Ответы на вопросы 8 лаба

1. Что такое обобщение (generic)?Обобщение — средство языка C#, позволяющее создавать программный код, содержащий единственное (типизированное) решение задачи для различных типов, с его последующим применением для любого конкретного типа (int, float, char и т.д.).

Использование обобщений дает следующие преимущества:

упрощение программного кода. Использование обобщений позволяет реализовывать алгоритм для любого типа элементов. Не нужно создавать похожие варианты алгоритма для различных типов (int, float, string и т.д.);

обеспечение типовой безопасности. В обобщенных элементах (классах, методах, коллекциях и т.п.) помещаются только объекты определенного типа, указываемого при их объявлении;

в обобщениях исключена необходимость явного приведения типа при преобразовании объекта или иного типа обрабатываемых данных;

повышение производительности. При использовании обобщений структурные типы передаются по значению. При этом не выполняется упаковка (boxing) и распаковка (unboxing), которые замедляют выполнение программы.

Как языковое средство, обобщения могут быть применены к:

классам;

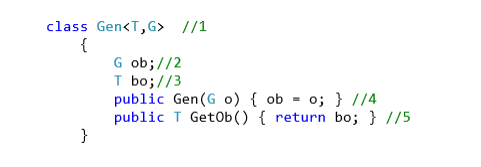
структурам;

интерфейсам;

методам;

делегатам.

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



…

3. Как можно наложить определенное ограничение на параметр?https://metanit.com/sharp/tutorial/3.38.php

<https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/generics/constraints-on-type-parameters>

через слово where : T aaa (где параметр T должен быть классом ааа либо его наследником )

4. Как можно наложить несколько ограничений на параметр?Если для универсального параметра задано несколько ограничений, то они должны идти в определенном порядке:

Название класса, class, struct. Причем мы можем одновременно определить только одно из этих ограничений

Название интерфейса

new()

пример:

interface IAccount

{

    int CurrentSum { get; set;}

}

class Person

{

    public string Name { get; set; }

}

class Transaction<T> where T: Person, IAccount, new()

{

}

5. Перечислите все существующие ограничения на типы данных

обобщения?В качестве ограничений мы можем использовать следующие типы:

Классы

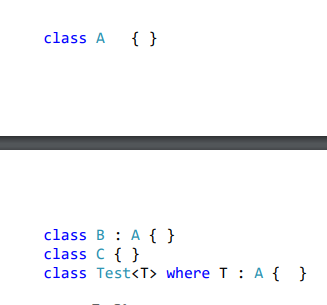
Интерфейсы

class - универсальный параметр должен представлять класс

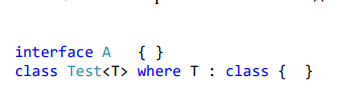
struct - универсальный параметр должен представлять структуру

new() - универсальный параметр должен представлять тип, который имеет общедоступный (public) конструктор без параметров

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



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



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



9. Приведите примеры, когда обобщенный класс может действовать как

базовый или производный класс.БАзовый

**class Gen<T> {**  
**T ob;**  
**public Gen(T о) {**  
**ob = о;**  
**}**  
**// Возвратить значение переменной ob.**  
**public T GetOb() {**  
**return ob;**  
**}**  
**}**  
**// Класс, производный от класса Gen.**  
**class Gen2<T> : Gen<T> {**  
**public Gen2(T o) : base(o) {**  
**// ...**  
**}**  
**}**  
**class GenHierDemo {**  
**static void Main() {**  
**Gen2<string> g2 = new Gen2<string>("Привет");**  
**Console.WriteLine(g2.GetOb());**  
**}**  
**}**

**Производный**

**// Пример необобщенного класса в качестве базового для**  
**// обобщенного производного класса.**  
**using System;**  
**// Необобщенный базовый класс,**  
**class NonGen {**  
**int num;**  
**public NonGen(int i) {**  
**num = i;**  
**}**  
**public int GetNum() {**  
**return num;**  
**}**  
**}**  
**// Обобщенный производный класс,**  
**class Gen<T> : NonGen {**  
**T ob;**  
**public Gen(T o, int i) : base(i) {**  
**ob = o;**  
**}**  
**// Возвратить значение переменной ob.**  
**public T GetOb() {**  
**return ob;**  
**}**  
**}**  
**// Создать объект класса Gen.**  
**class HierDemo3 {**  
**static void Main() {**  
**// Создать объект класса Gen с параметром типа string.**  
**Gen<String> w = new Gen<String>("Привет", 47);**  
**Console.Write(w.GetOb() + " ");**  
**Console.WriteLine(w.GetNum());**  
**}**  
**}**

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:

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.

12.Приведите пример обобщенного интерфейса.

public interface ISort<T>

where T : struct

{

void ReWrite();

}

// Реализуем интерфейс в классе MyObj

class MyObj<T> : ISort<T> where T : struct

{

public int longOb { get; set; }

T[] myarr;

public MyObj(int i)

{

longOb = i;

}

public MyObj(int i, T[] arr)

{

longOb = i;

myarr = new T[i];

for (int j = 0; j < arr.Length; j++)

myarr[j] = arr[j];

}

public void ReWrite()

{

Console.WriteLine("Тип: {0}",typeof(T));

Console.WriteLine("Массив объектов: ");

foreach (T t in myarr)

Console.Write("{0}\t",t);

Console.WriteLine("\n");

}

}13.В чем отличие обобщенных классов от обобщенных структур?Подобно классам, структуры также могут быть обобщенными. Они очень похожи на обобщенные классы, за исключением возможности наследования. В этой статье рассматривается обобщенная структура Nullable<T>, которая определена в .NET Framework. Итак, примером обобщенный структуры в .NET Framework является Nullable<T>.

Число в базе данных и число в языке программирования имеют важное отличие в своих характеристиках, поскольку число в базе данных может быть null. Число в C# не может быть null. Проблема существует не только с базами данных, но также с отображением данных XML на типы .NET.

Это отличие часто служит источником сложностей и требует массы дополнительной работы по отображению данных. Одно из решений состоит в отображении чисел из баз данных и файлов XML на ссылочные типы, потому что ссылочные типы могут иметь значение null. Однако это также означает дополнительные накладные расходы во время выполнения

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

Подобно паре Directory/DirectoryInfo для работы с файлами предназначена пара классов **File** и **FileInfo**. С их помощью мы можем создавать, удалять, перемещать файлы, получать их свойства и многое другое.

Некоторые полезные методы и свойства класса FileInfo:

* **CopyTo(path)**: копирует файл в новое место по указанному пути path
* **Create()**: создает файл
* **Delete()**: удаляет файл
* **MoveTo(destFileName)**: перемещает файл в новое место
* Свойство **Directory**: получает родительский каталог в виде объекта DirectoryInfo
* Свойство **DirectoryName**: получает полный путь к родительскому каталогу
* Свойство **Exists**: указывает, существует ли файл
* Свойство **Length**: получает размер файла
* Свойство **Extension**: получает расширение файла
* Свойство **Name**: получает имя файла
* Свойство **FullName**: получает полное имя файла

Класс File реализует похожую функциональность с помощью статических методов:

* **Copy()**: копирует файл в новое место
* **Create()**: создает файл
* **Delete()**: удаляет файл
* **Move**: перемещает файл в новое место
* **Exists(file)**: определяет, существует ли файл