**ВОПРОСЫ К ЛАБОРАТОРНОЙ РАБОТЕ №8:**

*private class People { }  
public class Generic<T> { }  
private Generic<People> one;  
public Generic<People> two; //ошибка, т.к. класс People**private****.***

доступность конструируемых типов определяется на основе пересечения  
доступности обобщённого типа и типа в списке аргументов

1) Что такое обобщение (generic)?

* **Обобщение (generic)** – параметризированный тип, позволяющий создать классы, структуры, интерфейсы, методы и делегаты, в которых обрабатываемые данные указываются в виде параметра.

Преимущество использования обобщений — в возможности избежать упаковки и распаковки значимых типов. Это позволяет увеличить производительность кода. Также благодаря обобщениям код можно использовать повторно.

*public class SuperArr  
 int[] s;*

*public class SuperArr<T>*

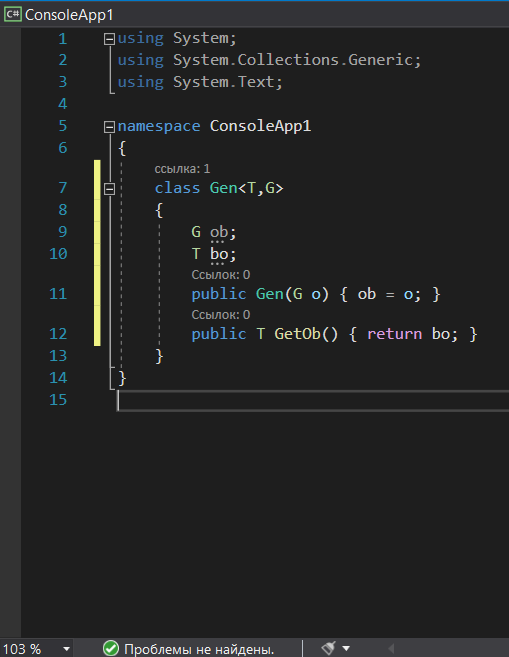
*T[] s;*

*SuperArr<int> a = new SuperArr<int>()  
SuperArr<Stack<int>> b = new SuperArr<Stack<int>>()  
SuperArr<People> c = new SuperArr<People>()*

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

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

Ошибки нет.



3) Как можно наложить определенное ограничение на параметр?

Универсальным параметром можно типизировать любой обобщённый тип любым типом данных, однако встречаются ситуации, когда необходимо конкретизировать обобщение. Для этого используется ключевое слово **where**:

**class A <T> where T : ограничения…**

Ограничением могут быть следующие типы:

* Классы (в качестве ограничения может применяться только один класс);
* интерфейсы;
* class – универсальный параметр должен быть классом;
* struct – универсальный параметр должен быть структурой;
* new() – универсальный параметр должен иметь общедоступный (public) конструктор без параметров.

4) Как можно наложить несколько ограничений на параметр?

При использовании нескольких ограничений, задать их можно в строгом порядке:

1. Название класса, class или struct. Эти ограничения нельзя применять одновременно.
2. Название интерфейса.
3. new().

Например, ***class UserList<T> where T: User , IUser , new() { }***

Можно задать ограничения нескольким универсальным параметрам:

***class UserList<T, U, V>***

***where T: User***

***where U: new()***

***{}***

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 наследник от него.

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

**Ограничение на интерфейс:**where T : IComparable<T>  
//должны реализовать обобщённый интерфейс.

**Ограничение типа значения:**  
class Test<T> where T : struct { }  
//аргумент имеет значимый тип (с поддержкой null не подходят).

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

public class Link<U> where U : A { }

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

**Ограничение на конструктор:**  
public class Link<U> where U : new() { }  
//тип должен иметь неабстрактный тип, имеющий открытый конструктор без параметров.

9) Приведите примеры, когда обобщенный класс может действовать как

базовый или производный класс.

class Gen2<T, V> : Gen<T> { // обобщённый класс как базовый

class Gen<T> : NonGen { // обобщённый класс как производный

class StringAccount : Account<string> // в этом случае, при наследовании, у родительского класса необходимо явным образом определить используемый тип данных

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

Иногда нам необходимо присвоить переменным универсальных параметров некоторое начальное значение, в том числе и null. Однако, напрямую мы его присвоить не сможем.

В этом случае нам надо использовать оператор default(T), который присваивает ссылочным типам в качестве значения null, а типам значений — значение 0

**T id = default(T)**  
// ссылочные типы – null, значимые – 0

11) Поясните как использовать статические переменные в обобщенных классах

class Account<T>  
 public static T session;

Account<int>.session = 5436;  
Account<string>session = “45612”;

12) Приведите пример обобщенного интерфейса.

Помимо обобщенных классов и методов, в C# допускаются обобщенные интерфейсы. Такие интерфейсы указываются аналогично обобщенным классам. Применяя обобщения, можно определять интерфейсы, объявляющие методы с обобщенными параметрами. Давайте рассмотрим пример применения обобщенных интерфейсов:

*public interface ISort<T>* // объявляем обобщенный интерфейс

*where T : struct*

*{*

*void ReWrite();*

*}*

13) В чем отличие обобщенных классов от обобщенных структур?

Struct не поддерживают наследование!

Структуры синтаксически очень похожи на классы, но существует принципиальное отличие, которое заключается в том, что класс – является ссылочным типом (reference type), а структуры – значимый тип (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.

FileStream - класс для чтения, записи, открытия и закрытия файлов в файловой системе.

StreamReader - для чтения данных из потока:

Метод ReadToEnd() считывает все данные из файла.

ReadLine() – считывает одну строку.

Свойство EndOfStream указывает, достигнут ли конец файла. Возвращает true или false.

StreamWriter - для записи данных в поток:

Метод WriteLine() записывает в файл построчно (то же самое, что и простая запись с помощью Write(), только в конце добавляется новая строка).

Нужно всегда помнить, что после работы с потоком, его нужно закрыть (освободить ресурсы), использовав метод Close().