[redacted]

[redacted]

Разработать программу в соответствии с заданием. Номер варианта определяется по номеру студента в списке студентов группы (номер 1 – задание 1; номер 2 – задание 2; …; номер 9 – задание 9; номер 10 – задание 1; номер 11 - задание -2; и т.д.)

Варианты 1-3

Классы, свойства, индексаторы. Одномерные, прямоугольные и ступенчатые массивы

Общие требования к программе для вариантов 1-3

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

В классе **Student** определить свойства с 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. С 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 с 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 с названием журнала;
- закрытое поле типа Frequency с информацией о периодичности выхода журнала;
- закрытое поле типа DateTime с 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 с 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 с 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. С 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 с датой публикации.

В классе **Paper** определить

- конструктор с 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 с 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 с 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 с 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.

Варианты 4-6

Наследование. Исключения. Интерфейсы. Итераторы и блоки итераторов

Общие требования к программе для вариантов 4-6

В классе **Person** из вариантов заданий 1-3 и в классах, дополнительно указанных ниже, 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 4. 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 с 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 с 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 с 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.

8. Student, вывести список сданных зачетов, для которых сдан и экзамен.

Вариант 5. 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 с названием издания;
- защищенное поле типа DateTime с 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 с 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.

Вариант 6. 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 с 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 с 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.

Варианты 7-9

Универсальные типы. Классы-коллекции. Методы расширения класса System.Linq.Enumerable

Общие требования к программе для вариантов 7-9

Во всех вариантах работы 7-9 требуется определить класс **TestCollections**, который содержит поля следующих типов

- System.Collections.Generic.List<TKey> ;
- System.Collections.Generic.List<string> ;
- System.Collections.Generic.Dictionary<TKey, TValue> ;
- System.Collections.Generic.Dictionary<string, TValue> .

Конкретные значения типовых параметров TKey и TValue зависят от варианта. Во всех вариантах тип ключа TKey и тип значения TValue связаны отношением базовый-производный. Во всех вариантах в классе TValue определено свойство, которое возвращает ссылку на объект типа TKey с данными, совпадающими с данными подобъекта базового класса (это свойство должно возвращать ссылку на объект типа TKey, а не ссылку на вызывающий объект TValue).

В конструкторе класса **TestCollections** создаются коллекции с заданным числом элементов. Надо сравнить время поиска элемента в коллекциях-списках List<TKey> и время поиска элемента по ключу и элемента по значению в коллекциях-словарях Dictionary<TKey,TValue>.

Для автоматической генерации элементов коллекций в классе TestCollections надо определить статический метод, который принимает один целочисленный параметр типа int и возвращает ссылку на объект типа TValue.

Каждый объект TValue содержит подобъект базового класса TKey. Соответствие между значениями целочисленного параметра метода и подобъектами TKey класса TValue должно быть взаимно-однозначным – равным значениям параметра должны отвечать равные объекты TKey и наоборот. Равенство объектов типа TKey трактуется так же, как это было сделано в лабораторной работе 2 при определении операций равенства объектов.

Все четыре коллекции содержат одинаковое число элементов. Каждому элементу из коллекции List<TKey> должен отвечать элемент в коллекции Dictionary<TKey, TValue> с равным значением ключа. Список List<string> состоит из строк, которые получены в результате вызова метода ToString() для объектов TKey из списка List<TKey>. Каждому элементу списка List<string> отвечает элемент в коллекции-словаре Dictionary<string, TValue> с равным значением ключа типа string.

Число элементов в коллекциях вводится пользователем в процессе работы приложения. Если при вводе была допущена ошибка, приложение должно обработать исключение, сообщить об ошибке ввода и повторить прием ввода до тех пор, пока не будет правильно введено целочисленное значение.

Для четырех разных элементов – первого, центрального, последнего и элемента, не входящего в коллекцию – надо измерить время поиска

- элемента в коллекциях List<TKey> и List<string> с помощью метода Contains;
- элемента по ключу в коллекциях Dictionary< TKey, TValue> и Dictionary <string, TValue > с помощью метода ContainsKey;

- значения элемента в коллекции Dictionary< TKey, TValue > с помощью метода ContainsValue.

Так как статический метод для автоматической генерации элементов должен обеспечивать взаимно-однозначное соответствие между значением целочисленного параметра метода и объектами TKey, этот метод можно использовать как при создании коллекций с большим числом элементов, так и для генерации элемента для поиска.

Вариант 7. Требования к программе

Определить новые версии классов **Person** и **Student** из заданий 4-6.

В класс **Person** добавить реализацию интерфейсов

- System.IComparable для сравнения объектов типа Person по полю с фамилией;
- System.Collections.Generic.IComparer<Person> для сравнения объектов типа Person по дате рождения.

В новой версии класса **Student** для списков зачетов и экзаменов использовать типы

- System.Collections.Generic.List<Test> для списка зачетов;
- System.Collections.Generic.List<Exam> для списка экзаменов.

В новой версии класса **Student** сохранить все остальные поля, свойства и методы из предыдущей версии класса, внести необходимые исправления в код свойств и методов из-за изменения типов полей для списков зачетов и экзаменов.

Определить **вспомогательный класс**, реализующий интерфейс System.Collections.Generic.IComparer<Student>, который можно использовать для сравнения объектов типа Student по среднему баллу.

Определить класс **StudentCollection**, который содержит

- закрытое поле типа System.Collections.Generic.List<Student>;
- метод void AddDefaults(), с помощью которого можно добавить некоторое число элементов типа Student для инициализации коллекции по умолчанию;
- метод void AddStudents (params Student[]) для добавления элементов в список List<Student>;
- перегруженную версию виртуального метода string ToString() для формирования строки с информацией обо всех элементах списка List<Student>, включающую значения всех полей, список зачетов и экзаменов для каждого элемента Student;
- метод string ToShortString(), который формирует строку с информацией обо всех элементах списка List<Student>, содержащую значения всех полей, средний балл, число зачетов и число экзаменов для каждого элемента Student, но без списков зачетов и экзаменов.

В классе **StudentCollection** определить методы, выполняющие сортировку списка List<Student>

- по фамилии студента с использованием интерфейса IComparable, реализованного в классе Person;
- по дате рождения студента с использованием интерфейса IComparer<Person>, реализованного в классе Person;
- по среднему баллу с использованием интерфейса IComparer<Student>, реализованного во вспомогательном классе.

В классе **StudentCollection** определить свойства и методы, выполняющие операции со списком List<Student> с использованием методов расширения класса System.Linq.Enumerable, и статические методы-селекторы, которые необходимы для выполнения соответствующих операций со списком:

- свойство типа double (только с методом get), возвращающее максимальное значение среднего балла для элементов списка List<Student>; если в коллекции нет элементов, свойство возвращает некоторое значение по умолчанию; для поиска максимального значения среднего балла использовать метод Max класса System.Linq.Enumerable;
- свойство типа IEnumerable<Student> (только с методом get), возвращающее подмножество элементов списка List<Student> с формой обучения Education.Specialist; для формирования подмножества использовать метод Where класса System.Linq.Enumerable;
- метод List<Student> AverageMarkGroup(double value), который возвращает список, в который входят элементы Student из списка List<Student> с заданным значением среднего балла; для формирования списка использовать методы Group и ToList класса System.Linq.Enumerable.

Определить класс **TestCollections**, в котором в качестве типа TKey используется класс Person, а в качестве типа TValue - класс Student. Класс содержит закрытые поля с коллекциями типов

- System.Collections.Generic.List<Person>;
- System.Collections.Generic.List<string>;
- System.Collections.Generic.Dictionary <Person, Student>;
- System.Collections.Generic.Dictionary <string, Student>.

В классе **TestCollections** определить

- статический метод с одним целочисленным параметром типа int, который возвращает ссылку на объект типа Student и используется для автоматической генерации элементов коллекций;
- конструктор с параметром типа int (число элементов в коллекциях) для автоматического создания коллекций с заданным числом элементов;
- метод, который вычисляет время поиска элемента в списках List<Person> и List<string>, время поиска элемента по ключу и время поиска элемента по значению в коллекциях-словарях Dictionary<Person, Student> и Dictionary<string, Student>.

1. В методе **Main()**

2. Создать объект типа StudentCollection. Добавить в коллекцию несколько различных элементов типа Student и вывести объект StudentCollection.

3. Для созданного объекта StudentCollection вызвать методы, выполняющие сортировку списка List<Student> по разным критериям, и после каждой сортировки вывести данные объекта. Выполнить сортировку

- по фамилии студента;
- по дате рождения;
- по среднему баллу.

4. Вызвать методы класса StudentCollection, выполняющие операции со списком List<Student>, и после каждой операции вывести результат операции. Выполнить

- вычисление максимального значения среднего балла для элементов списка;
- фильтрацию списка для отбора студентов с формой обучения Education.Specialist;
- группировку элементов списка по значению среднего балла; вывести все группы элементов.

5. Создать объект типа TestCollections. Вызвать метод для поиска в коллекциях первого, центрального, последнего и элемента, не входящего в коллекции. Вывести значения времени поиска для всех четырех случаев. Вывод должен содержать информацию о том, к какой коллекции и к какому элементу относится данное значение.

Вариант 8. Требования к программе

Определить новые версии классов **Edition** и **Magazine** из заданий 4-6.

В новой версии класса **Magazine** использовать типы

- System.Collections.Generic.List<Person> для списка редакторов журнала;
- System.Collections.Generic.List<Article> для списка статей в журнале.

В новых версиях классов **Edition** и **Magazine** сохранить все остальные поля, свойства и методы из предыдущей версии класса, внести необходимые исправления в код свойств и методов из-за изменения типов полей для списка редакторов и списка статей.

В класс **Edition** добавить реализацию

- интерфейса System.IComparable для сравнения объектов Edition по полю с названием издания;
- интерфейса System.Collections.Generic.IComparer<Edition> для сравнения объектов Edition по дате выхода издания.

Определить **вспомогательный класс**, реализующий интерфейс System.Collections.Generic.IComparer<Edition>, который можно использовать для сравнения объектов типа Edition по тиражу издания.

Определить класс **MagazineCollection**, который содержит

- закрытое поле типа System.Collections.Generic.List<Magazine>;

- метод void AddDefaults (), с помощью которого в список List<Magazine> можно добавить некоторое число элементов типа Magazine для инициализации коллекции по умолчанию;
- метод void AddMagazines (params Magazine []) для добавления элементов в список List<Magazine>;
- перегруженную версию виртуального метода string ToString() для формирования строки с информацией обо всех элементах списка List<Magazine>, в том числе значения всех полей, список редакторов журнала и список статей в журнале для каждого элемента Magazine;
- виртуальный метод string ToShortString(), который формирует строку с информацией обо всех элементах списка List<Magazine>, содержащую значения всех полей, средний рейтинг статей, число редакторов журнала и число статей в журнале для каждого элемента Magazine, но без списков редакторов и статей.

В классе **MagazineCollection** определить свойства и методы, выполняющие сортировку списка List<Magazine>

- по названию издания с использованием интерфейса IComparable, реализованного в классе Edition;
- по дате выхода издания с использованием интерфейса IComparer<Edition>, реализованного в классе Edition;
- по тиражу издания с использованием интерфейса IComparer<Edition>, реализованного во вспомогательном классе.

В классе **MagazineCollection** определить методы, выполняющие операции со списком List<Magazine> с использованием методов расширения класса System.Linq.Enumerable и статические методы-селекторы, которые необходимы для выполнения соответствующих операций с коллекциями:

- свойство типа double (только с методом get), возвращающее максимальное значение среднего рейтинга статей для элементов списка List<Magazine>; если в коллекции нет элементов, свойство возвращает некоторое значение по умолчанию; для поиска максимального значения среднего рейтинга статей надо использовать метод Max класса System.Linq.Enumerable;
- свойство типа IEnumerable<Magazine> (только с методом get), возвращающее подмножество элементов списка List<Magazine> с периодичностью выхода журнала Frequency.Monthly; для формирования подмножества использовать метод Where класса System.Linq.Enumerable;
- метод List<Magazine> RatingGroup(double value), который возвращает список, содержащий элементы Magazine из List<Magazine> со средним рейтингом статей, который больше или равен value; для формирования списка использовать методы Group и ToList класса System.Linq.Enumerable.

Определить класс **TestCollections**, в котором в качестве типа TKey используется класс Edition, а в качестве типа TValue - класс Magazine. Класс содержит закрытые поля с коллекциями типов

- System.Collections.Generic.List<Edition>;
- System.Collections.Generic.List<string>;

- System.Collections.Generic.Dictionary <Edition, Magazine>;
- System.Collections.Generic.Dictionary <string, Magazine>.

В классе **TestCollection** определить

- статический метод с одним целочисленным параметром типа int, который возвращает ссылку на объект типа Magazine и используется для автоматической генерации элементов коллекций;
- конструктор с параметром типа int (число элементов в коллекциях) для автоматического создания коллекций с заданным числом элементов;
- метод, который вычисляет время поиска элемента в списках List<Edition> и List<string>, время поиска элемента по ключу и время поиска элемента по значению в коллекциях-словарях Dictionary< Edition, Magazine> и Dictionary<string, Magazine>.

В методе **Main()**

1. Создать объект типа MagazineCollection. Добавить в коллекцию несколько элементов типа Magazine с разными значениями полей и вывести объект MagazineCollection.

2. Для созданного объекта MagazineCollection вызвать методы, выполняющие сортировку списка List<Magazine> по разным критериям, и после каждой сортировки вывести данные объекта. Выполнить сортировку

- по названию издания;
- по дате выхода издания;
- по тиражу издания.

3. Вызвать методы класса MagazineCollection, выполняющие операции со списком List<Magazine>, и после каждой операции вывести результат операции. Выполнить

- вычисление максимального значения среднего рейтинга статей для элементов списка; вывести максимальное значение;
- фильтрацию списка для отбора журналов с периодичностью выхода Frequency.Monthly, вывести результат фильтрации;
- группировку элементов списка по значению среднего рейтинга статей; вывести все группы элементов.

4. Создать объект типа TestCollections. Вызвать метод для поиска в коллекциях первого, центрального, последнего и элемента, не входящего в коллекции. Вывести значения времени поиска для всех четырех случаев. Вывод должен содержать информацию о том, к какой коллекции и к какому элементу относится данное значение.

Вариант 9. Требования к программе

Определить новые версии классов **Team** и **ResearchTeam** из заданий 4-6.

В новой версии класса **ResearchTeam** использовать типы

- System.Collections.Generic.List<Person>для списка участников проекта;
- System.Collections.Generic.List<Paper> для списка публикаций;

В новых версиях классов **Team** и **ResearchTeam** сохранить все остальные поля, свойства и методы из предыдущих версий, внести необходимые исправления в код свойств и методов из-за изменения типа полей для списков.

В новую версию класса **Team** добавить реализацию интерфейса System.IComparable для сравнения объектов Team по полю с номером регистрации.

В новую версию класса **ResearchTeam** добавить реализацию интерфейса System.Collections.Generic.IComparer<ResearchTeam> для сравнения объектов ResearchTeam по названию темы исследований.

Определить **вспомогательный класс**, реализующий интерфейс System.Collections.Generic.IComparer<ResearchTeam>, который можно использовать для сравнения объектов типа ResearchTeam по числу публикаций.

Определить класс **ResearchTeamCollection**, который содержит

- закрытое поле типа System.Collections.Generic.List<ResearchTeam>;
- метод void AddDefaults (), с помощью которого в список List<ResearchTeam> можно добавить некоторое число элементов типа ResearchTeam для инициализации коллекции по умолчанию;
- метод void AddResearchTeams (params ResearchTeam []) для добавления элементов в список List<ResearchTeam>;
- перегруженную версию виртуального метода string ToString() для формирования строки с информацией обо всех элементах списка List<ResearchTeam>, которая содержит значения всех полей, список участников проекта и список публикаций для каждого элемента ResearchTeam;
- виртуальный метод string ToShortString(), который формирует строку с информацией обо всех элементах списка List<ResearchTeam>, включающую значения всех полей, число участников проекта и число публикаций для каждого элемента ResearchTeam, но без списков участников и публикаций.

В классе **ResearchTeamCollection** определить методы, выполняющие сортировку списка List<ResearchTeam>

- по номеру регистрации с использованием интерфейса IComparable, реализованного в классе Team;
- по названию темы исследований с использованием интерфейса IComparer<ResearchTeam>, реализованного в классе ResearchTeam;
- по числу публикаций с использованием интерфейса IComparer<ResearchTeam>, реализованного во вспомогательном классе.

В классе **ResearchTeamCollection** определить свойства и методы, выполняющие операции со списком List<ResearchTeam> с использованием методов расширения класса System.Linq.Enumerable и статические методы-селекторы, которые необходимы для выполнения соответствующих операций со списком:

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

если в коллекции нет элементов, свойство возвращает некоторое значение по умолчанию; для поиска минимального значения номера регистрации надо использовать метод Min класса System.Linq.Enumerable;
- 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;
- 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.

Определить класс **TestCollections**, в котором в качестве типа TKey используется класс Team, а в качестве типа TValue - класс ResearchTeam. Класс содержит закрытые поля с коллекциями типов

- System.Collections.Generic.List<Team>;
- System.Collections.Generic.List<string>;
- System.Collections.Generic.Dictionary <Team, ResearchTeam>;
- System.Collections.Generic.Dictionary <string, ResearchTeam>.

В классе **TestCollections** определить

- статический метод с одним целочисленным параметром типа int, который возвращает ссылку на объект типа ResearchTeam и используется для автоматической генерации элементов коллекций;
- конструктор с параметром типа int (число элементов в коллекциях) для автоматического создания коллекций с заданным числом элементов;
- метод, который вычисляет время поиска элемента в списках List<Team> и List<string>, время поиска элемента по ключу и время поиска значения элемента в коллекциях-словарях Dictionary<Team, ResearchTeam> и Dictionary <string, ResearchTeam>.

В методе **Main()**

1. Создать объект типа ResearchTeamCollection. Добавить в коллекцию несколько элементов типа ResearchTeam с разными значениями полей и вывести объект ResearchTeamCollection.

2. Для созданного объекта ResearchTeamCollection вызвать методы, выполняющие сортировку списка List<ResearchTeam> по разным критериям, и после каждой сортировки вывести данные объекта. Выполнить сортировку

- по номеру регистрации;
- по названию темы исследований;
- по числу публикаций.

3. Вызвать методы класса ResearchTeamCollection, выполняющие операции со списком List<ResearchTeam>, и после каждой операции вывести результат операции. Выполнить

- вычисление минимального значения номера регистрации для элементов списка; вывести минимальное значение;

- фильтрацию проектов с продолжительностью исследований TimeFrame.TwoYears, вывести результат фильтрации;
- группировку элементов списка по числу публикаций; вывести все группы элементов из списка.

4. Создать объект типа TestCollections. Вызвать метод для поиска в коллекциях первого, центрального, последнего и элемента, не входящего в коллекции. Вывести значения времени поиска для всех четырех случаев. Вывод должен содержать информацию о том, к какой коллекции и к какому элементу относится данное значение.