# 课堂笔记

## 伴生对象

如何实现同个类既有普通方法又有静态方法 ？ 同Java里的需求。

Scala里没有static关键词，Scala取消static是一种进取的表现，这样得以保证了其面向对象模型的完整性。

有时候，我们还是需要类一级的属性和方法的。在Scala里，这便是伴生对象（Companion Object）的作用。

下面就是一个伴生对象的例子， CompanionObject.scala 的代码如下：

**class** CompanionObject()

{

**def** fun()

{

println("this is common function")

}

}

**object** CompanionObject

{

**def** staticFun()

{

println("this is static function")

}

}

这样fun() 就是普通方法，staticFun() 就是静态方法。

这个object就是我们所说的伴生对象，你会觉得这个伴生对象和Singleton异曲同工，实际上，伴生对象本身就是一个Singleton，不同的是，它有一个与之同名的类（这里的class Companion），二者可以相互访问彼此的私有成员。

我们用javap查看反编译的结果：

F:\wk\_sp\MyScala\bin\com\cloudy\scala\lesson8> javap CompanionObject.class

Compiled from "CompanionObject.scala"

public class com.cloudy.scala.lesson8.CompanionObject {

public static void staticFun();

public void fun();

public com.cloudy.scala.lesson8.CompanionObject();

}

F:\wk\_sp\MyScala\bin\com\cloudy\scala\lesson8> javap CompanionObject$.class

Compiled from "CompanionObject.scala"

public final class com.cloudy.scala.lesson8.CompanionObject$ {

public static final com.cloudy.scala.lesson8.CompanionObject$ MODULE$;

public static {};

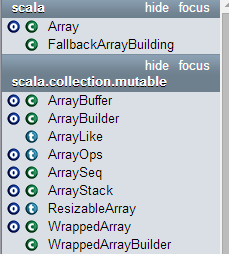
public void staticFun();

}

因为有了对应的class，object成了伴生object。从结果可以看出，伴生对象和它对应的类在字节码层面走到了一起（Companion类）。换句话说，在Scala里面的class和object在Java层面里面合二为一，class里面的成员成了实例成员，object成员成了static成员。

## 伴生对象的apply方法用途

Api文档里：



O表示伴生对象，C是class，t表示trait。

查看**HashMap的Api文档**

**自主开发一个支持如下实例化的类：**

**var** tt = Test("100")

如何升级为单例模式的？