Вот ответы на ваши вопросы об обобщениях (generic) в C#:

▎1. Что такое обобщение (generic)?

Обобщение (generic) — это механизм в C#, который позволяет создавать классы, интерфейсы и методы с параметрами типов. Это позволяет писать более универсальный и повторно используемый код, так как типы могут быть определены при создании экземпляра класса или вызове метода, а не во время написания кода.

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

▎3. Как можно наложить определенное ограничение на параметр?

Ограничения на параметры обобщений задаются с помощью ключевого слова where. Например:

class Test<T> where T : A { }

Здесь T должен быть либо классом A, либо производным от него.

▎4. Как можно наложить несколько ограничений на параметр?

Несколько ограничений могут быть наложены с помощью запятой в одном выражении where. Например:

class Test<T> where T : A, new() { }

В этом случае T должен быть производным от A и иметь параметрический конструктор.

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

• Класс: where T : class — ограничивает тип ссылочными типами.

• Структура: where T : struct — ограничивает тип значимыми типами.

• Унаследование: where T : A — ограничивает тип производными от указанного класса или интерфейса.

• Параметрический конструктор: where T : new() — требует наличия параметрического конструктора без параметров.

• Интерфейс: where T : IInterface — ограничивает тип реализациями указанного интерфейса.

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

class A { }

class B : A { }

class C { }

class Test<T> where T : A { }

Ограничение на тип T — это класс A или любой его производный класс.

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

interface A { }

class Test<T> where T : class { }

Ограничение на тип T — это ссылка на класс (то есть T должен быть ссылочным типом).

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

interface A { }

class Test<T> where T : struct { }

Ограничение на тип T — это значение типа (то есть T должен быть значимым типом).

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

Обобщенные классы могут действовать как базовые или производные классы, например:

class Base<T> { }

class Derived : Base<int> { }

В этом случае Derived является производным классом от обобщенного класса Base.

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

Оператор default может использоваться для получения значения по умолчанию для типа параметра обобщения. Это может быть полезно, когда необходимо инициализировать переменные типа T, например:

T value = default(T);

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

В обобщенных классах нельзя объявлять статические поля типа параметра, поскольку они не могут иметь разные значения для разных типов. Однако можно объявить статические поля с конкретным типом:

class Test<T>

{

public static int StaticValue;

}

Здесь StaticValue будет общим для всех экземпляров класса Test<T>.

▎12. Приведите пример обобщенного интерфейса.

public interface IRepository<T>

{

void Add(T item);

T Get(int id);

}

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

Основное отличие заключается в том, что классы являются ссылочными типами, а структуры — значимыми. Это означает, что при передаче структуры в метод происходит копирование, тогда как при передаче класса передается ссылка на объект.

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

Некоторые классы для работы с файлами в .NET:

• File: предоставляет методы для создания, копирования, удаления, перемещения и открытия файлов.

• FileInfo: предоставляет методы и свойства для работы с файлами.

• StreamReader: позволяет читать текстовые данные из файла.

• StreamWriter: позволяет записывать текстовые данные в файл.

Пример использования StreamReader:

using (StreamReader reader = new StreamReader("file.txt"))

{

string line;

while ((line = reader.ReadLine()) != null)

{

Console.WriteLine(line);

}

}

1. **Параметризованный тип данных:** Шаблонные классы позволяют создавать классы, которые могут принимать один или несколько параметров типа данных.