

Понятие пространства имен

Пространства имен(namespace) —это способ, благодаря которому **.NET** избегает конфликтов имен между классами

Пространство имен



```
System.Console.WriteLine("Hello world");
```

Полная квалификация имени стереотипа, включает имя пространства имен, в котором находится стереотип

Свойства пространства имен

Пространства имен имеют следующие свойства:

- Организация крупных проектов по созданию кода.
- Для их разделения используют оператор.(точка).
- Директива **using** исключает требование на указание имени пространства имен для каждого класса, избавляя от необходимости полной квалификации имен стереотипов
- Пространство имен **global** является корневым пространством имен: **global::System** всегда будет ссылаться на пространство имен платформы **.NET Framework System**

Using

Директива `using` – импортирует пространство имен, избавляя от необходимости полной квалификации имен стереотипов

Ключевое слово `using` также используется для создания операторов `using`, которые обеспечивают правильную обработку объектов `IDisposable`, например файлов и шрифтов

Директива `using` позволяет создавать псевдонимы пространства имен или типа. Это называется директива `using alias`

```
using MyClass = NamespaceA.NamespaceB.NamespaceC.MyClassC;
```

Директива `using alias` не может иметь открытый универсальный тип с правой части. Например, невозможно создать `using alias` для `List<T>`, но можно создать для `List<int>`.

```
//using System;

class Program
{
    static void Main()
    {
        // Int32 a = 1;

        int b = 2;
    }
}
```

В случае отсутствия импорта пространства имен **System**, полные имена базовых типов будут недоступными

Доступными будут только псевдонимы типов

Ключевое слово `internal` является модификатором доступа для типов и членов типов

```
extern alias L1;

namespace Namespaces
{
    class Program
    {
        static void Main()
        {
            L1.Library.MyClass my1 = new L1.Library.MyClass();

            my1.Method();
        }
    }
}
```

Доступ к типам или членам с модификатором доступа `protected internal` может осуществляться из текущей сборки или из типов, которые являются производными от содержащего их класса

Подключение алиаса

Для подключения алиаса выполните следующие действия:

- Добавьте в [References](#) необходимые сборки.
- Откройте папку [References](#).
- Правой кнопкой мыши кликните по сборке, откроется контекстное меню, в котором выберите пункт [Properties](#).
- В открывшемся окне свойств, в свойстве [Aliases](#), замените значение [global](#) на свое название

Обзор и применение модификаторов доступа

public – доступ к типу или члену возможен из любого другого кода в той же сборке или другой сборке, ссылающейся на него.

Protected – доступ к типу или элементу можно получить только из кода в том же классе или структуре, либо в производном классе.

Internal – доступ к типу или члену возможен из любого кода в той же сборке, но не из другой сборки.

Protected internal – доступ ограничен текущей сборкой или типами, которые являются производными от содержащего класса.

Private – доступ к типу или члену можно получить только из кода в том же классе или структуре.

#region

Директива **#region** позволяет указать блок кода, который можно разворачивать и сворачивать с помощью функции структурирования в редакторе кода **Visual Studio**

#if #endif

При обнаружении компилятором C# директивы `#if` , за которой далее следует директива `#endif`, компиляция кода между двумя директивами выполняется только в том случае, если определен указанный символ

```
static void Main()
{
    #if DEBUG
        Console.WriteLine("Debug version");
    #endif

    // TODO: Посмотрите в Task List
    // HACK: Посмотрите в Task List

    Console.WriteLine("Release version");
}
```

```
static void Main()
{
    #if (DEBUG && !VC_V7)
        Console.WriteLine("DEBUG is defined");
    #elif (!DEBUG && VC_V7)
        Console.WriteLine("VC_V7 is defined");
    #elif (DEBUG && VC_V7)
        Console.WriteLine("DEBUG and VC_V7 are defined");
    #else
        Console.WriteLine("DEBUG and VC_V7 are not defined");
    #endif
}
```

Это особенно полезно при компиляции кода для построения отладки или при компиляции для определенной конфигурации