链滴

四、组合与继承

作者：shiweichn
原文链接：https://ld246.com/article/1533372860118
来源网站：链滴

四、组合与继承

标签（空格分隔）： Scala学习笔记

---

1. 抽象类

```
abstract class Domain {
  def contents: Array[String]
}
```

上面第二行是个抽象方法，只有方法的声明，没有实现。与Java不同的是，在Java中，定义抽象方法须要使用 abstract 关键字。但是在Scala中，这样的方法就被认为是抽象方法。有抽象方法的类一定被定义为抽象类。

2. 定义无参数方法

```
abstract class Domain {
  def contents: Array[String]

  def height: Int = contents.length

  def width: Int = if (height == 0) 0 else contents(0).length
}
```

Domain 类中的三个方法都没有参数列表，甚至连空列表都没有'def width():Int'。这种无参数方法在cala中非常普遍。带有空括号的方法定义，被称为空括号方法(empty-paren method) 。推荐的惯例无论何时，只要方法中没有参数并且方法仅能通过读取所含对象的属性去访问可变状态（特指方法不改变可变状态），就使用无参数方法。

这个惯例支持统一访问原则 ，就是说客户代码不应由属性是通过字段实现还是方法实现而受到影响。如，我们可以把width 和 height 作为字段而不是方法来实现，只用把def改成val即可。

```
abstract class Domain {
  def contents: Array[String]

  val height: Int = contents.length

  val width: Int = if (height == 0) 0 else contents(0).length
}
```

两组的定义从客户的观点来看是完全相同的。唯一的差别是访问字段比调用方法略快，因为字段在类初始化的时候被预计算，而方法调用在每次调用的时候都要计算。另一个方面，使用字段要为每Domain对象分配更多的内存空间。因此一个属性是用字段表示更好还是用方法表示更好与类的使用况有关，并且类的使用情况可能会随着时间变化。

对于空括号，原则上，Scala的函数调用中可以省略所有的空括号。然而，在调用的方法超出其用者对象的属性时，推荐仍然写一对空的括号。也就是说无论何时当调用有副作用的方法时，应该确包含一对空括号，如:println()。永远不要定义没有括号的带副作用的方法，因为那样看上去像是在使字段，所以你的客户会很奇怪为什么它带有副作用。如果仅提供了对某个对象的访问，那么就省略括
。

3. 扩展类

Scala中的这个地方跟Java中基本一样。继承任然使用extends关键字，只不过在Java中任何类的顶级类都是Object，而Scala中是AnyRef

一些继承特性跟Java差不多，比如，私有不能被继承等等。

4. 重写方法和字段

统一访问原则只是Scala在对待字段和方法上比Java更统一的一个方法。另一个差异是Scala里的字段方法属于相同的命名空间。这使得字段可以重写无参数方法。

Scala禁止在同一个类中用同样的名称定义字段和方法，尽管Java允许这么做。

与Java为定义准备了四个命名空间(字段、方法、类型、包)相对，一般来说，Scala只有两个命名空间：

- 值（字段、方法、包、单例对象）
- 类型（类和特质名）

Scala把字段和方法放进同一个命名空间的理由很明确，因为这样就可以实现使用val重写无参函数。

5. 定义参数化字段

```
class Domain(param1: Array[String]) {
  val contents: Array[String] = param1

  val height: Int = contents.length

  val width: Int = if (height == 0) 0 else contents(0).length
}
```

Domain 类现在有一个可以从类外部访问的(不能重新赋值的)字段 contents。字段使用参数值初始化。

6. 调用超类构造器

```
class SubDomain(s: String) extends Domain(Array(s)) {

}
```

7. 使用 Override 修饰符

Scala要求，若子类重新了父类的具体成员则必须带有这个修饰符；若子类实现的是同名的抽象成员，则这个修饰符是可选的。

8. 多态和动态绑定

多态这个和Java中的没啥区别。编译时就确定了类型的是静态绑定，运行时才确定类型的是动态绑定。

9. 定义final成员

final 关键字可以保证成员不被重写，类不被继承。

10. 组合和继承

组合和继承是利用现存类定义新类的两个方法。如果追求的是根本上的代码重用，那么通常更推荐使组合而不是继承。只有继承才受累于脆基类问题，因为可能会在无意中更改超类时破坏子类。