* Extending a Class

在scala中你继承一个class，与java中相似，用extends关键字：

class Employee **extends** Person {  
 var salary = 0.0  
 ...  
}

和java中类似，你可以在子类中指定新的field和method，也可以重写父类的方法。你也可以声明class为final，这样他就不能被继承。也可以将method和field声明为final，这样他们就不能被重写。

NOTE：与java不同的是，java中final field是不可变的，类似scala中的val

* Overriding Methods

在scala中你必须使用override关键字来重写一个非abstract方法（如果重写的方法是abstract，可以省略override）

DEMO：

class Person {  
 ...  
 **override** def toString = s"${getClass.getName}[name=$name]"//override不能省略  
}

override关键字可以在一下情况下给出很多有用的错误信息，比如：

1. 当你需要重写method的名字写错的时候
2. 当你不小心把参数类型写错的时候
3. 当父类方法与子类方法冲突的时候

调用父类的方法和java一样，用super关键字：

class Employee extends Person {  
 ...  
 override def toString = s"${**super.**toString}[salary=$salary]"  
}

super.toString调用父类的toString方法，即Person.toString

* Type Checks and Casts

想检验一个实例是否属于某个class，可以用isInstanceOf方法。如果检验成功，你可以用asInstanceOf方法将引用转换。

if (p.isInstanceOf[Employee]) {  
 val s = p.asInstanceOf[Employee] // s has type Employee  
 ...  
}

如果p为指向class Employee或者它子类的实例（例如 Manager），p.isInstanceOf[Employee]返回true。如果p为null，p.isInstanceOf[Employee]返回false，p.asInstanceOf[Employee]返回null。

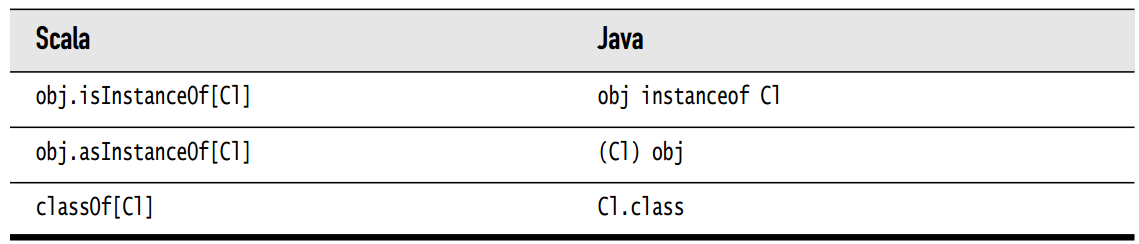
如果p不是一个Employee，那么p.asInstanceOf[Employee]会抛出异常。

如果你想检验p指向的是Employee的实例，而不是它的子类，可以用：

if (p.getClass == classOf[Employee])

classOf方法定义在scala.Predef object中，始终被导入。

下表显示了Scala和Java类型检查和强制转换之间的对应关系：



但是，模式匹配通常是使用类型检查和强制转换的更好选择，后面我会专门讲解scala中的模式匹配，例如：

p match {  
 case s: Employee => ... // Process s as an Employee  
 case \_ => ... // p wasn’t an Employee  
}