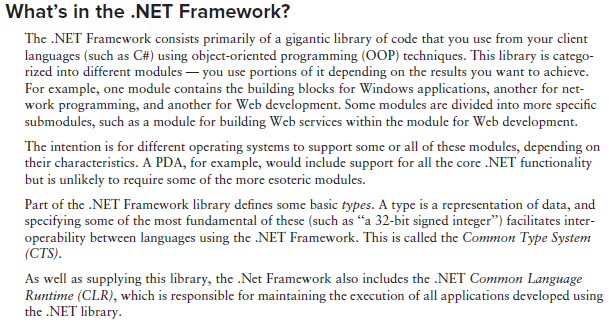
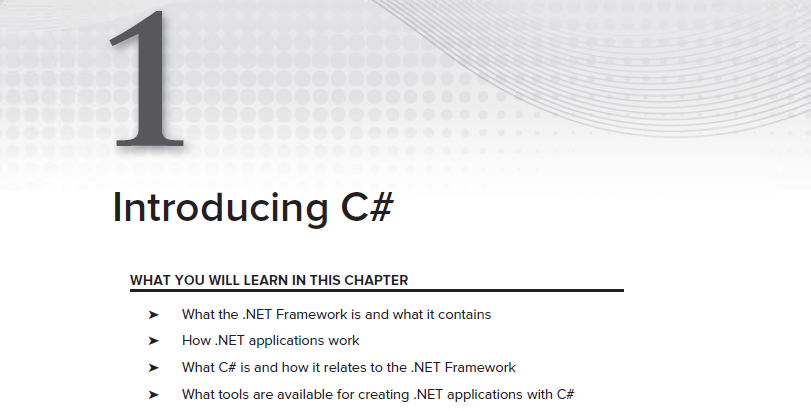
目录

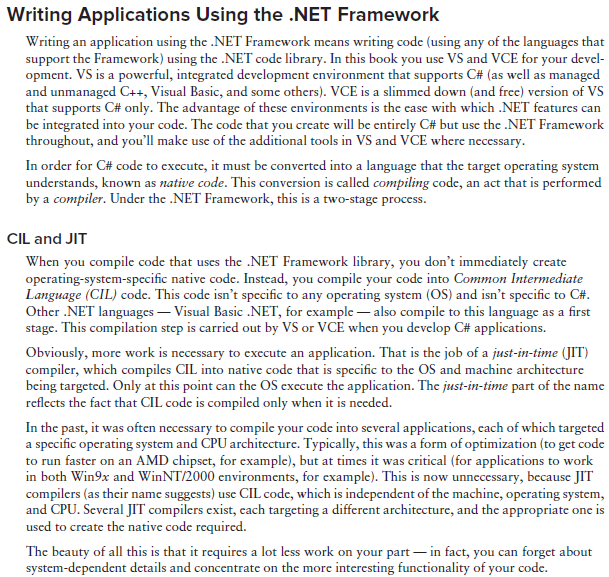
[什么是.Net框架，框架的编译运行原理 2](#_Toc424039663)

[变量的范围，常量表，verbatim，类型转换，操作符的优先级 8](#_Toc424039664)

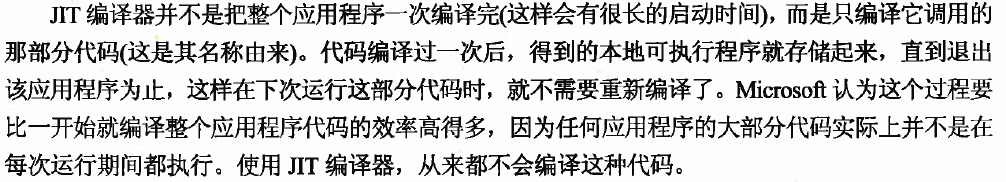
[Namespace，using，const 11](#_Toc424039665)

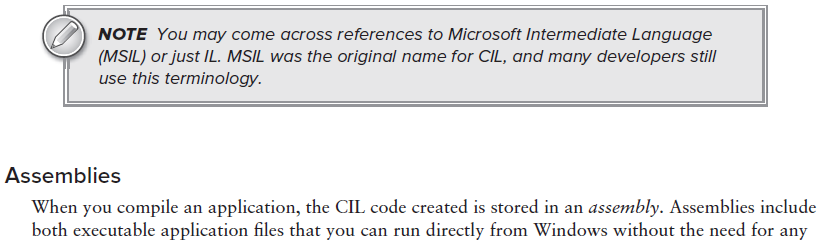
## 什么是.Net框架，框架的编译运行原理

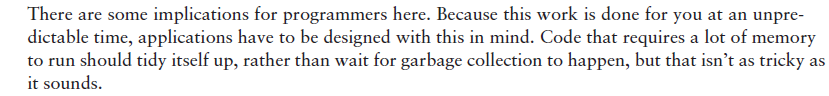
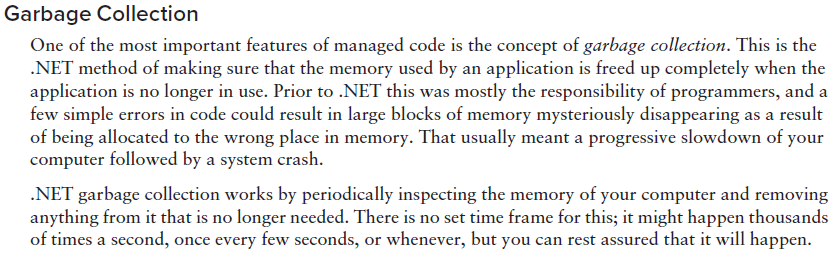
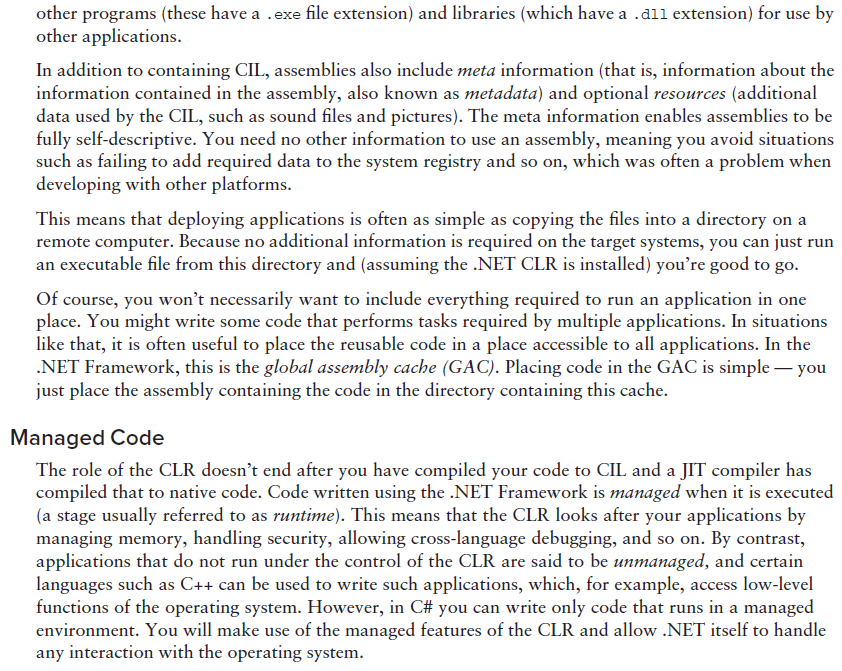


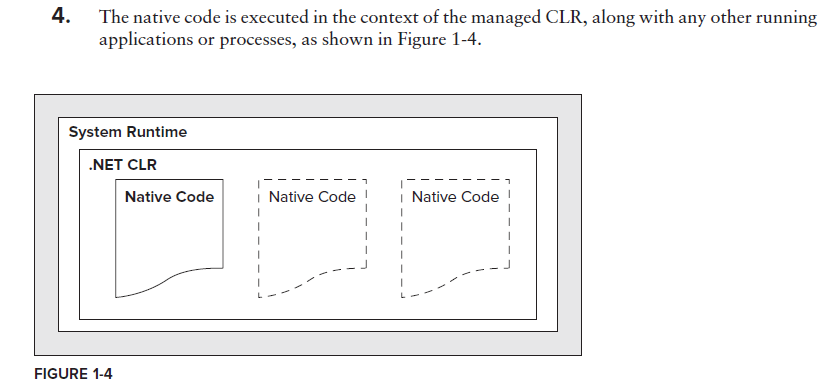
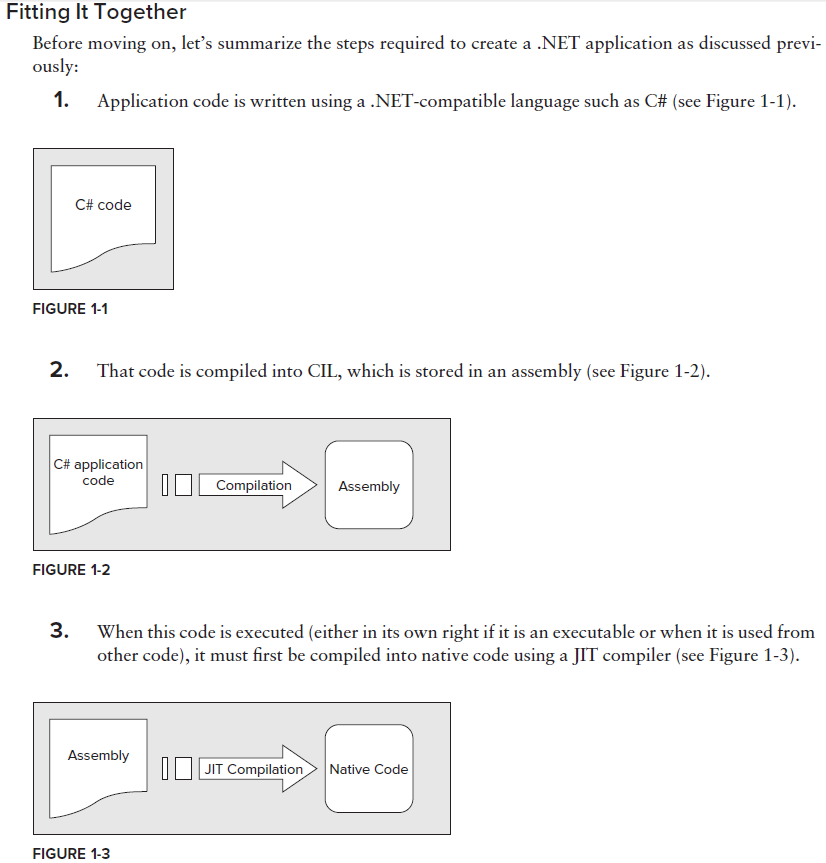


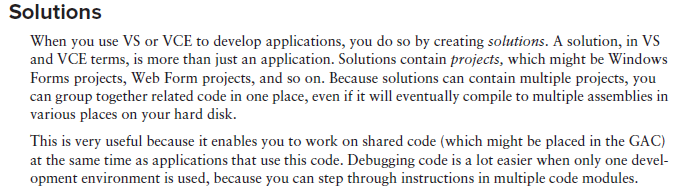
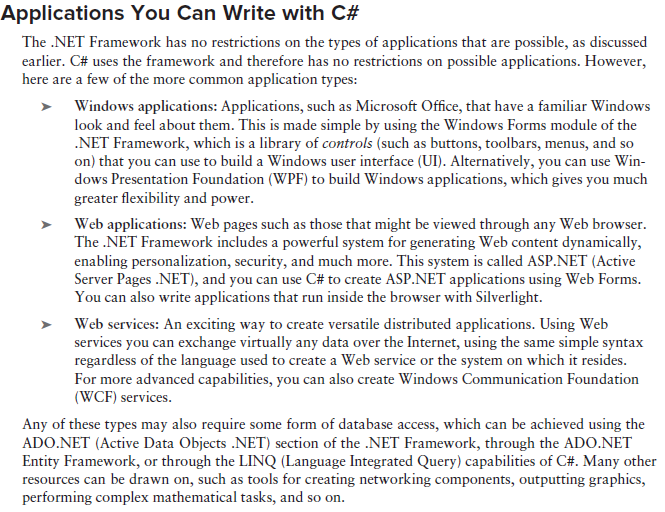
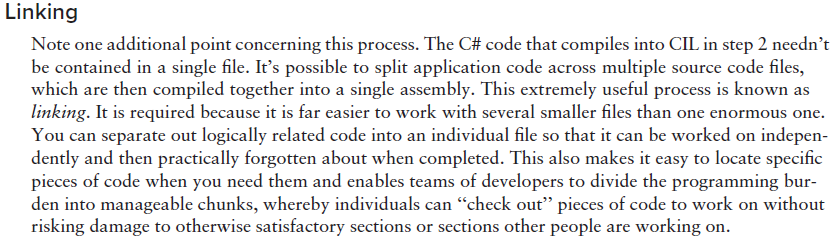
JIT在需要的时候把CIL编译成机器对应的native code，然后就可以运行



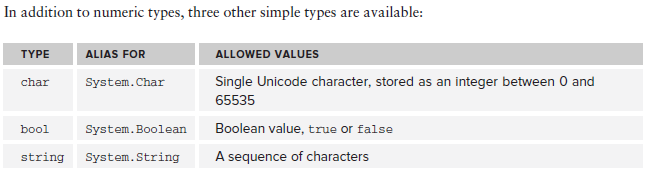
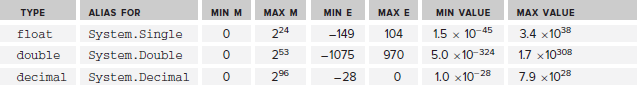
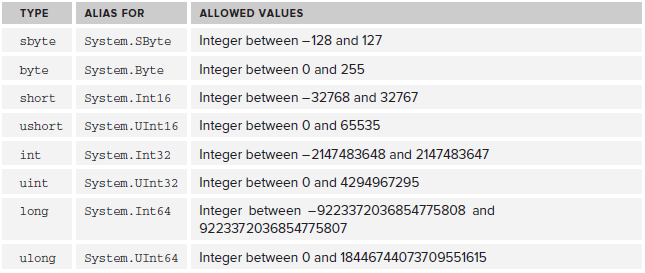




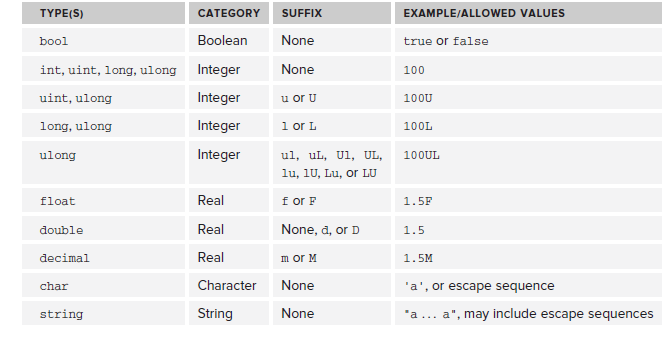




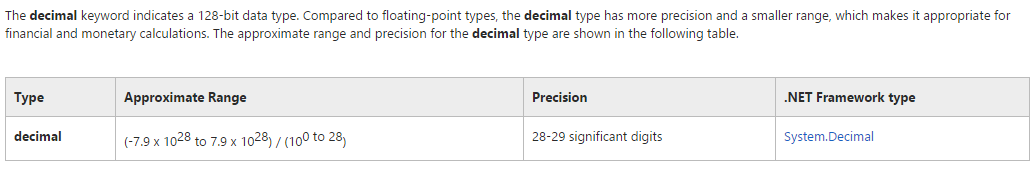
## 变量的范围，常量表，verbatim，类型转换，操作符的优先级



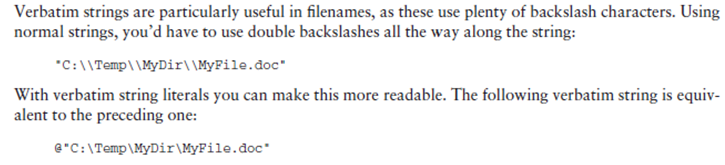
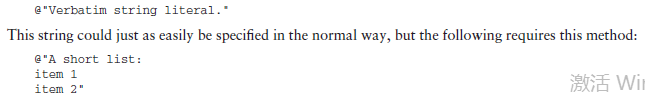
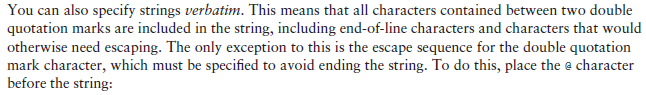
char在内存中其实是以integer的方式存储，每一个0-65535范围内的integer都有他对应的char，所以char比较大小也是比较他的integer的大小，也是因为这样char才能转型成int。



注意，带小数点的常量默认类型是double,所以float f= 1.5是错的，double不能自动转型成float，要写成float f = 1.5f

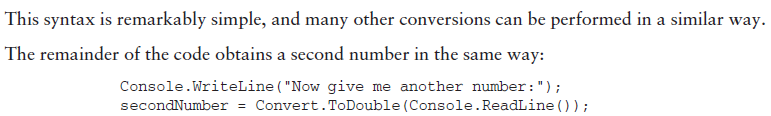


注意decimal，占128bit，他比double精度更高，适合金融或者货币这种高精度计算



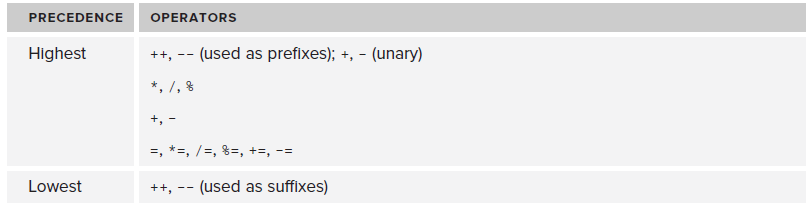
Verbatim非常有用，可以让\不被当做转义符来看待。

类型转换：



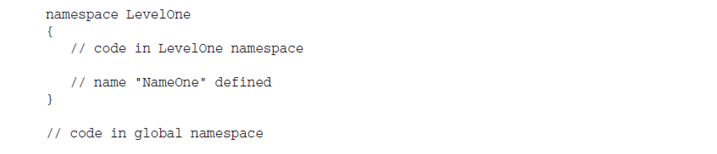
基本类型的转换和java一样，也是那幅图。

操作符的优先级

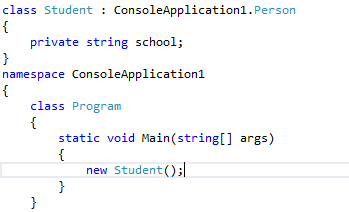


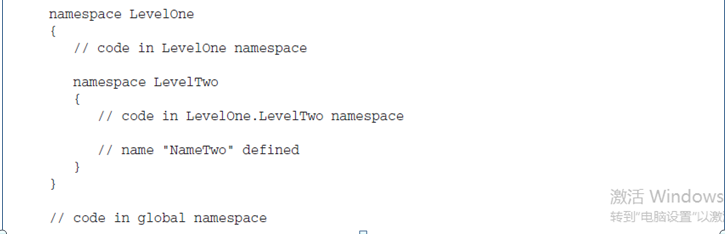
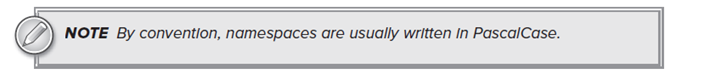
可以用括号来改变优先级

## Namespace，using，const，访问修饰符，sealed,virtual,override

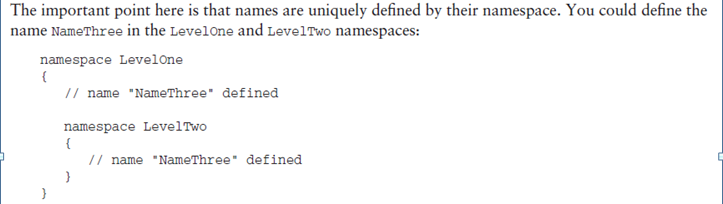


没有在namespace里定义的代码都在global namespace里

从上面程序中可以看到，global namespace里想要访问ConsoleApplication1里的类，需要加上空间名，而1里的类访问Student类就不需要

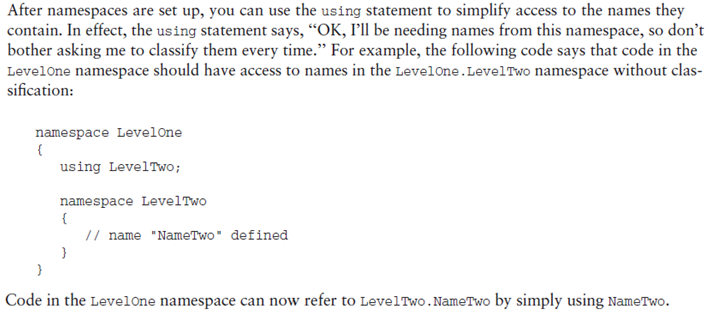


Namespace可以嵌套，在LevelOne中访问LevelTwo，要用LevelTwo.NameTwo，在global中访问则要用LevelOne.LevelTwo.NameTwo

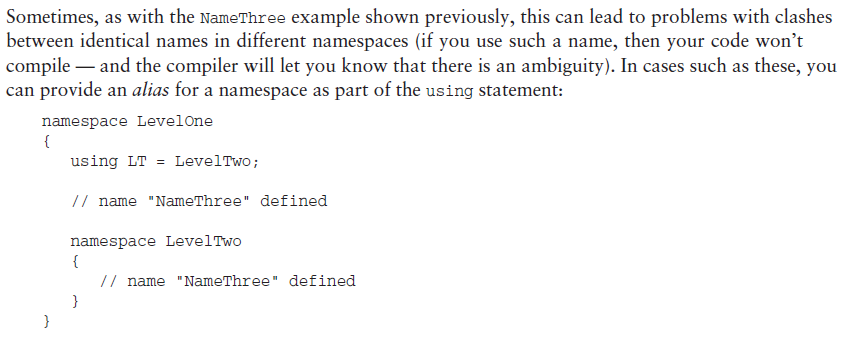


两个NameThree是不同的

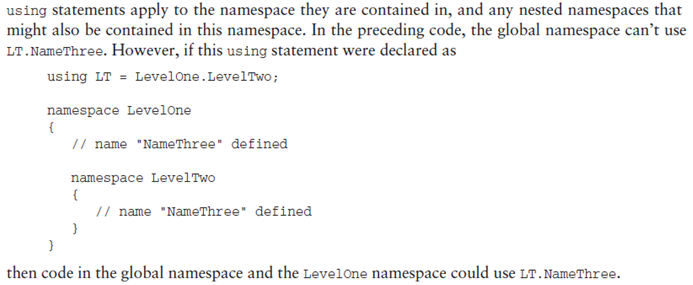
using keyword



在LevelOne里使用using LevelTwo,就可以直接在LevelOne里使用NameTwo，而不需要前缀了

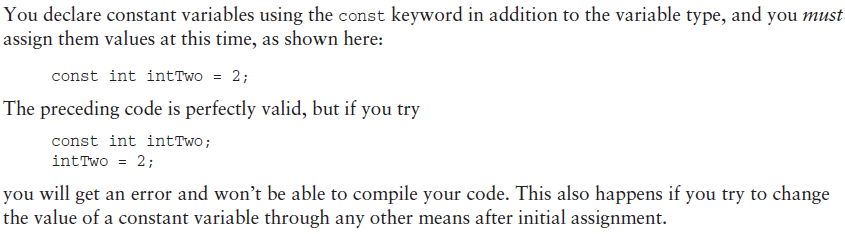


这样做会出现一个问题，看上面程序，LevelOne和LevelTwo里都定义了一个NameThree，如果在LevelOne里用了using LevelTwo，这样两个NameThree就无法区分开来，这时可以给LevelTwo取一个别名，using LT=LevelTwo

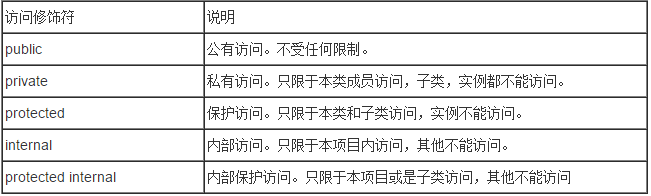


注意，using 影响当前namespace和当前namespace嵌套的namespace，比如上面代码，在global和LevelOne中均可以用LT.Name来代表LevelOne.LevelTwo.NameThree

const keyword

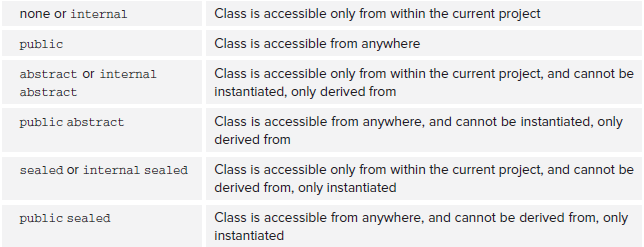


C#中没有final关键字，const类似于final关键字，但是只能变量，不能加在方法和类之前。那么怎么能规定一个类不能被继承，或者一个方法不能被重写呢？下面就会讲到，先来总结一下c#中的访问修饰符。



C#多了一个internal修饰符，类似于java中的default，不过default是包权限，而internal是项目内权限

再来看看class可以使用的前缀



这里的or代表默认情况，也就是说如果一个class不带修饰符，那他默认就是internal的，如果一个abstract class不带修饰符，那么他默认也是internal的，一个sealed class不带修饰符，那它默认也是internal的。也就是说class的默认修饰符是internal，就像java是default一样。

注意public,internal,proteced,private是权限修饰符，sealed和abstract不是，组合用的时候权限修饰符放在前面。

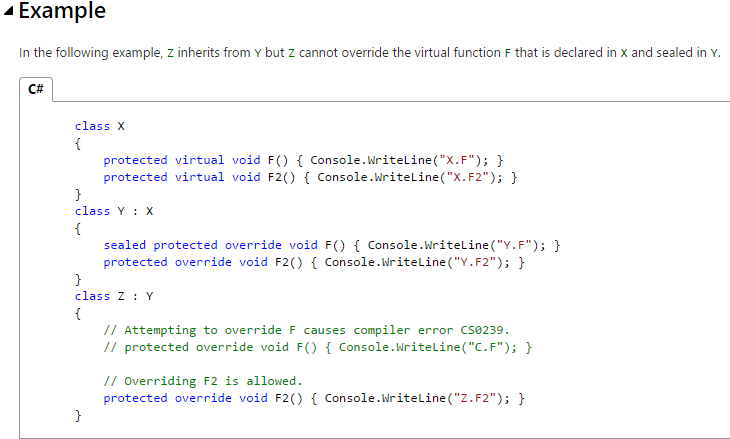
刚才说到防止class被继承，c#中是用sealed关键字，sealed关键字在方法重写中还有一个用途，稍后会讲解

virtual, override

C#中方法重写和java有显著不同，java是默认重写，c#是要显式的指明重写。在父类需要子类重写的方法必须用virtual（或者override）指定，virtual要写在返回值前面，如public virtual void….子类要重写这个方法需要用override关键字，public override void…所以你如果不想一个父类的某方法被重写，那么就不要给这个方法加virtual关键字。

C#中如果一个子类重写了父类方法，那它就用了override关键字，此时它的子类就可以再用override重写这个方法。那么如果不想让这个方法被重写下去了，该怎么办呢？这里就可以用到sealed关键字的另一个用途，看msdn里面的一段描述：

You can also use the **sealed** modifier on a method or property that overrides a virtual method or property in a base class. This enables you to allow classes to derive from your class and prevent them from overriding specific virtual methods or properties.



When you define new methods or properties in a class, you can prevent deriving classes from overriding them by not declaring them as [virtual](https://msdn.microsoft.com/en-us/library/9fkccyh4.aspx).

It is an error to use the [abstract](https://msdn.microsoft.com/en-us/library/sf985hc5.aspx) modifier with a sealed class, because an abstract class must be inherited by a class that provides an implementation of the abstract methods or properties.

When applied to a method or property, the **sealed** modifier must always be used with [override](https://msdn.microsoft.com/en-us/library/ebca9ah3.aspx).（虽然例子中sealed写在方法声明的最前面，但其实也可以写在protected的后面，推荐写在后面，这样和sealed用在class中的位置相同）

Because structs are implicitly sealed, they cannot be inherited.