1. Что такое обобщение (generic)

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.

обобщения являются не только конструкцией языка C#, но также определены для CLR. Это позволяет создавать экземпляры шаблонов с определенным типом-параметром на языке Visual Basic, даже если обобщенный класс определен на C#

class имя\_класса { // ...

1. . Пусть дан фрагмент листинга. В какой строчке содержится ошибка?
2. 3. Как можно наложить определенное ограничение на параметр?

С помощью выражения where T : Account мы указываем, что используемый тип T обязательно должен быть классом Account или его наследником. Благодаря подобному ограничению мы можем использовать внутри класса Transaction все объекты типа T именно как объекты Account и соответственно обращаться к их свойствам и методам. В качестве ограничения также может выступать и обобщенный класс: class Account<T>

{

    public T Id { get; private set; } // номер счета

    public int Sum { get; set; }

    public Account(T \_id)

    {

        Id = \_id;

    }

}

class Transaction<T> where T: Account<int>

{

    public T FromAccount { get; set; }  // с какого счета перевод

    public T ToAccount { get; set; }    // на какой счет перевод

    public int Sum { get; set; }        // сумма перевода

Однако класс Transaction может использовать только объекты класса Account<int> или его наследников.

В качестве ограничений мы можем использовать следующие типы:

* Классы
* Интерфейсы
* class - универсальный параметр должен представлять класс
* struct - универсальный параметр должен представлять структуру
* new() - универсальный параметр должен представлять тип, который имеет общедоступный (public) конструктор без параметров

**Стандартные ограничения**

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  {} |

А также можно задать с помощью слова **new** в качестве ограничения класс или структуру, которые имеют общедоступный конструктор без параметров:

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

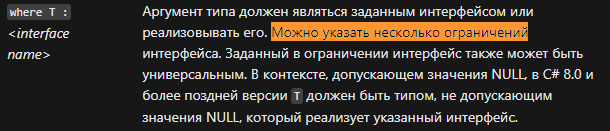
Если для универсального параметра задано несколько ограничений, то они должны идти в определенном порядке:

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

### Использование нескольких универсальных параметров

Если класс использует несколько универсальных параметров, то последовательно можно задать ограничения к каждому из них:

1. 4. Как можно наложить несколько ограничений на параметр?



С параметром типа может быть связано несколько ограничений. В этом случае ограничения указываются списком через запятую. Указывать ограничения class или struct одновременно с ограничением на базовый класс не разрешается. Далее по списку должно следовать ограничение на интерфейс, а последним по порядку — ограничение new(). Вы можете добавить любое количество ограничений типа, используя предложение where . Примеры:

import UIKit

func f<T>(t: T) where T: UIView, T: Encodable {}

class C<T> where T: UIView, T: Encodable {}

1. 5. Перечислите все существующие ограничения на типы данных обобщения?

В качестве ограничений мы можем использовать следующие типы:

* Классы
* Интерфейсы
* class - универсальный параметр должен представлять класс
* struct - универсальный параметр должен представлять структуру
* new() - универсальный параметр должен представлять тип, который имеет общедоступный (public) конструктор без параметров

1. 6. Какое ограничение на тип задано в следующем фрагменте листинга? class A { } class В : A { } class С { } class Test where T : A { }

Тест должен быть классом или наследником класса А

1. 7. Какое ограничение на тип задано в следующем фрагменте листинга? interface A { } class Test where T : class { }

Также можно задать в качестве использовались ограничения ссылочные типы:

1. 8. Какое ограничение на тип задано в следующем фрагменте листинга? interface A { } class Test where T : struct { }

В частности, можно указать ограничение, чтобы использовались только структуры или другие типы значений:

1. 9. Приведите примеры, когда обобщенный класс может действовать как базовый или производный класс.

 Главное отличие между иерархиями обобщенных и необобщенных классов заключается в том, что в первом случае аргументы типа, необходимые обобщенному базовому классу, должны передаваться всеми производными классами вверх по иерархии аналогично передаче аргументов конструктора.

В производном классе следует непременно указывать параметры типа, требующиеся его обобщенному базовому классу, даже если этот производный класс не обязательно должен быть обобщенным. Разумеется, в производный класс можно свободно добавлять его собственные параметры типа, если в этом есть потребность.

Первый вариант заключается в создание класса-наследника, который типизирован тем же типом, что и базовый:

Второй вариант представляет создание обычного необобщенного класса-наследника. В этом случае при наследовании у базового класса надо явным образом определить используемый тип:

Третий вариант представляет типизацию производного класса параметром совсем другого типа, отличного от универсального параметра в базовом классе. В этом случае для базового класса также надо указать используемый тип:

1. 10.В каких случаях в обобщениях может использоваться оператор default?

Иногда возникает необходимость присвоить переменным универсальных параметров некоторое начальное значение, в том числе и null. Но напрямую мы его присвоить не можем:

|  |  |
| --- | --- |
| 1 | T id = null; |

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

|  |  |
| --- | --- |
| 1  2  3  4 | class Account<T>  {  T id = default(T);  } |

1. 11.Поясните как использовать статические переменные в обобщенных классах.

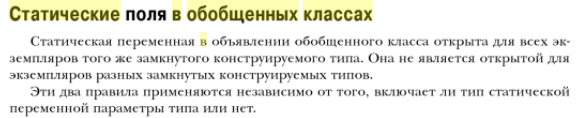
При типизации обобщенного класса определенным типом будет создаваться свой набор статических членов. Например, в классе Account определено следующее статическое поле:

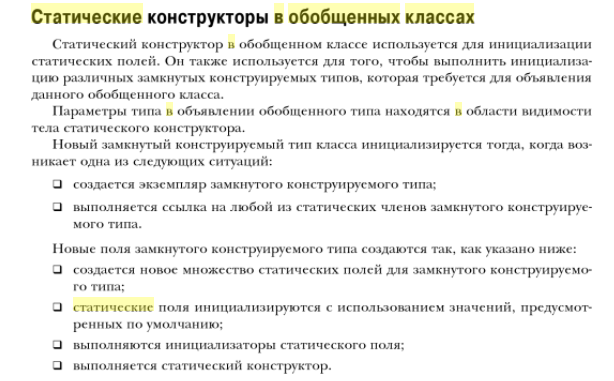
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | class Account<T>  {  public static T session;  public T Id { get; set; }  public int Sum { get; set; }  } |

Теперь типизируем класс двумя типами int и string:

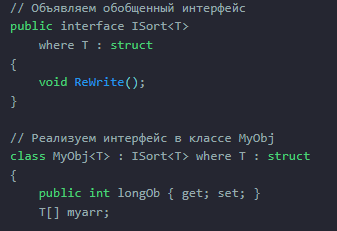
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | Account<int> account1 = new Account<int> { Sum = 5000 };  Account<int>.session = 5436;  Account<string> account2 = new Account<string> { Sum = 4000 };  Account<string>.session = "45245";  Console.WriteLine(Account<int>.session);      // 5436  Console.WriteLine(Account<string>.session);   // 45245 |

В итоге для Account<string> и для Account<int> будет создана своя переменная session.





1. 12.Приведите пример обобщенного интерфейса.



1. 13.В чем отличие обобщенных классов от обобщенных структур?

Они очень похожи на обобщенные классы, за исключением возможности наследования.

1. 14.Какие классы для работы с файлами вы знаете? Приведите пример

