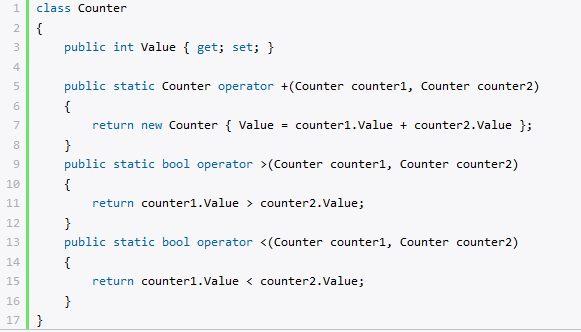
# **ПЕРЕГРУЗКА**

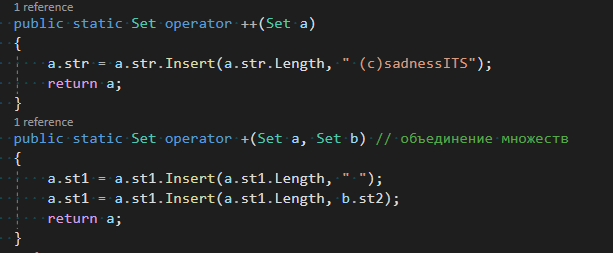
**Синтаксис:**



**Пример:**

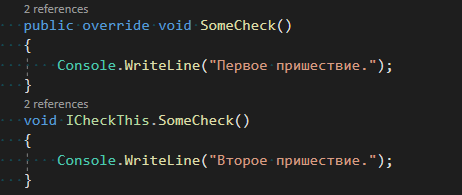


**Мой код:**



# **ИНТЕРФЕЙСЫ**

**Метод от класса и интерфейса:**

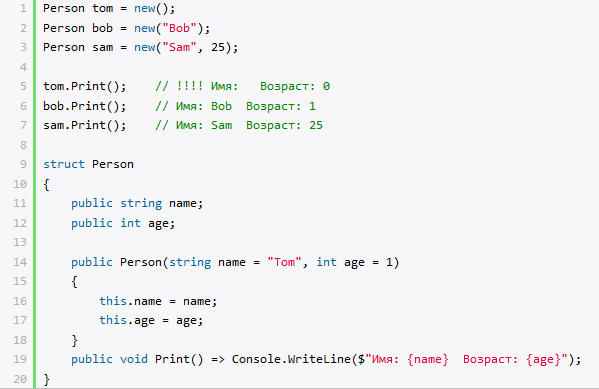


**Вызов методов в ходе выполнения Main():**

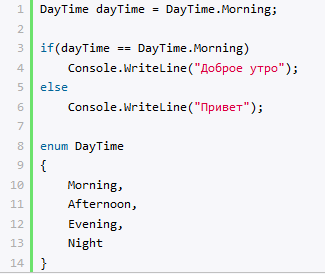


# **СТРУКТУРЫ И ПЕРЕЧИСЛЕНИЯ**

**Структура:**



**Перечисление:**

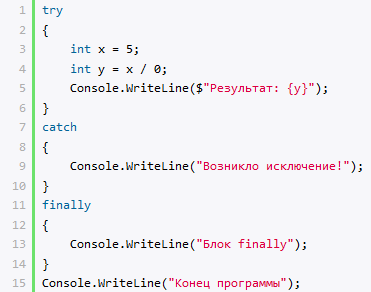


# **ИСКЛЮЧЕНИЯ:**

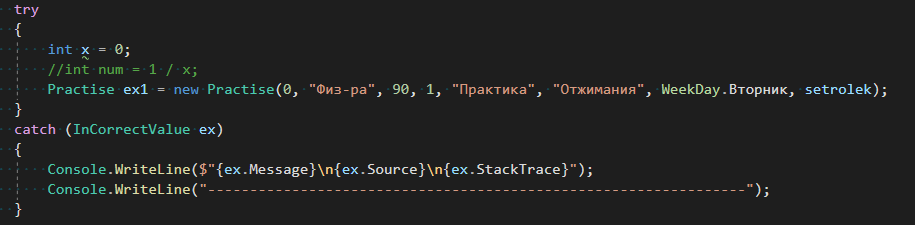
**Подключить:**

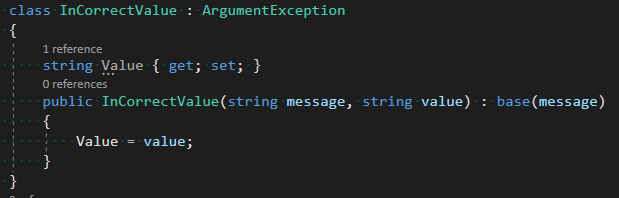


**Синтаксис:**



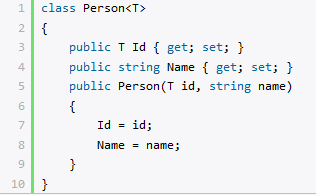
**Пример:**

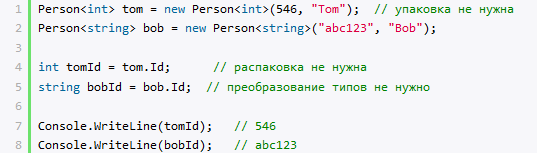




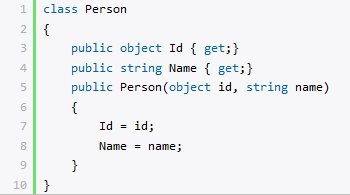
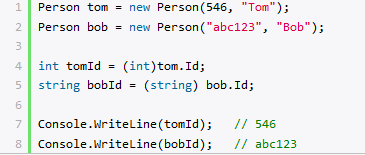
# **ОБОБЩЕНИЯ**

**Пример:**



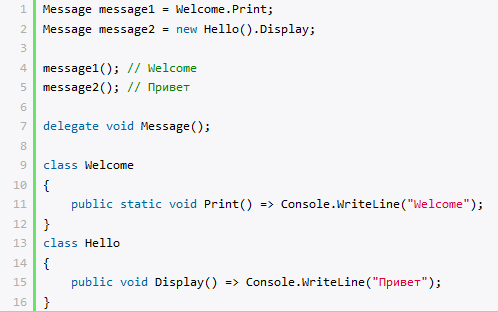


**Без обобщения и с упаковкой-распаковкой:**

# **ДЕЛЕГАТЫ, ЛЯМБДЫ**

**Делегаты:**



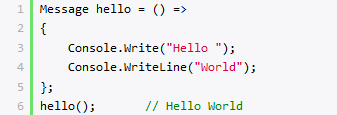
**Синтаксис лямбды:**



**Лябмда с неявной типизацией:**



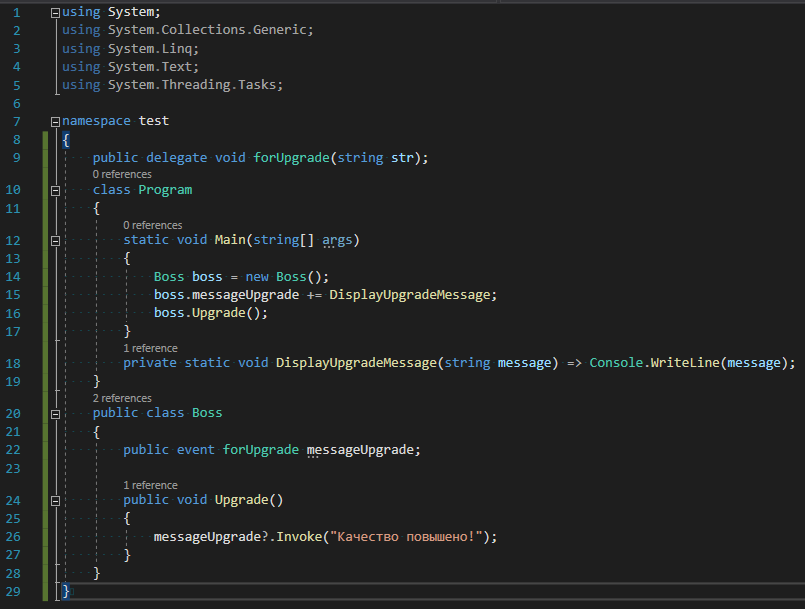
**Развернем лямбду (делегата нет на скрине, но есть под ним):**



7| delegate void Message();

# **СОБЫТИЯ**

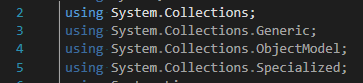
**Пример моего кода:**



По логике кода, Upgrade() должна все же содержать параметры, чтобы не только сообщение вывести, но и сделать какое-то изменение. Тут этого нет, потому что код получится слишком большим.

# **КОЛЛЕКЦИИ**

**Подключи:**

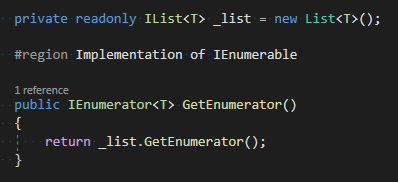


**Создание коллекции включает несколько этапов.**

1. Создание класса, который наследует интерфейс IList<t>:



1. Добавление свойств и необходимых полей.
2. Реализация всех методов интерфейса через объект коллекции этого интерфейса:

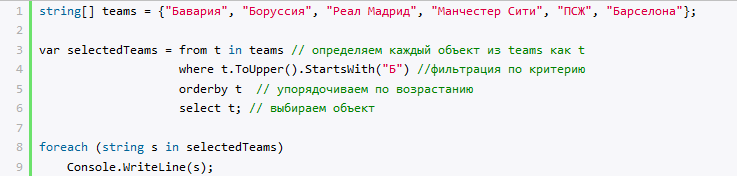


**Заметки:**

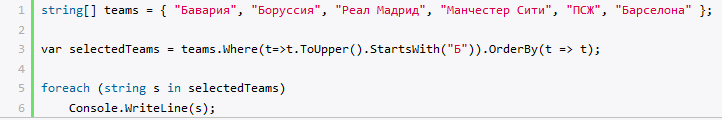
Дим, если решишь делать массив коллекций, то обращайся к элементу коллекции по вложенной формуле. [][], где первые скобки – элемент массива, а вторые – элемент коллекции.

# **LINQ**

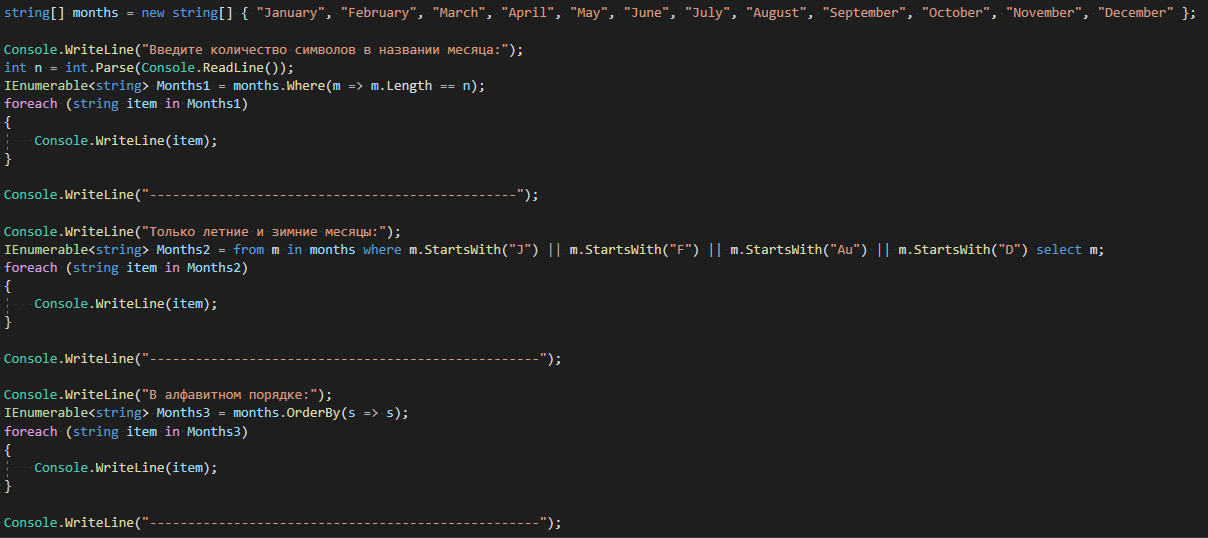
**Пример:**



**Если через методы расширения:**



**Мой код:**



**Список методов расширения:**

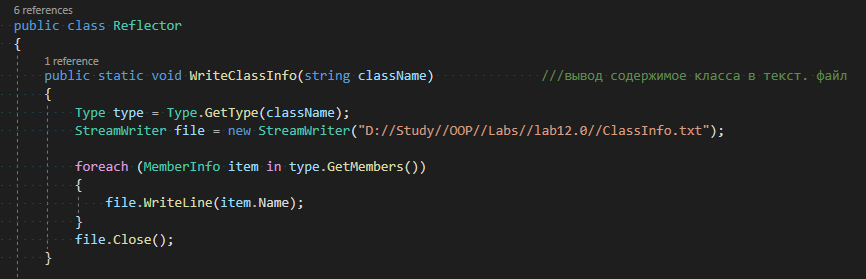
* Select: определяет проекцию выбранных значений
* Where: определяет фильтр выборки
* OrderBy: упорядочивает элементы по возрастанию
* OrderByDescending: упорядочивает элементы по убыванию
* ThenBy: задает дополнительные критерии для упорядочивания элементов возрастанию
* ThenByDescending: задает дополнительные критерии для упорядочивания элементов по убыванию
* Join: соединяет две коллекции по определенному признаку
* GroupBy: группирует элементы по ключу
* ToLookup: группирует элементы по ключу, при этом все элементы добавляются в словарь
* GroupJoin: выполняет одновременно соединение коллекций и группировку элементов по ключу
* Reverse: располагает элементы в обратном порядке
* All: определяет, все ли элементы коллекции удовлятворяют определенному условию
* Any: определяет, удовлетворяет хотя бы один элемент коллекции определенному условию
* Contains: определяет, содержит ли коллекция определенный элемент
* Distinct: удаляет дублирующиеся элементы из коллекции
* Except: возвращает разность двух коллекцию, то есть те элементы, которые создаются только в одной коллекции
* Union: объединяет две однородные коллекции
* Intersect: возвращает пересечение двух коллекций, то есть те элементы, которые встречаются в обоих коллекциях
* Count: подсчитывает количество элементов коллекции, которые удовлетворяют определенному условию
* Sum: подсчитывает сумму числовых значений в коллекции
* Average: подсчитывает cреднее значение числовых значений в коллекции
* Min: находит минимальное значение
* Max: находит максимальное значение
* Take: выбирает определенное количество элементов
* Skip: пропускает определенное количество элементов
* TakeWhile: возвращает цепочку элементов последовательности, до тех пор, пока условие истинно
* SkipWhile: пропускает элементы в последовательности, пока они удовлетворяют заданному условию, и затем возвращает оставшиеся элементы
* Concat: объединяет две коллекции
* Zip: объединяет две коллекции в соответствии с определенным условием
* First: выбирает первый элемент коллекции
* FirstOrDefault: выбирает первый элемент коллекции или возвращает значение по умолчанию
* Single: выбирает единственный элемент коллекции, если коллекция содердит больше или меньше одного элемента, то генерируется исключение
* SingleOrDefault: выбирает первый элемент коллекции или возвращает значение по умолчанию
* ElementAt: выбирает элемент последовательности по определенному индексу
* ElementAtOrDefault: выбирает элемент коллекции по определенному индексу или возвращает значение по умолчанию, если индекс вне допустимого диапазона
* Last: выбирает последний элемент коллекции
* LastOrDefault: выбирает последний элемент коллекции или возвращает значение по умолчанию

# **РЕФЛЕКСИЯ:**

**Подключи:**



**Запись инфы в файл и синтаксис работы с классом Type:**



Класс System.Type представляет изучаемый тип, инкапсулируя всю информацию о нем. С помощью его свойств и методов можно получить эту информацию. Некоторые из его свойств и методов:

* Метод FindMembers() возвращает массив объектов MemberInfo данного типа
* Метод GetConstructors() возвращает все конструкторы данного типа в виде набора объектов ConstructorInfo
* Метод GetEvents() возвращает все события данного типа в виде массива объектов EventInfo
* Метод GetFields() возвращает все поля данного типа в виде массива объектов FieldInfo
* Метод GetInterfaces() получает все реализуемые данным типом интерфейсы в виде массива объектов Type
* Метод GetMembers() возвращает все члены типа в виде массива объектов MemberInfo
* Метод GetMethods() получает все методы типа в виде массива объектов MethodInfo
* Метод GetProperties() получает все свойства в виде массива объектов PropertyInfo
* Свойство Name возвращает имя типа
* Свойство Assembly возвращает название сборки, где определен тип
* Свойство Namespace возвращает название пространства имен, где определен тип
* Свойство IsArray возвращает true, если тип является массивом
* Свойство IsClass возвращает true, если тип представляет класс
* Свойство IsEnum возвращает true, если тип является перечислением
* Свойство IsInterface возвращает true, если тип представляет интерфейс

# **РАБОТА С ФАЙЛАМИ И ПОТОКАМИ**

**Directory**

Статический класс Directory предоставляет ряд методов для управления каталогами. Некоторые из этих методов:

* CreateDirectory(path): создает каталог по указанному пути path
* Delete(path): удаляет каталог по указанному пути path
* Exists(path): определяет, существует ли каталог по указанному пути path. Если существует, возвращается true, если не существует, то false
* GetCurrentDirectory(): получает путь к текущей папке
* GetDirectories(path): получает список подкаталогов в каталоге path
* GetFiles(path): получает список файлов в каталоге path
* GetFileSystemEntries(path): получает список подкаталогов и файлов в каталоге path
* Move(sourceDirName, destDirName): перемещает каталог
* GetParent(path): получение родительского каталога
* GetLastWriteTime(path): возвращает время последнего изменения каталога
* GetLastAccessTime(path): возвращает время последнего обращения к каталогу
* GetCreationTime(path): возвращает время создания каталога

**Класс DirectoryInfo**

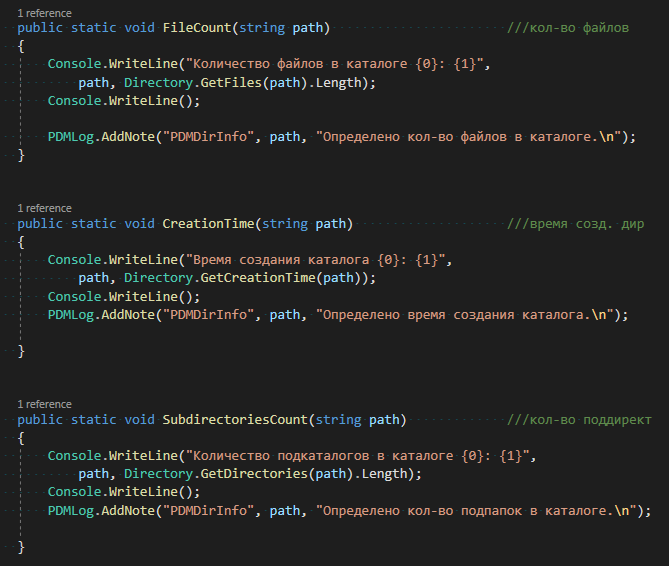
Основные методы класса DirectoryInfo:

* Create(): создает каталог
* CreateSubdirectory(path): создает подкаталог по указанному пути path
* Delete(): удаляет каталог
* GetDirectories(): получает список подкаталогов папки в виде массива DirectoryInfo
* GetFiles(): получает список файлов в папке в виде массива FileInfo
* MoveTo(destDirName): перемещает каталог

Основные свойства класса DirectoryInfo:

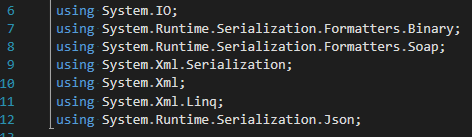
* CreationTime: представляет время создания каталога
* LastAccessTime: представляет время последнего доступа к каталогу
* LastWriteTime: представляет время последнего изменения каталога
* Exists: определяет, существует ли каталог
* Parent: получение родительского каталога
* Root: получение корневого каталога
* Name: имя каталога
* FullName: полный путь к каталогу

**Пример кода:**

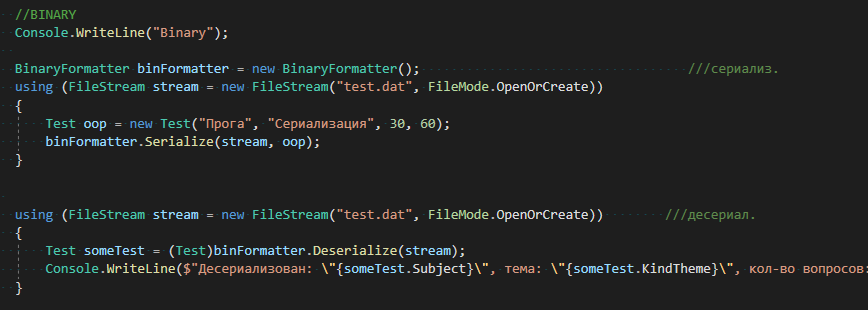


# **СЕРИАЛИЗАЦИЯ**

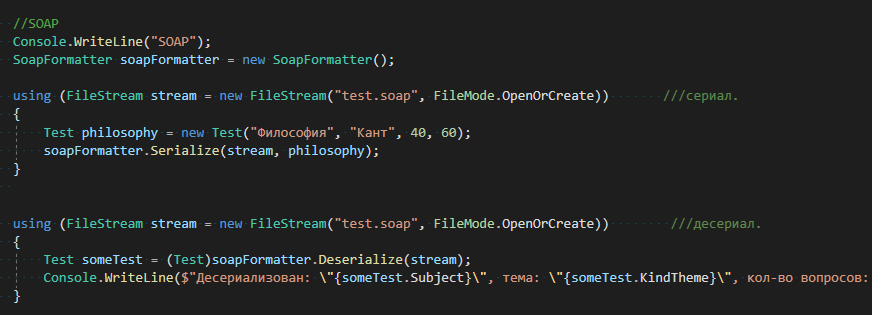
**Подключи:**



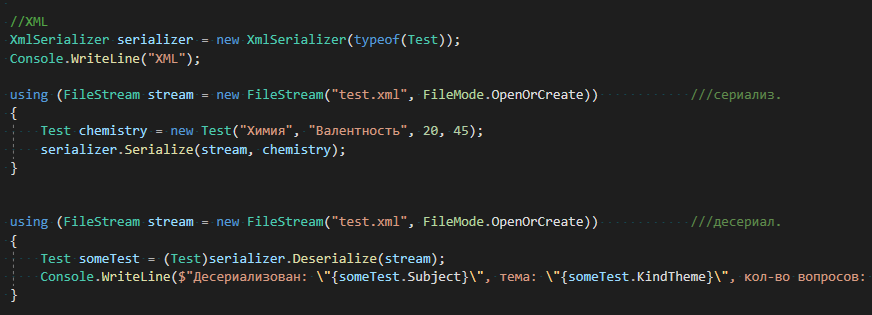
**Binary:**

****

**SOAP:**

****

**XML:**



**JSON:**

