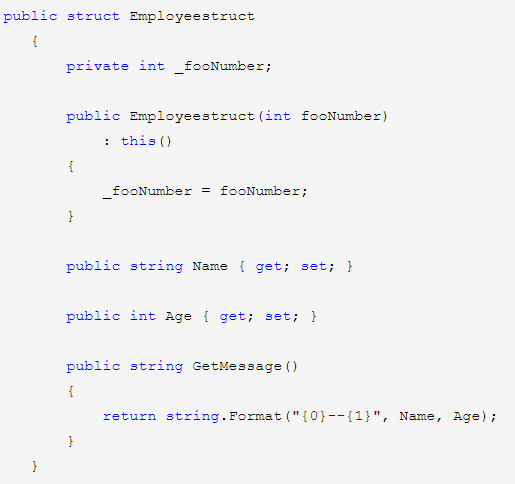
C# struct

为什么需要struct?

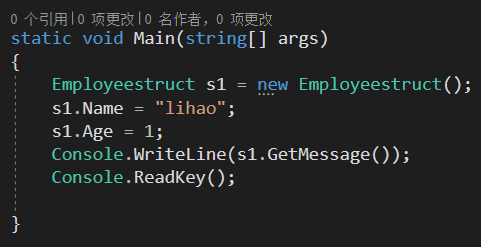
总所周知,在java中并没有struct的概念,那么C#为何引入struct呢?最基本原因它是可以创建值类型的类型,使在托管环境中有更好的性能.

区别于java,C#有值类型和引用类型的概念(java只有引用类型).引用类型的实例分配在堆上,当对象没有被引用时被GC回收;值类型的实例分配在栈上,当离开其作用域后内存被回收.值类型本身存储的是值本身,引用类型存储的是引用,C#语言提供的原始类型除了string都是值类型.

在C#中struct是值类型,class是引用类型,可以通过enum和struct关键字创建值类型的对象.**使用值类型可以减少托管堆上对象的数量,从而减少GC的负担,提高性能**;但是值类型也有明显的缺点:**通过值类型传递较大对象的开销比较昂贵、装箱和拆箱对性能造成影响**。



调用：



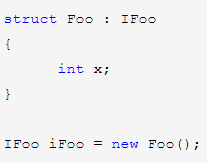
从上可以看到，class和struct非常相似，不过它们还是有本质上的区别

1. Struct和inheritance（继承）

Structs从System.ValueType继承而class继承与System.Object.

Structs可以 实现接口,但不能从另一个classes、structs继承，而且不能作为其他classes的基类。要知道，当你把structs作为接口使用时，就进行了一次隐形装箱，因为接口作为引用类型。

如下所示：



Foo的实例被创建并赋值给iFoo（此时的iFoo是IFoo类即接口，此时就应该是object类型，故发生了隐式装箱。）

1. Constructors

尽管CLR允许，但是不允许在struct中定义无参的构造函数。对于值类型，编辑器默认情况下不生产默认的构造函数，也不调用默认的构造函数，所以C#编译器**不允许**使用者定义**无参**的构造函数。由于structs不生成默认的构造函数，所以不能初始化字段：

Struct MyFoo

{

int x = 1；

}

上述写法是错误的。

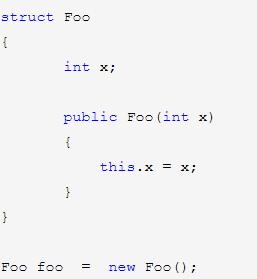
记住，编译器把初始化放在构造函数中，由于struct没有默认的构造函数，所以不能初始化字段。

有趣的一点是:

Foo foo = new Foo();

尽管struct中构造函数不能带参数，但是new Foo()还是可以这么写。

New之后的实例仍然是一个值类型，即分配到栈上。



此时，已经重载了构造函数，但是仍然能够调用new Foo()。

此时初始化x的值为默认值0。