**公有继承(public)**

公有继承的特点是基类的公有成员和保护成员作为派生类的成员时，它们都保持原有的状态，而基类的私有成员仍然是私有的，不能被这个派生类的子类所访问。

**私有继承(private)**

------------------默认的继承方式（如果缺省，默认为private继承)

私有继承的特点是基类的公有成员和保护成员都作为派生类的私有成员，并且不能被这个派生类的子类所访问。

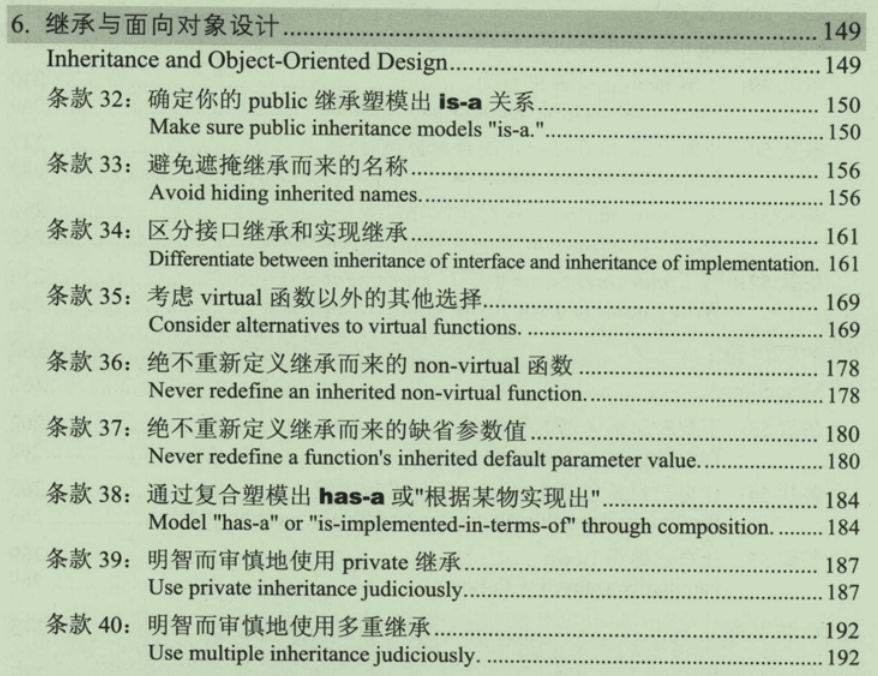
子类也不能转换成相应的基类，如果转换，会报错：“不允许对不可访问的基类进行转换”。

**(3) 保护继承(protected)**

保护继承的特点是基类的所有公有成员和保护成员都成为派生类的保护成员，并且只能被它的派生类成员函数或友元访问，基类的私有成员仍然是私有的。

**总结**

c++面向对象都讲究一个封装性，所以不可能继承私有成员，不然哪有封装性可言，所以继承针对public和protect的成员，如果是那个继承就放到继承类的那个继承方式的模块中，private继承就放到继承类的private中



**条例32 看到public继承想到is-a**

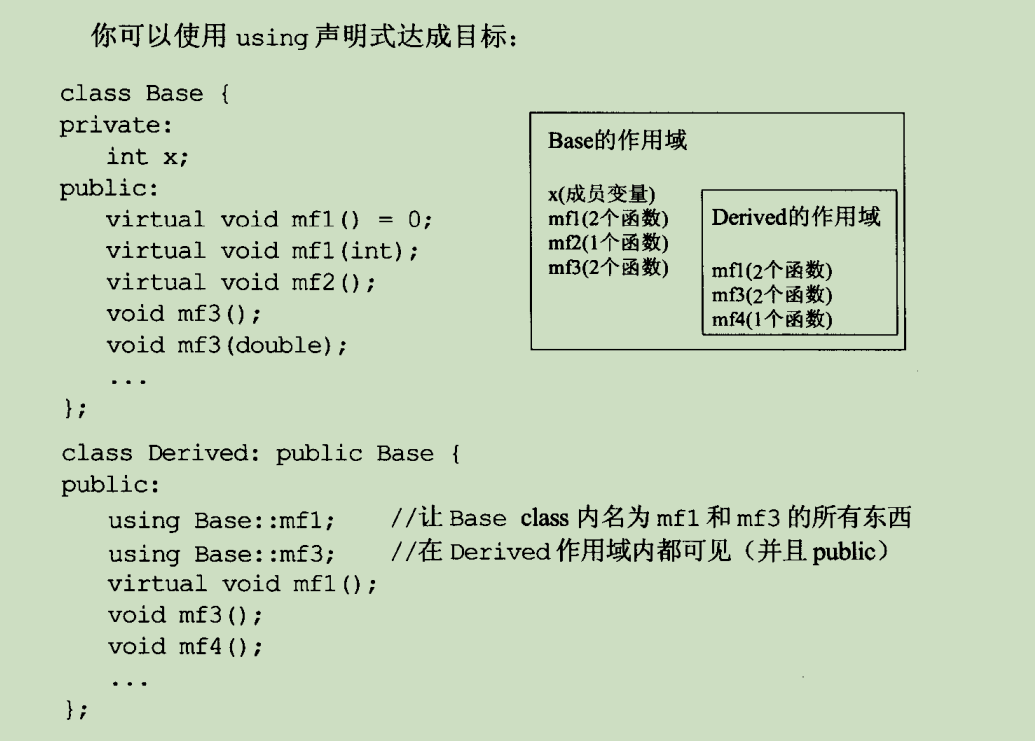
**is-a的内涵**

is-a代表，继承类是基类的特例，所以继承类包含基类public所有的部分，缺少一个函数，也就是基类具备的特点，继承类没有，那不就是打破了is-a 特例的性质。

**is-a注意事项**

is-a 代表public继承关系，继承类只是子类的特例。基类是鸟，继承类是企鹅，所以鸟可以飞，所以企鹅也可以飞 ，这种public继承就出现问题，为了区分，应该写成两个类，会飞的鸟，和企鹅，所以以后继承的时候，考虑到企鹅和鸟的关系。

**条款33**



**图片的继承关系**

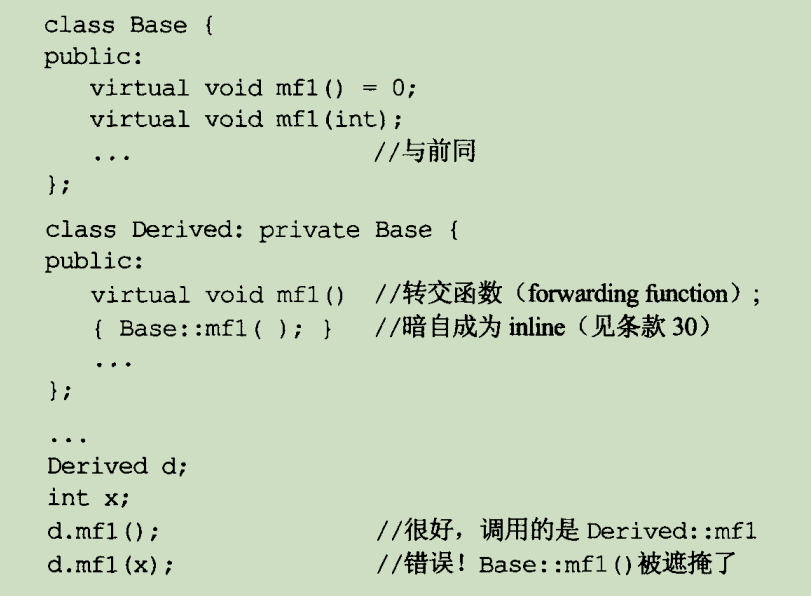
derived的作用域在base作用域类

**名字查找打破is-a关系**

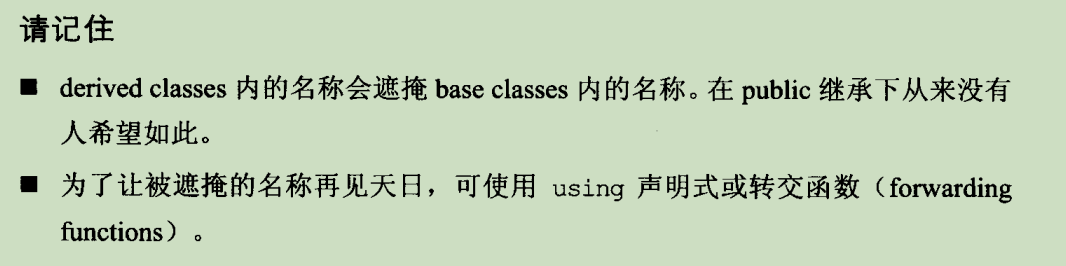
我们清楚发现derived重新实现了mf1和mf3，如果不书写 using Base：：mf1；岂不是就意味着，我们的derived类中看不到 base中的mf1函数，这种c++自带的名字查找进制，实际上变相的打破了is-a的继承精神。

**using声明带来了的问题**

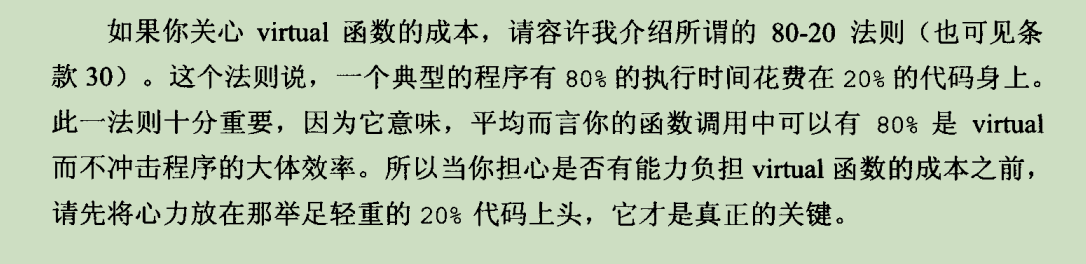
using声明一个base类的函数，意味着他在继承类中，都可以看到这个函数，如果是私有继承就会带来一点问题，如何在public看到这个base函数，下面的转换可以参考，套个皮而已。



、



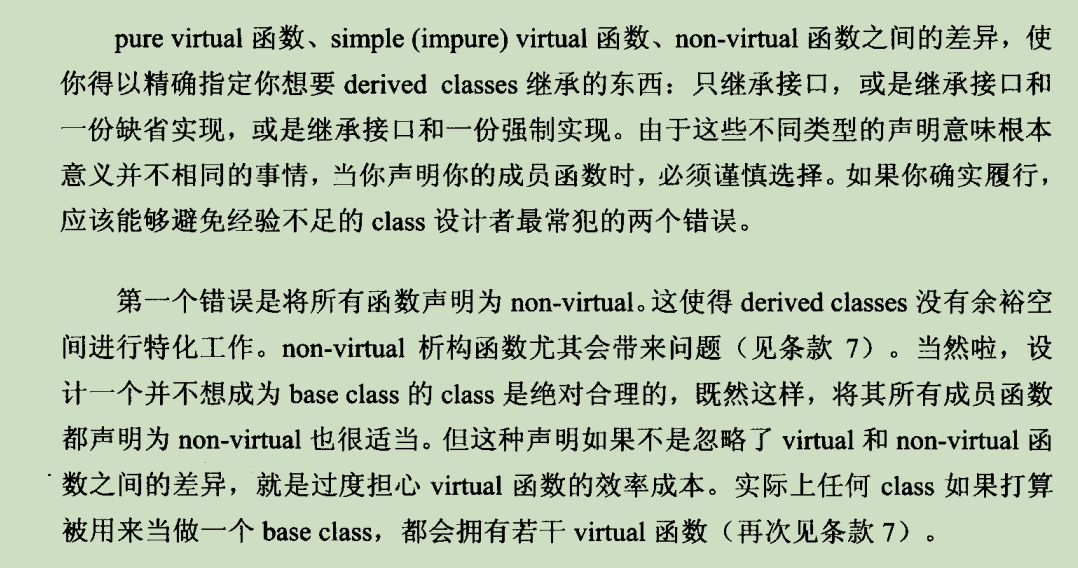
**条款34 基类函数的不同声明，以此提高性能**

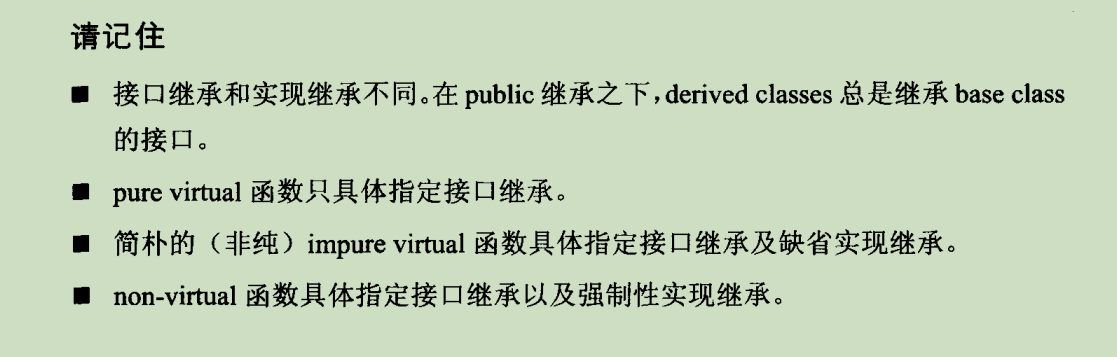


pure virtual 代表你可以使用这个接口或者不使用（具体就体现在继承类中是否实现这个函数）

impure virtual 函数 代表你必须使用这个接口，你可以决定是否使用基类的版本或者自己实现（缺省）

non virtual 函数 不可更改，强制继承



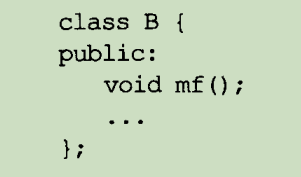


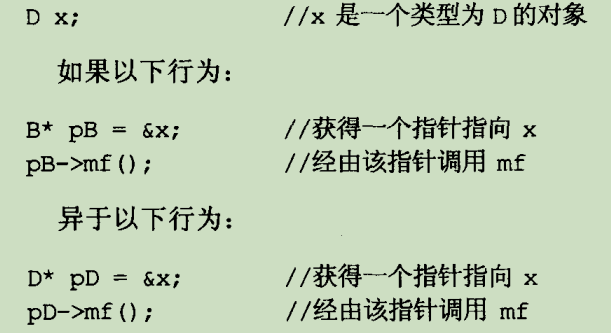
**条款35**

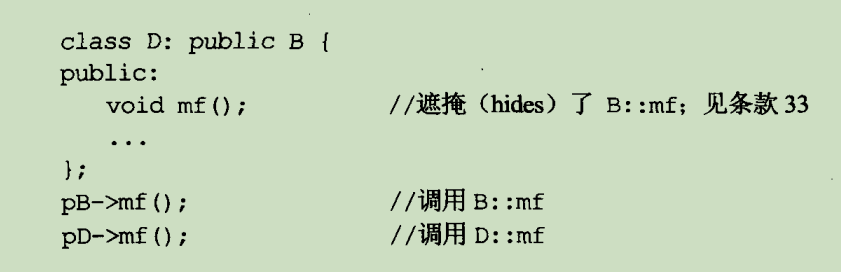
设计模式有点难以理解

**条款36**

non -virtual 是静态绑定





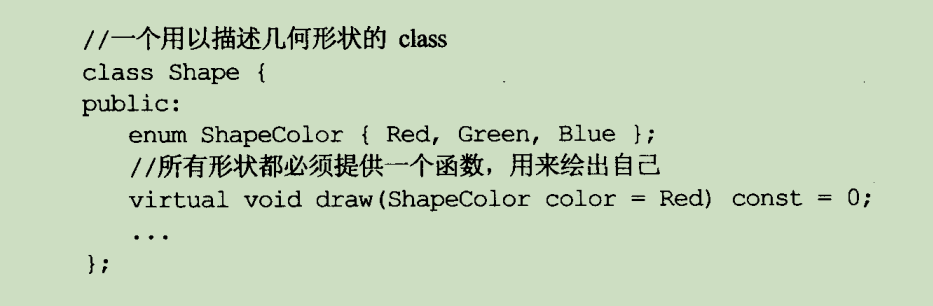


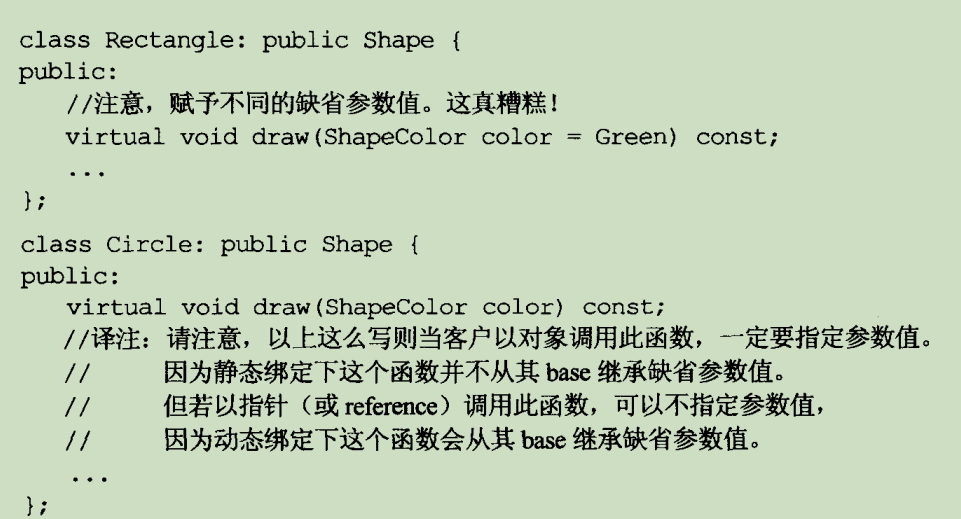
这个时候你可能回好奇 pB指向class D，为什么会调用B::mf呢，因为只有virtual才是动态绑定，所以多态在这里没有，如果你在继承类中重新non -virtual 函数，违背了 is-a的原则

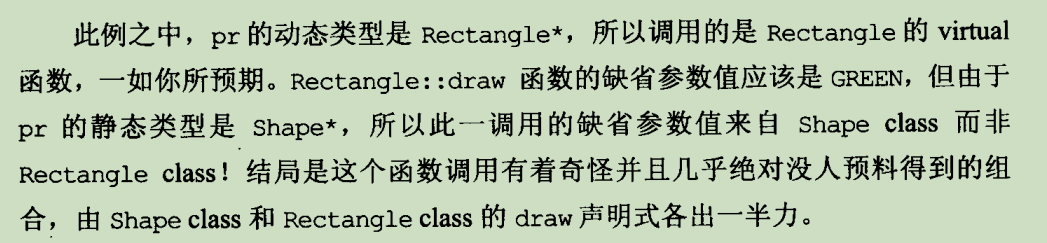
条款37

一旦完成了静态绑定，动态绑定不会改变他。

对于那种给与了默认参数的virtual 函数调用需要小心，因为你使用base 类指针指向继承类，这个base类的函数默认参数是静态绑定，所以这个base类调用这个函数，由于他的静态类型是base类，他就会使用这个静态绑定参数

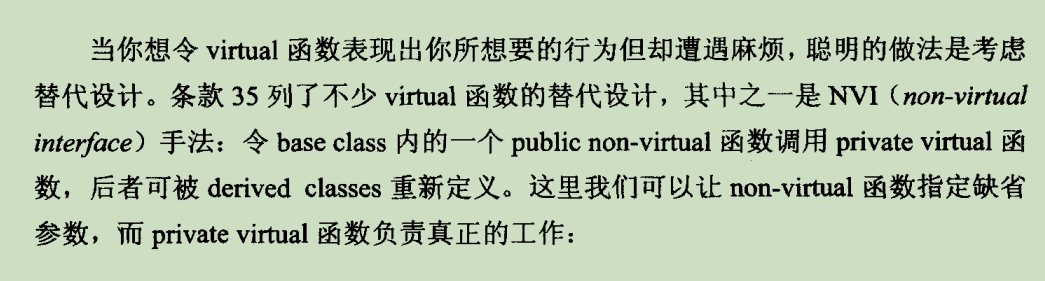




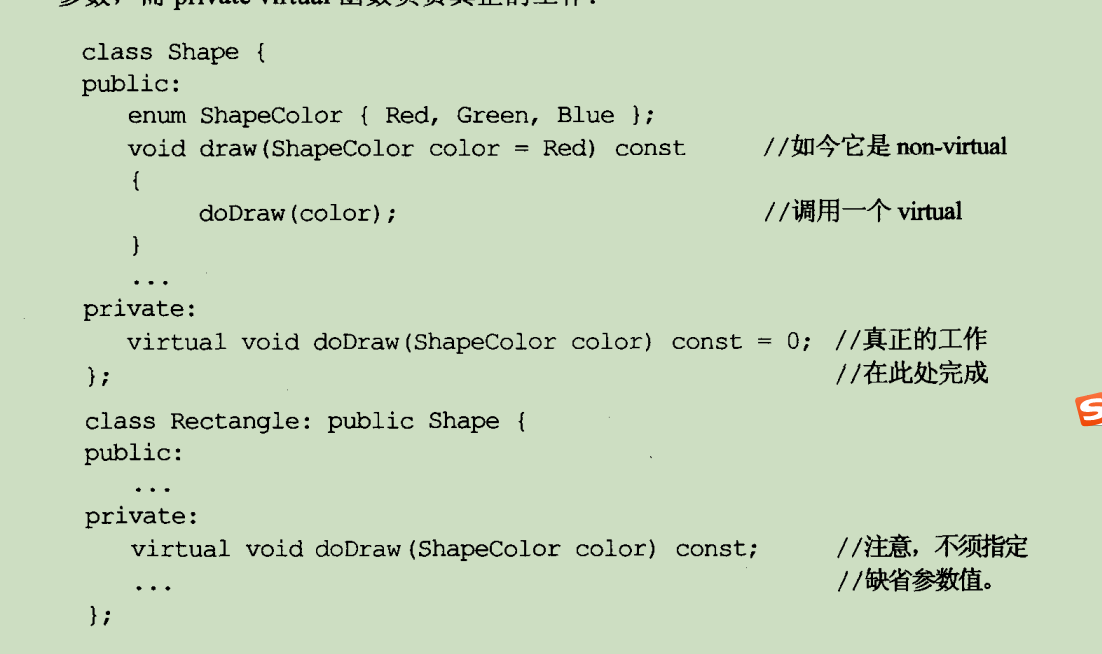


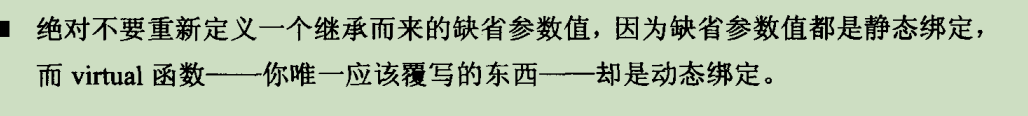
解决方案

你不想出现上面的静态绑定，



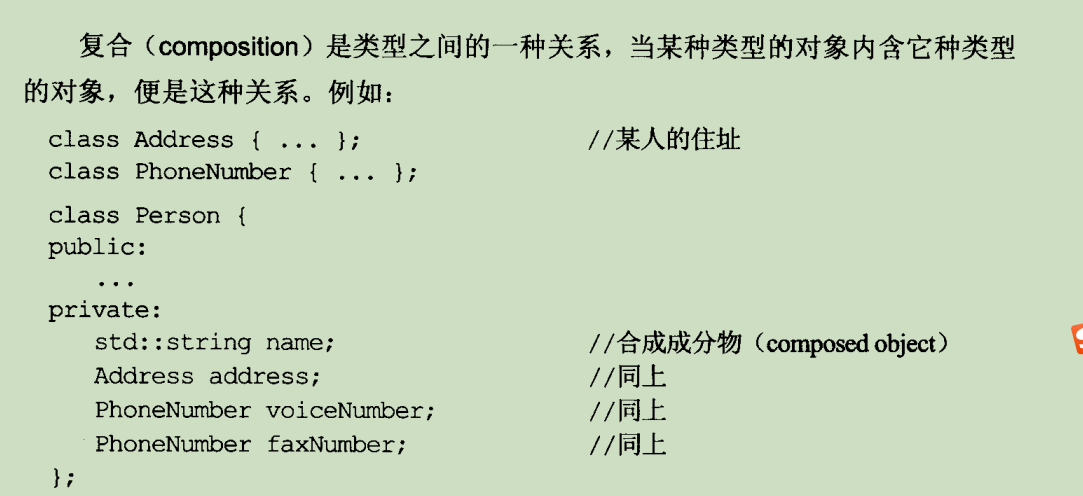
这样解决了缺省参数问题





**条款38**

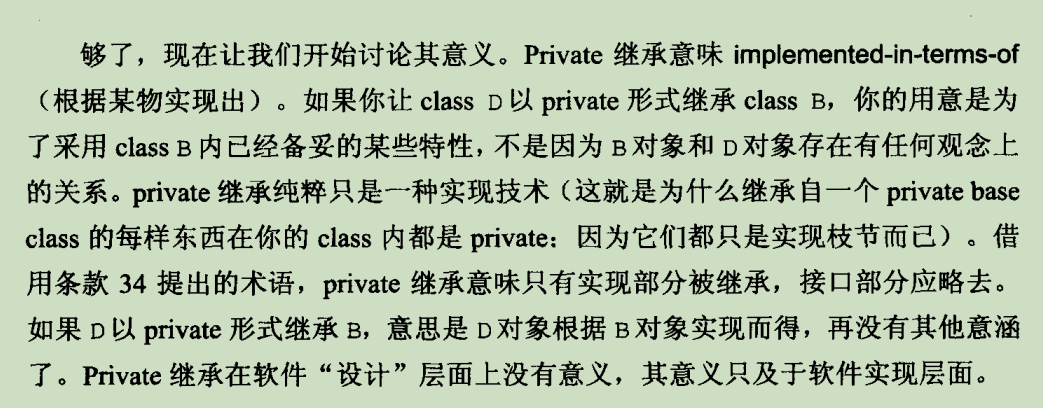
**复合是has-a**



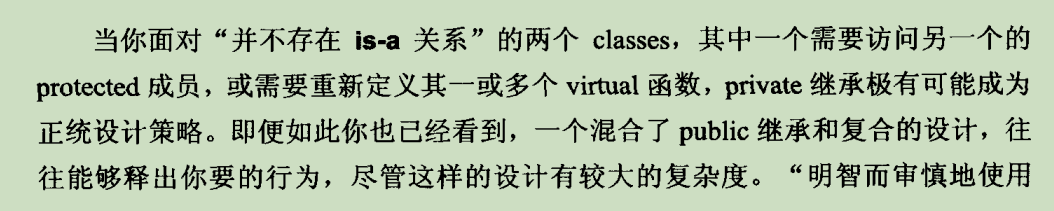
复合这种只是一种has-a的关系，比如人可以拥有电话号码，而不是人是一个电话号码，所以public的继承对于这种复合思路，是绝对的滥用，public是is-a的关系。

**条款39**

private跟复合的关系很像，因为base类的函数都在继承类的private中，所以如果我们想要通过实现多态就不现实了，无法调用继承类private的函数。所以private继承，子类不会像基类转换了。

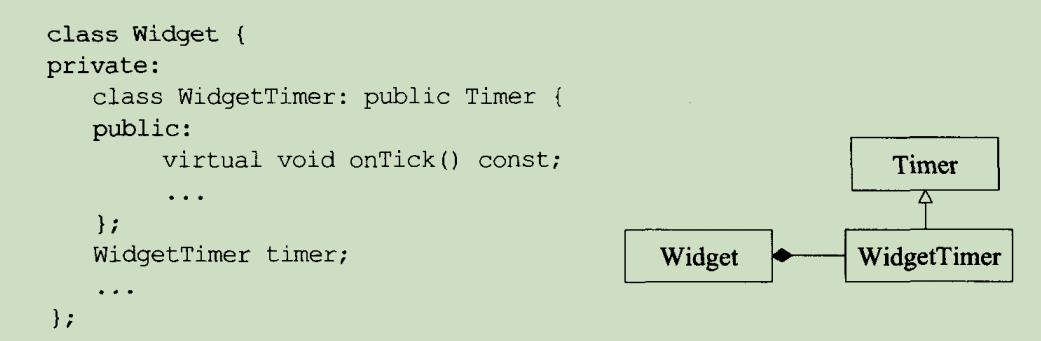


如何看待复合和private继承。



**private设计模式**

private继承的优势

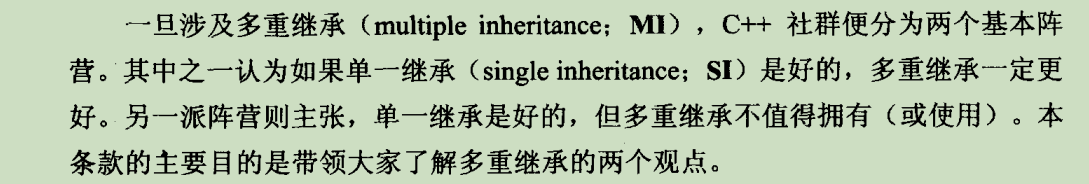


**EBO（c++存储思想）**

一个空类如果复合，那么他一定会产生大小，多数是编译器为了进行8和12,16这种空间补齐，如果是单向继承，可以做到不产生大小。、、

**条款40**

能不用多重继承，就不用，时间慢，效率低，内存大



**问题**

