版本号	描述	日期(更新日期)/更新人
1.0	Android Studio Lua技术	2019/08/13 **刘志保**
Github地址	https://github.com/MMLoveMeMM/AngryPandaLua	
	java工程参考上面工程根目录下LuaJava.zip工程	

Lua语言在学习过程中,看起来是支持面向对象编程的,但是实际用起来根本不好使,所以感觉Lua的面向对象编程就是"半桶水",为什么这么说呢?Lua的类与类之间可以继承,但是仅仅是继承方法和变量,但是缺少"父与子"之间的家属关系,子类无法调用父类的方法,也就是说父类方法要想使用,子类需要自己重写一次,即使和父类方法内容一样,因为在创建子类的时候,父类是不进行创建,也不会运行构造函数的,这样就导致了子类可以使用的时候,父类由于没有构造,导致父类方法无法使用,即没有父类方法.

具体的测试样例就不再介绍了,经过前面的不断学习,应该可以说太简单了.

下面主要介绍如何解决Lua面向对象的问题,这个问题也是网上提出的解决方案,我自己经过反复测试和运用,的确没问题.

```
--
-- Created by IntelliJ IDEA.
-- User: zhibao.liu
-- Date: 2019/7/31
-- Time: 19:42
-- To change this template use File | Settings | File Templates.
--
local _class={}

 function class(super)
  local class_type={}
  class_type.ctor=false
  class_type.super=super
  class_type.new=function(...)
  local obj={}
  do
  local create
  create = function(c,...)
  if c.super then
  create(c.super,...)
```

```
 end
 if c.ctor then
 c.ctor(obj,...)
 end
 end

 create(class_type,...)
 end
 setmetatable(obj,{ __index=_class[class_type] })
 return obj
 end
 local vtbl={}
 _class[class_type]=vtbl

 setmetatable(class_type,{__newindex=
 function(t,k,v)
 vtbl[k]=v
 end
 })

 if super then
 setmetatable(vtbl,{__index=
 function(t,k)
 local ret=_class[super][k]
 vtbl[k]=ret
 return ret
 end
 })
 end

 return class_type
end

```

利用这个基类主要是为了完成super关键字的功能,父类在此基类的基础上进行构建,然后子类再进行父类构建即可.

下面创建一个父类:

```
--
-- Created by IntelliJ IDEA.
-- User: zhibao.liu
```

```
-- Date: 2019/7/31
-- Time: 19:43
-- To change this template use File | Settings | File Templates.
--
require "base"
base_type=class() -- 定义一个基类 base_type

 function base_type:ctor(x) -- 定义 base_type 的构造函数
  print("base_type ctor")
  self.x=x
 end

 function base_type:print_x() -- 定义一个成员函数 base_type:print_x
  print(self.x)
 end

 function base_type:print_show(info) -- 定义一个成员函数 base_type:print_x
  print(info)
 end

 function base_type:print_show(info1,info2) -- 定义一个成员函数 base_type:p
rint_x
  print(info2)
 end

 function base_type:hello() -- 定义另一个成员函数 base_type:hello
  print("hello base_type")
 end

```

然后再看看子类,并且子类创建继承于上面的父类:

```
--
-- Created by IntelliJ IDEA.
-- User: zhibao.liu
-- Date: 2019/7/31
-- Time: 19:43
-- To change this template use File | Settings | File Templates.
--
require "base_type"
test=class(base_type) -- 定义一个类 test 继承于 base_type
```

```
10
11 function test:ctor() -- 定义 test 的构造函数
12  print("test ctor")
13 end
14
15 function test:print_show(dd)
16  print(dd)
17 end
18 function test:hello() -- 重载 base_type:hello 为 test:hello
19  self:print_show("liuzhibao self method")
20  print("hello test")
21 end
22
```

使用如下:

```
1 acc=test.new(); -- 输出两行，base_type ctor 和 test ctor 。这个对象被正确的构造了。
2 acc:print_show("this is sub class print show");
3 acc:print_show("show one","show two");
```

注意new()前面是点号".",不是冒号":"!

Lua类的方法调用是采用":"冒号作为连接符,类(或者类对象)的变量调用采用点号"."连接.

Lua类和模块对比:

Lua类	Lua模块
类(类对象)调用方法采用冒号":"	模块调用模块的方法采用点号"."
类(类对象)调用变量采用点号"."	模块调用模块变量采用点号"."
	模块中可以开发全部函数/方法,在调用时不需要指定模块名,参见下面的例子
	self关键字指本身,类似于this,但是上面的super是没有用的

```
1 libmodule={}
2
3 libmodule.var = 100
4
5 function libmodule.max(n1,n2)
6  print("libmodule.max :",n1,n2)
7  if(n1>n2) then
```

```
 return n1;
 else
  return n2;
  end

end

function hellomodule()
  print("this is hello module !")
end

return libmodule
```

上面两个方法,一个是libmodule.max(n1,n2)方法,一个是hellomodule()方法,这个方法时全局方法,在其他文件可以直接调用.
综上述,类和模块的调用方式显而易见,模块最后一行还要标上:

```
return libmodule
```

上面那个基类还是有点错误,因为在子类中不能够调用父类的方法.更新基类如下:

```
--
-- Created by IntelliJ IDEA.
-- User: zhibao.liu
-- Date: 2019/7/31
-- Time: 19:42
-- To change this template use File | Settings | File Templates.
--
local _class={}

function class(super)
  local class_type={}
  class_type.ctor=false
  class_type.super=super
  class_type.new=function(...)
  local obj={}
  do
  local create
  create = function(c,...)
  if c.super then
  create(c.super,...)
```

```
  end
  if c.ctor then
  c.ctor(obj,...)
  end
  end

  create(class_type,...)
  end
  setmetatable(obj,{ __index=_class[class_type] })
  return obj
  end
  local vtbl={}
  _class[class_type]=vtbl

  setmetatable(class_type,{__newindex=
  function(t,k,v)
  vtbl[k]=v
  end
  })

  if super then
  setmetatable(vtbl,{__index=
  function(t,k)
  local ret=_class[super][k]
  vtbl[k]=ret
  return ret
  end
  })
  end

  return class_type
end

```

上面的子类不变,这样调用父类的方法可以如下:

```
self:super("print_x","heloo");
```

这样放到子类中调用即可.