链滴

二、Scala 类与对象

作者：shiweichn
原文链接：https://ld246.com/article/1533372604123
来源网站：链滴

二、Scala类与对象

标签（空格分隔）： Scala学习笔记

```
/**
  * Created by Sweeney on 2017/10/26.
  */
class CheckSumAccumulator {
  private var sum = 0;

  def add(b: Byte): Unit = {
    sum += b
  }

  def checksum(): Int = {
    return ~(sum & 0xFF) + 1
  }

}
```

Scala中定义类跟在Java中定义类区别不是很大，只是Scala中成员默认的访问级别就是public，也就说定义公有属性不用加public关键字。但是定义私有属性任然使用private。

CheckSumAccumulator 类中定义了两个方法，Scala里方法参数的一个重要特性就是她们都是val。果想在方法里给参数重新赋值，就会报错。

CheckSumAccumulator 中的方法虽然没有问题，但是代码不够简洁。如果方法中没有发现任何显的返回语句，Scala方法将返回方法中最后一次计算得到的值。方法的推荐风格是尽量避免使用返回句。

所以可以改造成：

```
/**
  * Created by Sweeney on 2017/10/26.
  */
class CheckSumAccumulator {
  private var sum = 0;

  def add(b: Byte): Unit = sum += b

  def checksum(): Int = ~(sum & 0xFF) + 1
}
```

对于类中的 add 方法那样的结果类型为 Unit 的方法来说，执行的目的就是为了它的副作用。**通常我定义副作用为能够改变方法之外的某处状态或执行I/O活动的方法。**所以像 add 这样仅仅为了副作用执行的方法有另一种表达方式：去掉结果类型和等号，把方法放到花括号内。例如：

```
/**
  * Created by Sweeney on 2017/10/26.
  */
class CheckSumAccumulator {
  private var sum = 0;
```

```
  def add(b: Byte) {
    sum += b
  }
  def checksum(): Int = ~(sum & 0xFF) + 1
}
```

Singleton 对象

```
/**
  * Created by Sweeney on 2017/10/26.
  */
class CheckSumAccumulator {
  private var sum = 0;

  def add(b: Byte) {
    sum += b
  }

  def checksum(): Int = ~(sum & 0xFF) + 1
}

object CheckSumAccumulator {

  import collection.mutable.Map

  private val cache = Map[String, Int]()

  def calculate(s: String): Int = {
    if (cache.contains(s))
      cache(s)
    else {
      val acc = new CheckSumAccumulator
      for (c <- s)
        acc.add(c.toByte)
      val cs = acc.checksum()
      cache += (s -> cs)
      cs
    }
  }
}
```

Scala中不能定义静态成员，而是代之以定义单例对象，除了用 object 关键字替代了 class 关键字以，单例对象的定义与类定义一致。

当单例对象与某个类共享同一个名称时，它就被称为这个类的伴生对象，类和它的伴生对象必须义在同一个文件里。类被称为是这个单例对象的伴生类，类和它的伴生对象可以互相访问其私有成员。

定义单例对象并没有定义类型，如果只有单例对象的定义，就不能定义单例对象的变量。因为类是由单例对象的伴生类定义的。然而单例对象扩展了父类并可以混入特质。因此，可以使用类型调用例对象的方法，或者用类型的实例变量指代单例对象，并把它传递给需要类型参数的方法。

类和单例对象间的差别是，单例对象不带参数，而类可以。因为单例对象不是用new关键字实例

的，所以没机会传递给它实例化参数。每个单例对象都被实现为虚构类的实例，并指向静态的变量，此他们与Java静态类有着相同的初始化语义。单例对象在第一次被访问的时候才会被初始化。

不与伴生类共享名称的单例对象被称为独立对象。