# Scala基础知识

## 10. Scala单例对象、伴生对象

### 10.1 Scala单例对象

Scala中没有java中的静态类、静态方法、静态成员，相应的scala中提供了Object对象，在Object对象中所有的成员和方法都是静态的。

package lesson10  
  
/\*\*  
 \* Created by hipparchus on 2017/2/9.  
 \*/  
  
class College{  
 val id = College.studentNo  
 private var number = 0  
 def addClass(number:Int){this.number += number}  
}  
  
object College{  
 private var studentNo = 0  
 def newStudentNo = {  
 studentNo += 1  
 studentNo  
 }  
}  
  
object ObjectOps {  
 def main(args: Array[String]) {  
 println(College.newStudentNo)  
 println(College.newStudentNo)  
 }  
}

在以上代码中，Object College就是一个Object对象，它的方法可以通过*对象名.方法名*的方式来调用，但是需要注意的是在main方法中调用的Object对象的方法或成员不能使用private关键字修饰。

输出结果：

1

2

由于Object对象内的方法和成员都是静态的，所以Object对象可以作为静态变量区，存放一些公共的变量和方法。例如一些配置信息需要在Object对象中定义。

与java的静态类不同的是，java的静态类只要加载静态类的时候会对静态类的方法执行，但是scala中的Object对象只用在第一次调用时才会执行其内部的方法和成员，而不是每次加载时都执行。其实Object对象自身也存在构造器，在上例中，Object College存在一个无参数的构造器，在第一次调用时会调用构造器来创建Object College对象。

### 10.2 类的伴生对象

如果存在一个类与一个Object对象名字相同，则称这个object对象是这个类的伴生对象，这个类是这个对象的伴生类。

class College{  
 val id = College.studentNo  
 private var number = 0  
 def addClass(number:Int){this.number += number}  
}  
  
object College{  
 private var studentNo = 0  
 def newStudentNo = {  
 studentNo += 1  
 studentNo  
 }  
}

在伴生对象中会对类的一些静态配置和一些隐式转换进行的封装。伴生类可以访问伴生对象的所有成员（包括私有成员与非私有成员），例如上例中class College可以访问伴生对象的私有成员 studentNo。值得一提的是伴生对象并不是在伴生类的作用范围内的，所以访问伴生类的成员时需要使用*对象名.成员名*的方式。

## 11.scala中的apply方法

在Scala中apply方法有两种用法一种是在Object伴生对象中调用apply方法，另一种是在类中调用apply方法,前者使用比较多。

### 11.1 Object中的apply

在Object中使用apply方法大多数是用于构造对象，利用半生对象的apply方法来实现伴生类的对象的实例化。

package lesson11

/\*\*

\* Created by hipparchus on 2017/2/9.

\* apply function

\*/

class ApplyTest{

def apply() = println("I am into Spark!")

def haveATry: Unit ={

println("Have a try on apply!")

}

}

object ApplyTest{

def apply() = {

println("I am into Scala!")

new ApplyTest

}

}

object ApplyOperation {

def main(args: Array[String]) {

val arr = Array(1,2,3,4,5)

val a = ApplyTest()

a.haveATry

}

}

这里有一个伴生类ApplyTest和其伴生对象object ApplyTest，并且中伴生对象中实现了apply方法返回一个ApplyTest的对象。

当在main方法中通过*val a =ApplyTest()* 方式创建对象时，scala会调用object ApplyTest 的apply方法，来创建对象。

输出：

I am into Scala!

Have a try on apply!

在构建集合时，一般都是采用这种方式来构建集合对象的，例如创建数组对象：

val arr = Array(1,2,3,4,5)

实际上是调用了Array类的半生对象的apply方法。使用这种方式有如下几个好处：1)进行单例控制，2)可以控制对象构建的权限控制，3）避免多次使用new来创建对象，4）通过在伴生类apply方法中构建抽象类的子类实例的方式来实例化抽象类。

### 11.2 Class中的apply方法

这种方式使用的比较少。

class ApplyTest{

def apply() = println("I am into Spark!")

def haveATry: Unit ={

println("Have a try on apply!")

}

}

object ApplyOperation {

def main(args: Array[String]) {

val a = new ApplyTest

a.haveATry

println(a())

}

}

一般调用class的apply的方法是通过*对象名()*方式来调用的。

结果：

Have a try on apply!

I am into Spark!

()