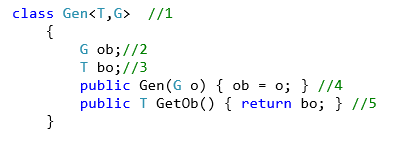
1. Что такое обобщение?

Термин *обобщение* означает параметризированный тип. Особая роль параметризированных типов состоит в том, что они позволяют создавать классы, структуры, интерфейсы, методы и делегаты, в которых обрабатываемые данные указываются в виде параметра.





1. Как можно наложить определенное ограничение на параметр?

class Account<T> where T : struct (class, new())

1. Несколько ограничений?

class Transaction<U, V>

        where U : Account<int>

        where V : struct

{ }

1. Перечислить все существующие ограничения на типы обобщения?

На базовый класс, на ссылочный тип, тип значения, на конструктор, на интерфейс

Если для универсального параметра задано несколько ограничений, то они должны идти в определенном порядке:

1. Название класса, class, struct. Причем мы можем одновременно определить только одно из этих ограничений
2. Название интерфейса
3. new()
4. class A

class В : A { }

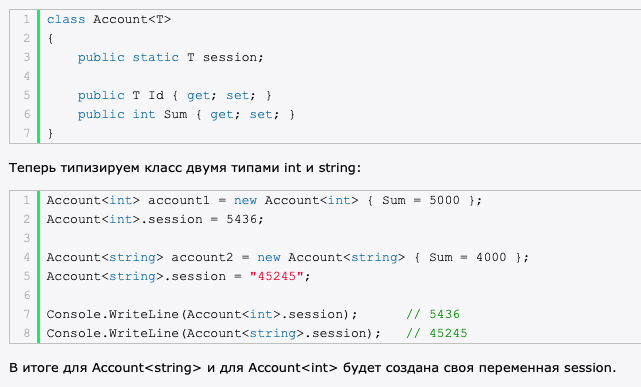
class С { }

class Test<T> where T : A { } - базовый класс

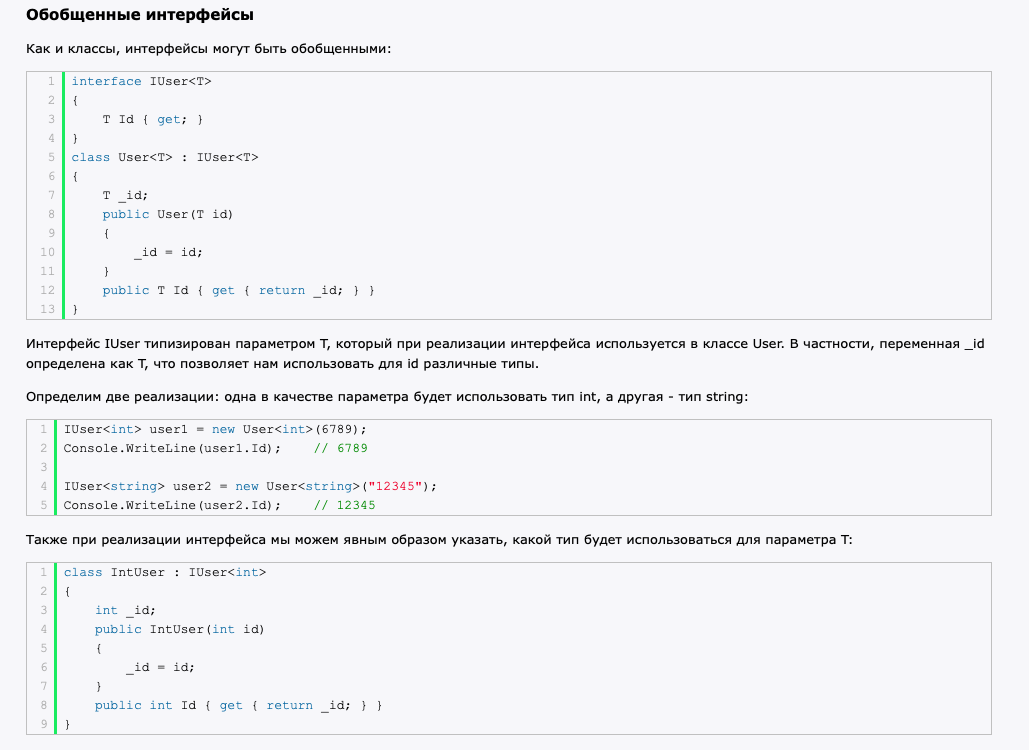
1. класс
2. структура
3. Когда обобщенный класс может действовать как базовый или производный?
4. Оператор default

В этом случае нам надо использовать оператор **default(T)**. Он присваивает ссылочным типам в качестве значения null, а типам значений - значение 0

1. Статические переменный в обобщенных классах



1. Пример обобщенного интерфейса



1. Отличие обобщенных классов от обобщенных структур

Есть ряд стандартных ограничений, которые мы можем использовать. В частности, можно указать ограничение, чтобы использовались только структуры или другие типы значений:

|  |  |
| --- | --- |
| 1  2 | class Account<T> where T : struct  {} |

При этом использовать в качестве ограничения конкретные структуры в отличие от классов нельзя.

Также можно задать в качестве ограничения ссылочные типы:

|  |  |
| --- | --- |
| 1  2 | class Transaction<T> where T : class  {} |

1. Классы для работы с файлами?

Класс FileStream не очень удобно применять для работы с текстовыми файлами. К тому же для этого в пространстве System.IO определены специальные классы: **StreamReader** и **StreamWriter**.

Здесь мы использовали один из них: new StreamWriter(writePath, false, System.Text.Encoding.Default). В качестве первого параметра передается путь к записываемому файлу. Второй параметр представляет булевую переменную, которая определяет, будет файл дозаписываться или перезаписываться. Если этот параметр равен true, то новые данные добавляются в конце к уже имеющимся данным. Если false, то файл перезаписывается. И если в первом случае файл перезаписывается, то во втором делается дозапись в конец файла.

Третий параметр указывает кодировку, в которой записывается файл.