private class People { }  
public class Generic<T> { } //обобщенный класс  
private Generic<People> one;  
public Generic<People> two; //ошибка

доступность к-руемых типов опр. на основе пересеч.  
доступности обобщ. типа и типа в списке аргументов

1) Что такое обобщение (generic)?

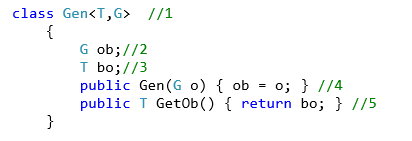
Обобщение – параметризированный тип. Позволяет создавать классы, структуры, интерфейсы, методы и делегаты, в которых обрабатываемые данные указываются в виде параметра. Позволяет создавать программный код, содержащий единственное (типизированное) решение задачи для различных типов, с его последующим применением для любого конкретного типа (int, float …)

**class имя\_класса<список\_параметров\_типа> { // ...** – форма объявления обобщенного класса

**имя\_класса<список\_аргументов\_типа> имя\_переменной = new имя\_класса<список\_параметров\_типа> (список\_аргументов\_конструктора);** - синтаксис объявления ссылки на обобщенный класс.

MyObj<int> - закрыто сконструированный тип  
MyObj<T> - открыто сконструированный тип

2. Пусть дан фрагмент листинга. В какой строчке содержится ошибка?



Public G GetOb() {return ob;}

Вы объявили, что метод возвращает объект типа T, но возвращает поле bo, которое также является типом T. Таким образом, невозможно однозначно определить, какой объект вернуть. Вероятно, вы хотели вернуть поле ob, которое имеет тип G.

3) Как можно наложить определенное ограничение на параметр?

class A<T> where T : ограничения…

class имя\_класса<параметр\_типа> where параметр\_типа : ограничения { // ...

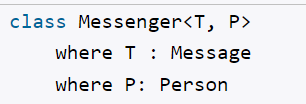
|  |  |
| --- | --- |
|  |  |

4) Как можно наложить несколько ограничений на параметр?

1. class (баз.) / struct  
2. interface  
3. new()

class MyObj<T> where T : MyClass, IMyInterface, new() {…  
//Параметр тип T должен быть заменен аргументом типа, наследующим от класса, реализующим интерфейс, использующим конструктор без параметра.

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



5) Перечислите все существующие ограничения на типы данных обобщения.

**Неприкрытое ограничение типа: (связь между парам. типа)**class MyObj<T,V> where V : T {…  
//V=T или насл. от него

**Ограничение типа ссылки:**public class Test<T> where T : class { }  
//все типы-классы, интерфейсы, делегаты, массивы, string

**Ограничение на интерфейс:**public class Test<T> where T : IComparable<T>  
//должен реализовывать обобщенный интерфейс

**Ограничение типа значения:**  
public class Test<T> where T : struct { }  
//Параметр типа должен быть унаследован от System.ValueType (с поддержкой null не подходят)

**Ограничение на базовый класс:**  
public class A { }  
public class B : A { }  
public class C { }

public class Test<T> where T : A { }

T<B> b = new T<B>()  
T<C> c = new T<C>() //нельзя

// T – наследник базового класса A или реализует интерфейс базового класса

// where T: BaseClass — параметр типа должен быть классом BaseClass или классом, производным от него;

**Ограничение на конструктор:**  
public class Test<T> where T : new() { }

// T имеет конструктор по умолчанию (мб создан без аргументов)  
//тип д. иметь неабстр. тип, имеющий открытый к-р без параметров

6. Какое ограничение на тип задано в следующем фрагменте листинга?

class A { }

class В : A { }

class С { }

class Test<T> where T : A { }

Ограничение гласит, что параметр типа T должен быть либо типом A, либо наследником (производным классом) от A. То есть, T может быть экземпляром класса A, класса B (поскольку B наследует от A), но не может быть экземпляром класса C, поскольку C не наследует от A.

7. Какое ограничение на тип задано в следующем фрагменте листинга?

interface A { }

class Test<T> where T : class { }

Ограничение типа-ссылки.

Т.е. T должен быть ссылочным типом или классом, реализующим интерфейс A.

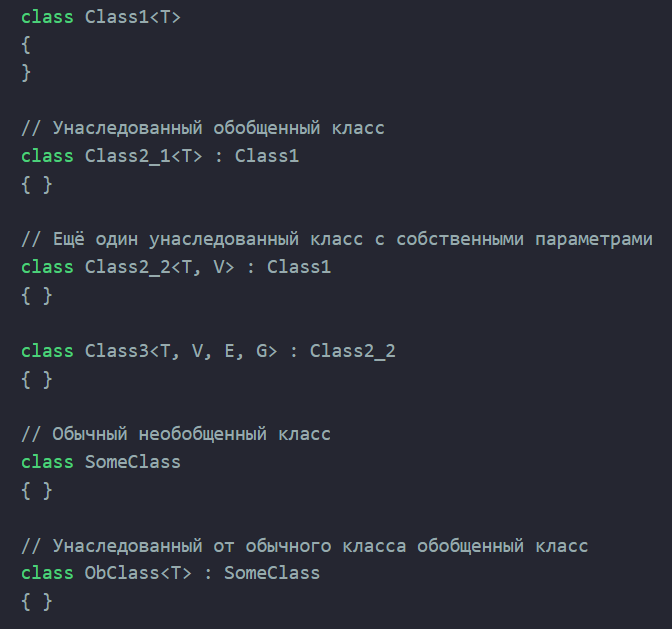
8. Какое ограничение на тип задано в следующем фрагменте листинга?

interface A { }

class Test<T> where T : struct { }

Ограничение типа-значения.

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



public class GenericBaseClass<T>{

public T Value { get; set; }

public GenericBaseClass(T value){

Value = value;}}

public class DerivedClass : GenericBaseClass<int> {

public DerivedClass(int value) : base(value){}}

public class GenericBaseClass<T>{

public T Value { get; set; }

public GenericBaseClass(T value) {

Value = value;}}

public class AnotherDerivedClass<T> : GenericBaseClass<T>{

public AnotherDerivedClass(T value) : base(value){ }

public void AdditionalMethod() {// доп функциональность }}

10) В каких случаях в обобщениях может использоваться оператор default?

T id = default(T); - определение значения по умолчанию для типа данных.  
//ссылочные типы – null, значимые – 0

Получить обобщенное значение по умолчанию для параметра типа.

Чтобы получить значение по умолчанию для обобщенного типа.

public class GenericClass<T>

{

public T GetDefaultValue()

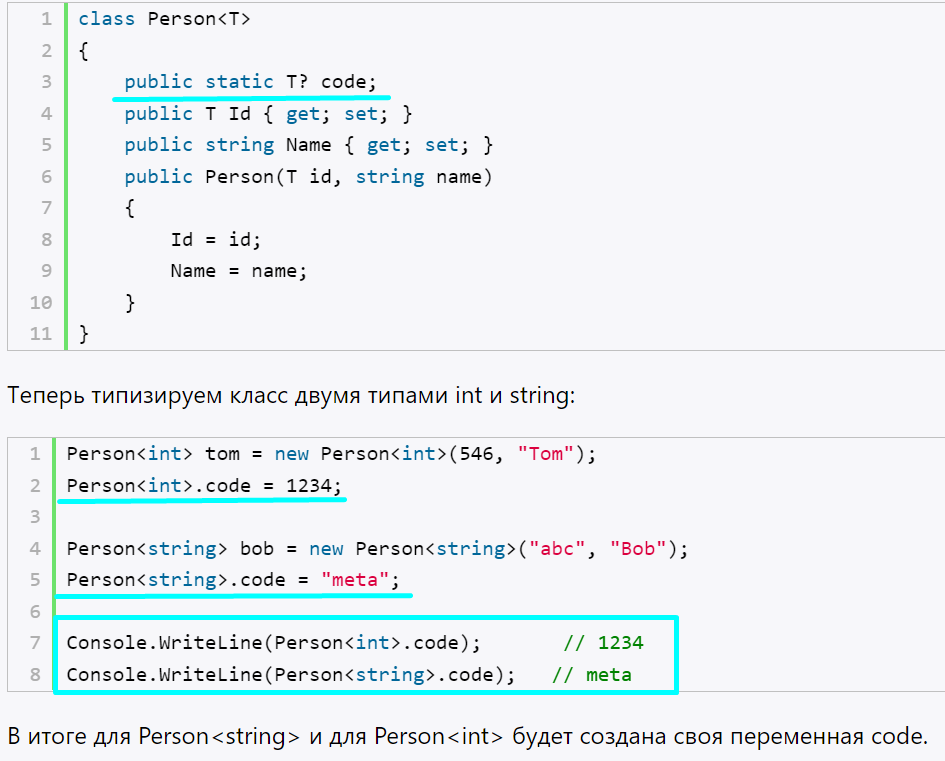
{

return default(T);

}

}

11) Поясните как использовать статические переменные в обобщенных классах



public class GenericClass<T> {

private static int instanceCount = 0;

public GenericClass() {

instanceCount++;

}

public static int GetInstanceCount(){

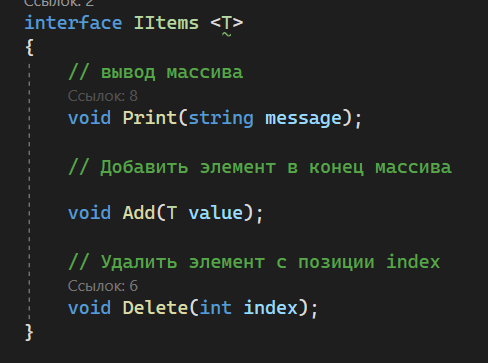
return instanceCount;

}

}

Статические переменные в обобщенных классах общие для всех экземпляров класса, и они используются для хранения данных, которые должны быть общими для всех экземпляров класса, независимо от конкретного типа.

12. Приведите пример обобщенного интерфейса.

public interface IGenericInterface<T>{

T GetValue();

void SetValue(T value);

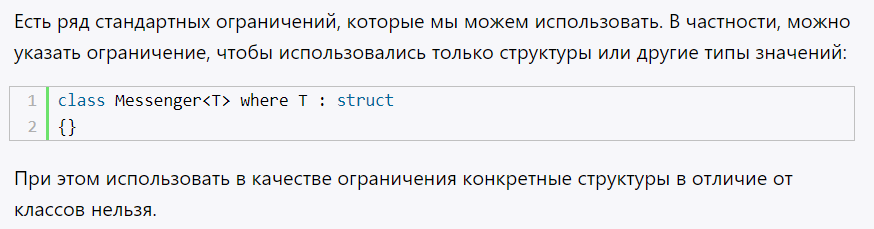
}

Этот интерфейс определяет два метода, GetValue и SetValue, которые работают с обобщенным типом T.

13) В чем отличие обобщенных классов от обобщенных структур?

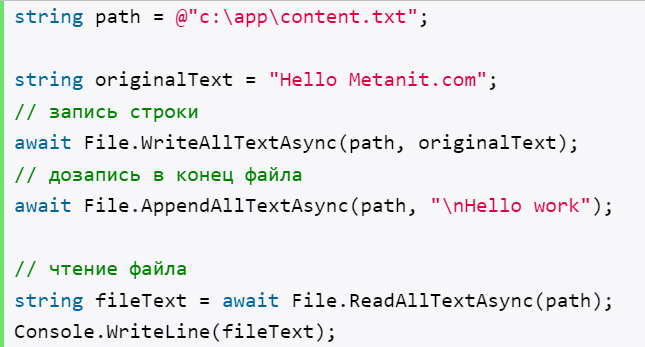
не поддерживают наследование от других классов или структур (struct)

Вероятнее всего, те же, что и между классом и структурой. Класс – является ссылочным типом (reference type), а структуры – значимый класс (value type). А следовательно, классы всегда создаются в, так называемой, “куче” (heap), а структуры создаются в стеке (stack).



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

* **File** и **FileInfo**. создавать, удалять, перемещать файлы, получать их свойства и многое другое.



* **WriteAllText (string path, string? contents)** / **Task WriteAllTextAsync (string path, string? contents, CancellationToken cancellationToken)**

записывают строку в текстовый файл. Если файл не существует, он создается. Если существует, то перезаписывается

* **AppendAllText(String, String)** / **AppendAllTextAsync(String, String, CancellationToken)**

добавляют в файл строку. Если файл не существует, то он создается

* **string ReadAllText (string path)** / **Task<string> ReadAllTextAsync (string path, CancellationToken cancellationToken)**

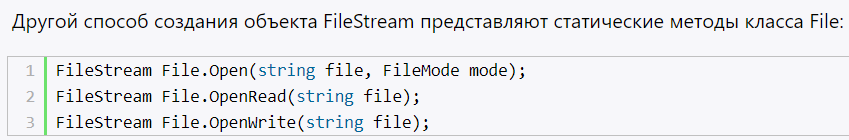
считывают содержимое текстового файла в строку

* Класс **FileStream** представляет возможности по считыванию из файла и записи в файл. Он позволяет работать как с текстовыми файлами, так и с бинарными. 

*где fileMode – режим доступа, filename –путь к файлу*

* Класс file





StreamReader, StreamWriter

* StreamWriter writer = new (path, false)// StreamReader reader = new StreamReader(path) (используя using)