.

В классе **Paper** определить

* конструктор c параметрами типа string, Person, DateTime для инициализации всех свойств класса;
* конструктор без параметров, инициализирующий все свойства класса некоторыми значениями по умолчанию;
* перегруженную(override) версию виртуального метода string ToString() для формирования строки со значениями всех полей класса.

Определить класс **ResearchTeam**, который имеет

* закрытое поле типа string c названием темы исследований;
* закрытое поле типа string с названием организации;
* закрытое поле типа int – регистрационный номер;
* закрытое поле типа TimeFrame для информации о продолжительности исследований;
* закрытое поле типа Paper\*+, в котором хранится список публикаций.

В классе **ResearchTeam** определить конструкторы:

* конструктор c параметрами типа string, string, int, TimeFrame для инициализации соответствующих полей класса;
* конструктор без параметров, инициализирующий поля класса значениями по умолчанию.

В классе **ResearchTeam** определить свойства c методами get и set:

* свойство типа string для доступа к полю с названием темы исследований;
* свойство типа string для доступа к полю с названием организации;
* свойство типа int для доступа к полю с номером регистрации;
* свойство типа TimeFrame для доступа к полю с продолжительностью исследований;
* свойство типа Paper\*+ для доступа к полю со списком публикаций по теме исследований.

В классе **ResearchTeam** определить

* свойство типа Paper ( только с методом get), которое возвращает ссылку на публикацию с самой поздней датой выхода; если список публикаций пустой, свойство возвращает значение null;
* индексатор булевского типа (только с методом get) с одним параметром типа TimeFrame; значение индексатора равно true, если значение поля с информацией о продолжительности исследований совпадает со значением индекса, и false в противном случае;
* метод void AddPapers ( params Paper[] ) для добавления элементов в список публикаций;
* перегруженную версию виртуального метода string ToString() для формирования строки со значениями всех полей класса, включая список публикаций;
* виртуальный метод string ToShortString(), который формирует строку со значениями всех полей класса без списка публикаций.

В методе **Main()**

1. Создать один объект типа ResearchTeam, преобразовать данные в текстовый вид с помощью метода ToShortString() и вывести данные.
2. Вывести значения индексатора для значений индекса TimeFrame.Year, TimeFrame.TwoYears, TimeFrame.Long.
3. Присвоить значения всем определенным в типе ResearchTeam свойствам, преобразовать данные в текстовый вид с помощью метода ToString() и вывести данные.
4. С помощью метода AddPapers (params Paper \*+) добавить элементы в список публикаций и вывести данные объекта ResearchTeam.
5. Вывести значение свойства, которое возвращает ссылку на публикацию с самой поздней датой выхода;
6. Сравнить время выполнения операций с элементами одномерного, двумерного прямоугольного и двумерного ступенчатого массивов с одинаковым числом элементов типа Paper.

# Лабораторная работа 2

## Наследование. Исключения. Интерфейсы. Итераторы и блоки итераторов

**Информация для всех вариантов**

В классе **Person** из лабораторной работы 1 и в классах, дополнительно указанных в вариантах, надо

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

Реализация виртуального метода bool Equals (object obj) в классе System.Object определяет равенство объектов как равенство ссылок на объекты. Некоторые классы из базовой библиотеки BCL переопределяют метод Equals(). В классе System.String этот метод переопределен так, что равными считаются строки, которые совпадают посимвольно. Реализация метода Equals() в структурном типе DateTime равенство объектов DateTime определяет как равенство значений.

В лабораторной работе требуется переопределить метод Equals так, чтобы объекты считались равными, если равны все данные объектов. Для класса Person это означает, что равны даты рождения и посимвольно совпадают строки с именем и фамилией.

Определение операций == и != должно быть согласовано с переопределенным методом Equals, т.е. критерии, по которым проверяется равенство объектов в методе Equals, должны использоваться и при проверке равенства объектов в операциях == и !=.

Переопределение виртуального метода int GetHashCode() также должно быть согласовано с операциями == и !=. Виртуальный метод GetHashCode() используется некоторыми классами базовой библиотеки, например, коллекциями-словарями. Классы базовой библиотеки, вызывающие метод GetHashCode() из пользовательского типа, предполагают, что равным объектам отвечают равные значения хэш-кодов. Поэтому в случае, когда под равенством объектов понимается совпадение данных (а не ссылок), реализация метода

GetHashCode() должна для объектов с совпадающими данными возвращать равные значения хэш-кодов.

В классах, указанных в вариантах лабораторной работы, требуется определить метод object DeepCopy() для создания полной копии объекта. 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.ArrayList, следует иметь в виду, что определенные в классе ArrayList конструктор ArrayList(ICollection) и метод Clone() при создании копии коллекции, состоящей из элементов ссылочных типов, копируют только ссылки. Метод DeepCopy() должен создать как копии элементов коллекции ArrayList, так и полные копии объектов, на которые ссылаются элементы коллекции. Для типов, содержащих коллекции, реализация метода DeepCopy() упрощается, если в типах элементов коллекций также определить метод

DeepCopy().

## Вариант 1. Требования к программе

Определить интерфейс

**interface IDateAndCopy**

**{ object DeepCopy();**

**DateTime Date { get; set; }**

**}**

Определить новые версии классов **Exam**, **Person** и **Student** из лабораторной работы 1. В классы **Exam**, **Person** и **Student** добавить реализацию интерфейса IDateAndCopy. Новую версию класса **Student** определить как класс, производный от класса **Person**.

Все поля новой версии класса **Person** определить с доступом protected, сохранить все свойства, определенные в первой версии класса.

В новой версии класса **Person** дополнительно

* переопределить метод virtial bool Equals (object obj) и определить операции == и != так, чтобы равенство объектов типа Person трактовалось как совпадение всех данных объектов, а не ссылок на объекты Person;
* переопределить виртуальный метод int GetHashCode();
* определить виртуальный метод object DeepCopy();
* реализовать интерфейс IDateAndCopy.

Определить класс **Test**, который имеет два открытых автореализуемых свойства, доступных для чтения и записи:

* свойство типа string, в котором хранится название предмета;
* свойство типа bool для информации о том, сдан зачет или нет.

В классе **Test** определить:

* конструктор c параметрами типа string и bool для инициализации свойств класса;
* конструктор без параметров, инициализирующий все свойства класса некоторыми значениями по умолчанию;
* перегруженную(override) версию виртуального метода string ToString() для формирования строки со значениями всех свойств класса.

Класс **Student** определить как производный от класса **Person**. Новая версия класса **Student** имеет следующие поля:

* закрытое поле типа Education для информации о форме обучения;
* закрытое поле типа int для номера группы;
* закрытое поле типа System.Collections.ArrayList, в котором хранится список зачетов (объекты типа **Test**);
* закрытое поле типа System.Collections.ArrayList для списка экзаменов (объекты типа **Exam**).

Код следующих конструкторов, методов и свойств из старой версии класса **Student** необходимо изменить с учетом того, что часть полей класса перемещена в базовый класс Person, и в новой версии класса Student список экзаменов хранится в коллекции System.Collections.ArrayList:

* конструктор c параметрами типа Person, Education, int для инициализации соответствующих полей класса;
* конструктор без параметров для инициализации по умолчанию;
* свойство типа Person; метод get свойства возвращает объект типа Person, данные которого совпадают с данными подобъекта базового класса, метод set присваивает значения полям из подобъекта базового класса;
* свойство типа double ( только с методом get), в котором вычисляется средний балл как среднее значение оценок в списке сданных экзаменов;
* свойство типа System.Collections.ArrayList с методами get и set для доступа к полю со списком экзаменов;
* метод void AddExams ( params Exam[] ) для добавления элементов в список экзаменов;
* перегруженная версия виртуального метода string ToString() для формирования строки со значениями всех полей класса, включая список зачетов и экзаменов;
* виртуальный метод string ToShortString(), который формирует строку со значениями всех полей класса без списка зачетов и экзаменов, но со значением среднего балла.

Дополнительно в новой версии класса **Student**

* определить перегруженную версию виртуального метода object DeepCopy();
* реализовать интерфейс IDateAndCopy;
* определить свойство типа int с методами get и set для доступа к полю с номером группы. В методе set бросить исключение, если присваиваемое значение меньше или равно 100 или больше 599. При создании объекта- исключения использовать один из определенных в библиотеке CLR классов- исключений, инициализировать объект-исключение с помощью конструктора с параметром типа string, в сообщении передать информацию о допустимых границах для значения свойства.

В новой версии класса **Student** определить

* итератор для последовательного перебора всех элементов (объектов типа object) из списков зачетов и экзаменов (объединение);
* итератор c параметром для перебора экзаменов (объектов типа Exam) с оценкой больше заданного значения.

В методе **Main()**

1. Создать два объекта типа Person с совпадающими данными и проверить, что ссылки на объекты не равны, а объекты равны, вывести значения хэш- кодов для объектов.
2. Создать объект типа Student, добавить элементы в список экзаменов и зачетов, вывести данные объекта Student.
3. Вывести значение свойства типа Person для объекта типа Student.
4. С помощью метода DeepCopy() создать полную копию объекта Student. Изменить данные в исходном объекте Student и вывести копию и исходный объект, полная копия исходного объекта должна остаться без изменений.
5. В блоке try/catch присвоить свойству с номером группы некорректное значение, в обработчике исключения вывести сообщение, переданное через объект-исключение.
6. С помощью оператора foreach для итератора, определенного в классе Student, вывести список всех зачетов и экзаменов.
7. С помощью оператора foreach для итератора с параметром, определенного в классе Student, вывести список всех экзаменов с оценкой выше 3.

### Дополнительное задание:

В классе **Student**

* реализовать интерфейс System.Collections.IEnumerable для перебора названий всех предметов (объектов типа string), которые есть как в списке зачетов, так и в списке экзаменов (пересечение). Для этого определить вспомогательный класс StudentEnumerator, реализующий интерфейс System.Collections.IEnumerator.
* определить итератор для перебора сданных зачетов и экзаменов (объектов типа object), для этого определить метод, содержащий блок итератора и использующий оператор yield; сданный экзамен – экзамен с оценкой больше 2;
* определить итератор для перебора всех сданных зачетов (объектов типа Test), для которых сдан и экзамен, для этого определить метод, содержащий блок итератора и использующий оператор yield.

В методе **Main()**

1. С помощью оператора foreach для объекта типа Student вывести список предметов, которые есть как в списке зачетов, так и в списке экзаменов.
2. С помощью оператора foreach для итератора, определенного в классе Student, вывести список всех сданных зачетов и сданных экзаменов.

10.С помощью оператора foreach для итератора, определенного в классе Student, вывести список сданных зачетов, для которых сдан и экзамен.

## Вариант 2. Требования к программе

Определить интерфейс

**interface IRateAndCopy**

**{ double Rating { get;}**

**object DeepCopy();**

**}**

Определить новые версии классов **Person**, **Article** и **Magazine** из лабораторной работы 1. Класс **Magazine** определить как производный от класса **Edition**. В классы **Article** и **Magazine** добавить реализацию интерфейса IRateAndCopy.

В новой версии класса **Person** дополнительно

* переопределить метод virtial bool Equals (object obj) и определить операции == и != так, чтобы равенство объектов типа Person трактовалось как совпадение всех данных объектов, а не ссылок на объекты Person;
* переопределить виртуальный метод int GetHashCode();
* определить виртуальный метод object DeepCopy().

В новой версии класса **Article** дополнительно

* определить виртуальный метод object DeepCopy();
* реализовать интерфейс IRateAndCopy.

Определить класс **Edition**. Класс **Edition** имеет

* защищенное(protected) поле типа string c названием издания;
* защищенное поле типа DateTime c датой выхода издания;
* защищенное поле типа int с тиражом издания;

В классе **Edition** определить:

* конструктор с параметрами типа string, DateTime, int для инициализации соответствующих полей класса;
* конструктор без параметров для инициализации по умолчанию;
* свойства c методами get и set для доступа к полям типа;
* виртуальный метод object DeepCopy();
* свойство типа int с методами get и set для доступа к полю с тиражом издания; в методе set свойства бросить исключение, если присваиваемое значение отрицательно. При создании объекта-исключения использовать один из определенных в библиотеке CLR классов-исключений, инициализировать объект-исключение с помощью конструктора с параметром типа string, в сообщении передать информацию о допустимых значениях свойства.

В классе **Edition** переопределить (override):

* виртуальный метод virtial bool Equals (object obj) и определить операции

== и != так, чтобы равенство объектов типа Edition трактовалось как совпадение всех данных объектов, а не ссылок на объекты Edition;

* виртуальный метод int GetHashCode();
* перегруженную версию виртуального метода string ToString() для формирования строки со значениями всех полей класса.

Новая версия класса **Magazine** имеет базовый класс **Edition** и следующие поля:

* закрытое поле типа Frequency с информацией о периодичности выхода журнала;
* закрытое поле типа System.Collections.ArrayList со списком редакторов журнала (объектов типа **Person**).
* закрытое поле типа System.Collections.ArrayList, в котором хранится список статей в журнале (объектов типа **Article**).

Код следующих конструкторов, методов и свойств из старой версии класса **Magazine** необходимо изменить с учетом того, что часть полей класса перемещена в базовый класс Edition, и в новой версии класса Magazine для списка статей используется тип System.Collections.ArrayList:

* конструктор с параметрами типа string, Frequency, DateTime, int для инициализации соответствующих полей класса;
* конструктор без параметров для инициализации по умолчанию;
* свойство типа double (только с методом get), в котором вычисляется среднее значение рейтинга статей в журнале;
* свойство типа System.Collections.ArrayList для доступа к полю со списком статей в журнале;
* метод void AddArticles (params Article[]) для добавления элементов в список статей в журнале;
* перегруженная версия виртуального метода string ToString() для формирования строки со значениями всех полей класса, включая список статей и список редакторов;
* виртуальный метод string ToShortString(), который формирует строку со значениями всех полей класса без списка статей и списка редакторов, но со значением среднего рейтинга статей в журнале.

Дополнительно в новой версии класса **Magazine** реализовать

* свойство типа System.Collections.ArrayList для доступа к списку редакторов журнала;
* метод void AddEditors (params Person[]) для добавления элементов в список редакторов;
* перегруженную (override) версию виртуального метода object DeepCopy();
* интерфейс IRateAndCopy;
* свойство типа Edition; метод get свойства возвращает объект типа Edition, данные которого совпадают с данными подобъекта базового класса, метод set присваивает значения полям из подобъекта базового класса.

В новой версии класса **Magazine** определить

* итератор с параметром типа double для перебора статей с рейтингом больше некоторого заданного значения;
* итератор с параметром типа string для перебора статей, в названии которых есть заданная строка.

В методе **Main()**

1. Создать два объекта типа Edition с совпадающими данными и проверить, что ссылки на объекты не равны, а объекты равны, вывести значения хэш- кодов для объектов.
2. В блоке try/catch присвоить свойству с тиражом издания некорректное значение, в обработчике исключения вывести сообщение, переданное через объект-исключение.
3. Создать объект типа Magazine, добавить элементы в списки статей и редакторов журнала и вывести данные объекта Magazine.
4. Вывести значение свойства типа Edition для объекта типа Magazine.
5. С помощью метода DeepCopy() создать полную копию объекта Magazine. Изменить данные в исходном объекте Magazine и вывести копию и исходный объект, полная копия исходного объекта должна остаться без изменений.
6. С помощью оператора foreach для итератора с параметром типа double вывести список всех статей с рейтингом больше некоторого заданного значения.
7. С помощью оператора foreach для итератора с параметром типа string вывести список статей, в названии которых есть заданная строка.

### Дополнительное задание:

В классе **Magazine**

* реализовать интерфейс System.Collections.IEnumerable для перебора

статей (объектов типа Article), авторы которых не входят в список редакторов журнала; для этого определить вспомогательный класс MagazineEnumerator, реализующий интерфейс System.Collections.IEnumerator.

* определить итератор для перебора статей (объектов типа Article), авторы которых являются редакторами журнала, для этого определить метод, содержащий блок итератора и использующий оператор yield.
* определить итератор для перебора редакторов журнала (объектов типа Person), у которых нет статей в журнале, для этого определить метод, содержащий блок итератора и использующий оператор yield.

В методе **Main()**

1. С помощью оператора foreach для объекта типа Magazine вывести список статей, авторы которых не являются редакторами журнала.
2. С помощью оператора foreach для итератора, определенного в классе Magazine, вывести список статей, авторы которых являются редакторами журнала.

10.С помощью оператора foreach для итератора, определенного в классе Magazine, вывести список редакторов, у которых нет статей в журнале.

## Вариант 3. Требования к программе

Определить интерфейс

**interface INameAndCopy**

**{ string Name { get; set;} object DeepCopy();**

**}**

Определить новые версии классов **Person**, **Paper** и **ResearchTeam** из лабораторной работы 1. Класс **ResearchTeam** определить как производный от класса **Team**. В классы **Team** и **ResearchTeam** добавить реализацию интерфейса INameAndCopy.

В классе **Paper** определить виртуальный метод object DeepCopy(). В новой версии класса **Person** дополнительно

* переопределить метод virtial bool Equals (object obj) и определить операции == и != так, чтобы равенство объектов типа Person трактовалось как совпадение всех данных объектов, а не ссылок на объекты Person;
* переопределить виртуальный метод int GetHashCode();
* определить виртуальный метод object DeepCopy().

Определить класс **Team**. Класс **Team** имеет

* защищенное (protected) поле типа string с названием организации;
* защищенное поле типа int – регистрационный номер.

В классе **Team** определить:

* конструктор с параметрами типа string и int для инициализации полей класса;
* конструктор без параметров для инициализации по умолчанию;
* свойство типа string для доступа к полю с названием организации;
* свойство типа int для доступа к полю с номером регистрации; в методе set бросить исключение, если присваиваемое значение меньше или равно 0; при создании объекта-исключения использовать один из определенных в библиотеке CLR классов-исключений, инициализировать объект-исключение с помощью конструктора с параметром типа string.

В классе **Team**

* определить виртуальный метод object DeepCopy();
* реализовать интерфейс INameAndCopy.

В классе **Team** переопределить (override):

* виртуальный метод virtial bool Equals (object obj) и определить операции

== и != так, чтобы равенство объектов типа Team трактовалось как совпадение всех данных объектов, а не ссылок на объекты Team;

* виртуальный метод int GetHashCode();
* виртуальный метод string ToString() для формирования строки со значениями всех полей класса.

Новая версия класса **ResearchTeam** имеет базовый класс **Team** и следующие поля:

* закрытое поле типа string c названием темы исследований;
* закрытое поле типа TimeFrame с информацией о продолжительности исследований;
* закрытое поле типа System.Collections.ArrayList со списком участников проекта (объектов типа Person);
* закрытое поле типа System.Collections.ArrayList для списка публикаций (объектов типа Paper).

Код следующих конструкторов, методов и свойств из старой версии класса **ResearchTeam** необходимо изменить с учетом того, что часть полей класса перемещена в базовый класс **Team**, и в новой версии класса ResearchTeam для списка публикаций используется тип System.Collections.ArrayList:

* конструктор c параметрами типа string, string, int, TimeFrame для инициализации соответствующих полей класса;
* конструктор без параметров для инициализации по умолчанию;
* свойство типа System.Collections.ArrayList для доступа к полю со списком публикаций;
* свойство типа Paper (только с методом get), которое возвращает ссылку на публикацию с самой поздней датой выхода; если список публикаций пустой, свойство возвращает значение null;
* метод void AddPapers (params Paper\*+ ) для добавления элементов в список публикаций;
* перегруженная версия виртуального метода string ToString() для формирования строки со значениями всех полей класса, включая список публикаций и список участников проекта;
* метод string ToShortString(), который формирует строку со значениями всех полей класса без списка публикаций и списка участников проекта.

Дополнительно в новой версии класса **ResearchTeam** определить

* перегруженную версию виртуального метода object DeepCopy();
* свойство типа System.Collections.ArrayList для доступа к полю со списком участников проекта;
* метод void AddMembers ( params Person[] ) для добавления элементов в список участников проекта;
* свойство типа Team; метод get свойства возвращает объект типа Team, данные которого совпадают с данными подобъекта базового класса, метод set присваивает значения полям из подобъекта базового класса;
* реализовать интерфейс INameAndCopy.

В новой версии класса **ResearchTeam** определить

* итератор для последовательного перебора участников проекта (объектов типа Person), не имеющих публикаций;
* итератор с параметром типа int для перебора публикаций, вышедших за последние n лет, в котором число n передается через параметр итератора.

В методе **Main()**

1. Создать два объекта типа Team с совпадающими данными и проверить, что ссылки на объекты не равны, а объекты равны, вывести значения хэш- кодов для объектов.
2. В блоке try/catch присвоить свойству с номером регистрации некорректное значение, в обработчике исключения вывести сообщение, переданное через объект-исключение.
3. Создать объект типа ResearchTeam, добавить элементы в список публикаций и список участников проекта и вывести данные объекта ResearchTeam.
4. Вывести значение свойства Team для объекта типа ResearchTeam.
5. С помощью метода DeepCopy() создать полную копию объекта ResearchTeam. Изменить данные в исходном объекте ResearchTeam и вывести копию и исходный объект, полная копия исходного объекта должна остаться без изменений.
6. С помощью оператора foreach для итератора, определенного в классе ResearchTeam, вывести список участников проекта, которые не имеют публикаций.
7. С помощью оператора foreach для итератора с параметром, определенного в классе ResearchTeam, вывести список всех публикаций, вышедших за последние два года.

### Дополнительное задание:

В классе ResearchTeam

* реализовать интерфейс System.Collections.IEnumerable для перебора участников проекта (объектов типа Person), у которых есть публикации; для этого определить вспомогательный класс ResearchTeamEnumerator, реализующий интерфейс System.Collections.IEnumerator.
* определить итератор для перебора участников проекта (объектов типа Person), имеющих более одной публикации, для этого определить метод, содержащий блок итератора и использующий оператор yield.
* определить итератор для перебора публикаций (объектов типа Paper), вышедших за последний год, для этого определить метод, содержащий блок итератора и использующий оператор yield.

В методе **Main()**

1. С помощью оператора foreach для объекта типа ResearchTeam вывести список участников проекта, у которых есть публикации.
2. С помощью оператора foreach для итератора, определенного в классе ResearchTeam, вывести список участников проекта, имеющих более одной публикации.
3. C помощью оператора foreach для итератора, определенного в классе ResearchTeam, вывести список публикаций, вышедших за последний год.