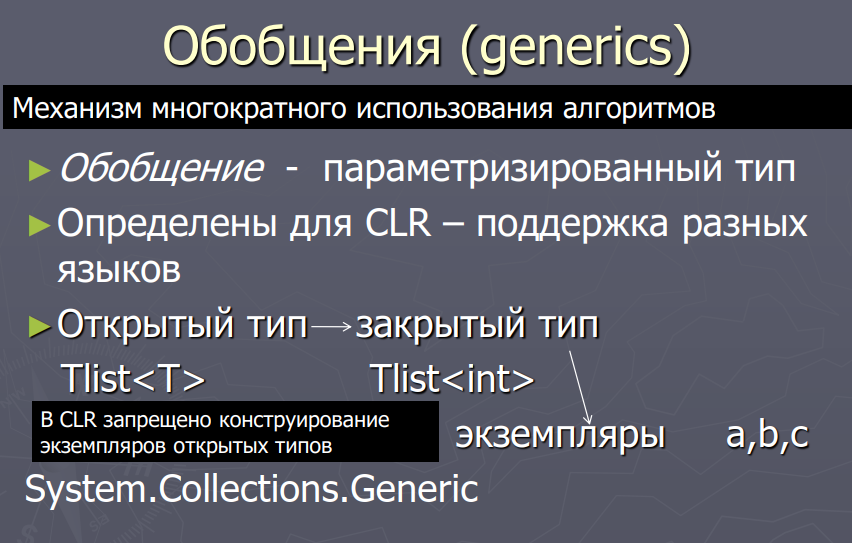
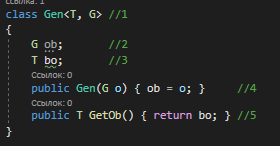
1. Что такое обобщение (generic)?

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. С помощью обобщений можно, например, создать единый класс, который автоматически становится пригодным для обработки разнотипных данных. Класс, структура, интерфейс, метод или делегат, оперирующий параметризированным типом данных, называется обобщенным, как, например, обобщенный класс или обобщенный метод.

***class имя\_класса<список\_параметров\_типа> { // ...***

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



Ошибок нет

1. Как можно наложить определенное ограничение на параметр?

Это делается с помощью **оператора where** при указании параметра типа:

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

***class Account<T> where T : struct //*** ограничение, чтобы использовались только

***{}*** структуры

***class Transaction<T> where T : class //*** ссылочные типы

***{}***

***class Transaction<T> where T : new() //*** можно задать с помощью слова ***new*** в

***{}*** качестве ограничения класс или структуру, которые имеют общедоступный конструктор без параметров

1. Как можно наложить несколько ограничений на параметр?

Указать несколько через запятую.

Если накладывать несколько ограничений на универсальный параметр, то они должны идти в определённом порядке:

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

interface IAccount { int CurrentSum { get; set; } }

class Person { public string Name { get; set; } }

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

1. Перечислите все существующие ограничения на типы данных обобщения?

Ограничение на интерфейс (ограничение сужает перечень типов, которые можно передать в обобщённом аргументе, и расширяет возможности по работе с этими типами.

Ограничение типа значения (гарантирует компилятору, что указанный аргумент будет иметь значимый тип (с поддержкой null не подходят под это)).

Ограничение на базовый класс.

Ограничение ссылочного типа.

Ограничение на конструктор.

Ограничение на связь параметров.

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

class A { }  
class В : A { }  
class С { }  
class Test<T> where T : A { }

Ограничение на базовый класс

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

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

Ограничение на интерфейс

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

interface A { }  
class Test<T> where T : struct { }

Ограничение типа значения

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

**class Gen<T> { //…// }**

**class Gen2<T, V> : Gen<T> { //…// }** // Класс, производный от класса Gen.

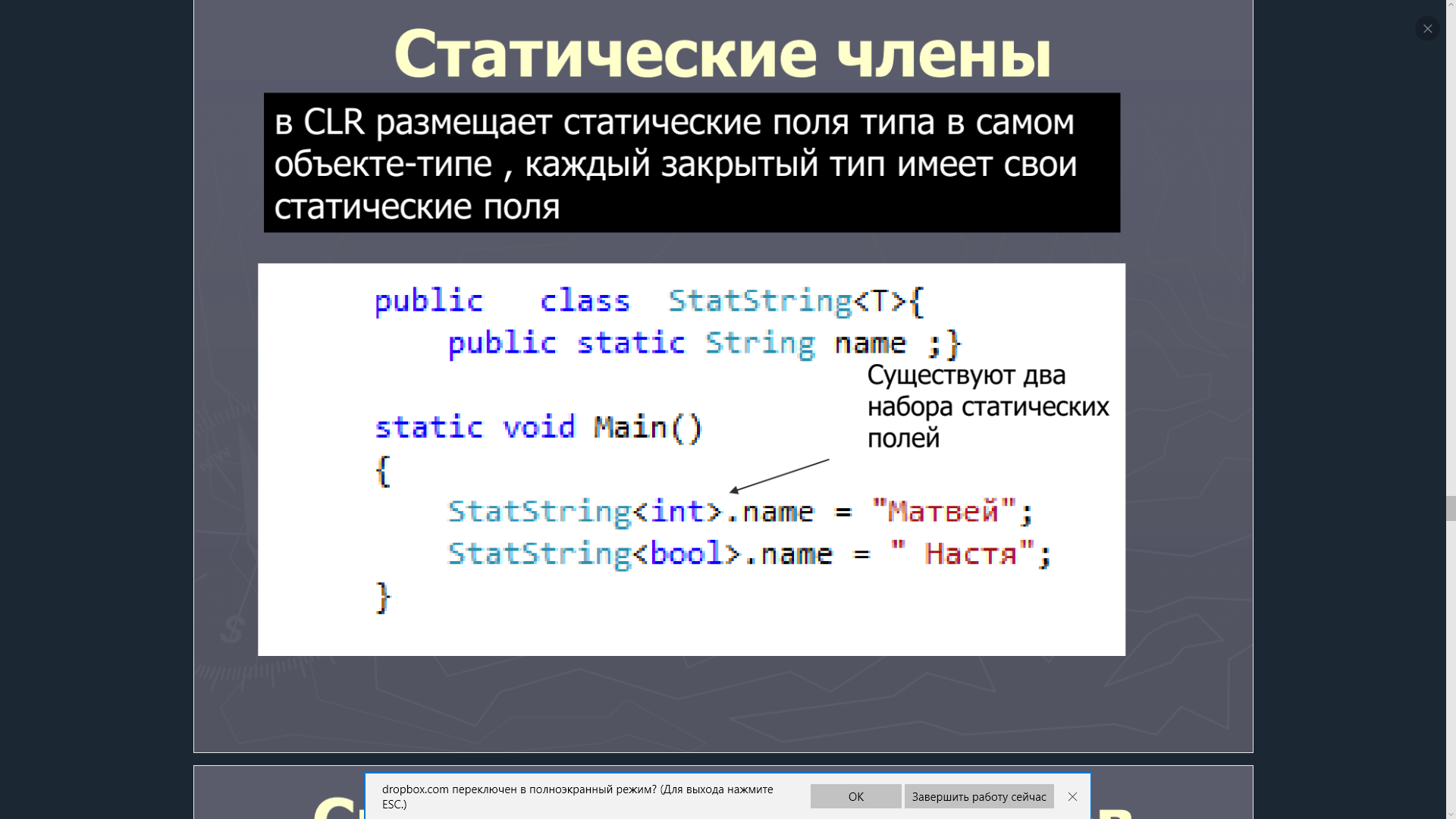
**class NonGen { //…// }**  
**class Gen<T> : NonGen { //…// } //** Обобщенный производный класс

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

При необходимости присвоить переменным универсальных параметров некоторое начальное значение, в том числе и null. **Default()** присваивает ссылочным типам в качестве значения null, а типам значений - значение 0:

**class Account<T> { T id = default(T); }**

1. Поясните как использовать статические переменные в обобщенных классах.



1. Приведите пример обобщенного интерфейса.

Применяя обобщения, можно определять интерфейсы, объявляющие методы с обобщенными параметрами.

public interface ISort<T> where T : struct

{ void ReWrite(); }

1. В чем отличие обобщенных классов от обобщенных структур?

У обобщенной структуры нет возможности наследования.

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

Методы и свойства класса **FileInfo**:

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

Класс **File** реализует похожую функциональность с помощью статических методов:

1. **Copy()**: копирует файл в новое место
2. **Create()**: создает файл
3. **Delete()**: удаляет файл
4. **Move()**: перемещает файл в новое место
5. **Exists(file)**: определяет, существует ли файл

FileStream

string path = @"C:\apache\hta.txt";

FileInfo fileInf = new FileInfo(path);

if (fileInf.Exists)

{

Console.WriteLine("Имя файла: {0}", fileInf.Name);

Console.WriteLine("Время создания: {0}", fileInf.CreationTime);

Console.WriteLine("Размер: {0}", fileInf.Length);

}