lua如何注册c++类

开始前需要明确几个概念：

1. c++中的指针变量在lua中是userdata（如果不需要lua替你管理回收则是lightuserdata），而指针变量的含义是指向某个内存地址，地址就是一个值，所以指针变量本质上和一个int变量是没有区别的。
2. lua中的数据类型只有普通类型（number、bool、string、function、thread、nil、userdata）+table，只有table才能有成员（这个成员可以是普通类型或者table），table可以直接访问成员，如果成员不存在，则去看table的元表是否有\_\_index方法去执行。

number、bool、string、function、thread、nil这几个类型共享一个元表，并且我们无法去改变其共享的这个元表（除非我们去改lua源码），而userdata和table可以有自己的元表。

综合1、2两点可以得出以下结论：c++的指针(lua中的userdata)是一个很特殊的值，它没有成员，但是它有元表，如果访问它的成员，那么它就会去执行它元表的\_\_index方法，如果没找到就报错。

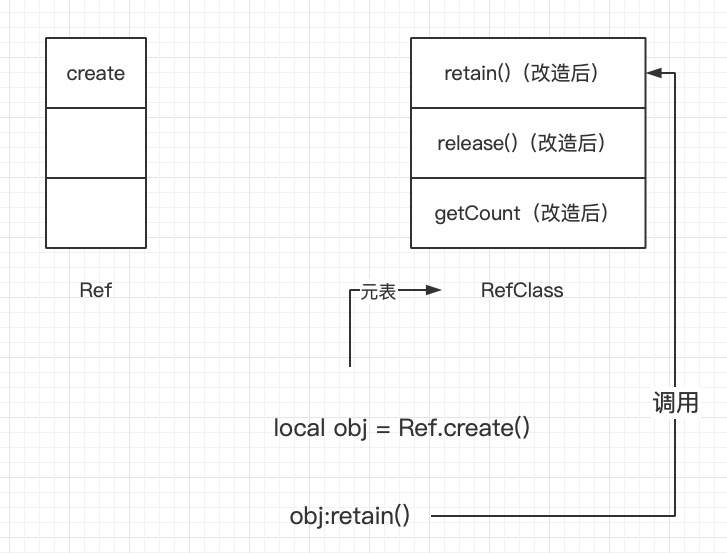
因此我们所谓的lua中通过对某个对象以 obj:method 这样的形式调用该对象的方法时，本质上并不是直接执行我们在c++中的obj->method，而是lua先去看obj是否有method，没有则去找它的元表，调用元表的\_\_index方法。而\_\_index除了可以赋值为方法还可以赋值为表，如果将\_\_index赋值为元表自身，那么就会取在元表中找obj所要访问的方法(key)对应的value给obj，而元表中对应的value的注册形式就是模块名+方法的形式。

因此核心点就来了，如果我们要注册一个c++的类到lua中使用，那么就要分以下几步进行：

1. 为要注册的类的create方法（也就是创建这个类型对象的方法）提供一个全局接口（可以通过 模块名+方法的形式注册）
2. 设置一个全局元表，将类的所有方法改造后注册到全局元表中。
3. create方法里为每个新创建的对象设置元表为2所创建的全局元表。

示意图如下图所示：

比如要注册一个名叫Ref的类到lua，其中将设置的元表叫RefClass



通过Ref.create()创建一个Ref类型的对象，然后对该对象调用retain()方法，则该方法回去访问元表的retain()方法，由于冒号调用语法糖会将自身作为参数传入，因此只需在RefClass改造后的这个函数中从lua栈栈顶获取到对象，然后对该对象执行真正意义上对应的方法即可。

以模块名+方法形式注册与这样以面向对象形式注册的对比：

1. 以模块名+方法形式注册的调用对象方法与我们在c++中面向对象的调用方法不一致。
2. 以模块名+方法形式注册传入userdata即可，并未对该userdata的类型检测，如果我传其它类型的userdata对象进入，再强转换为目标类去调用方法，将可能造成访问错误内存地址。而以面向对象形式注册由于注册了元表，在lua中是可以通过lua\_checkudata方法去检测传入的对象是否为目标类型（实际上判断的是元表是否为之前设置的那个名字的元表）。