class 包含fields,methods ,分号推测 ,singleton objects

第1节 Classes, fields, and methods

1.一个class使用new去创建该class的实例object对象

2.class中使用val和var定义属性,变量可以引用object对象。

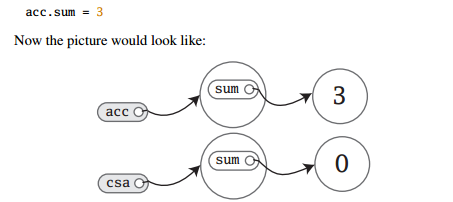
3.class中使用def定义method,去执行一个代码逻辑。

4.属性可以控制对象的状态 或者数据内容 或者object对象;

而方法是使用属性对应的数据或者状态去做真实的工作。

例如:





5.上面的例子虽然object定义的是val,说明你不能在重新赋予一个新的对象实例给该变量。

例如acc = new ChecksumAccumulator是编译不通过的。除非变成var。

但是该val对象内的属性field是可以被重新赋值的,因为sum是var。

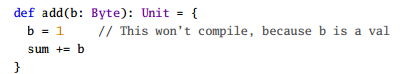
6.scala默认是public的,因此外面可以直接修改sum的值,所以属性应该设置为private的,只允许内部的method去修改该值。

class ChecksumAccumulator {

private var sum = 0

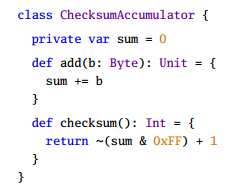
}

7.定义的方法参数类型是val的



可以看到不允许重新赋值,因此类型是val的

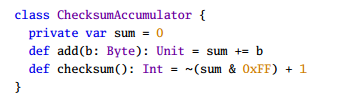
8．如何优化代码



a.return是明显多余的

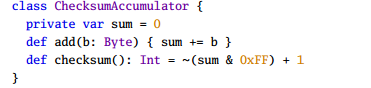
b.一行代码可以取消{}

因此优化后代码:



c. 如果一个方法返回Unit,则可以取消返回值类型和等号,并且取消{}包围body

优化代码:



9．取消=号,则返回值就是Unit的,不在乎方法的最后一行是否有具体的返回值

a.scala> def aa(x:Int) {3}

aa: (x: Int)Unit

可以看到只要取消了=号,返回值就是一个unit的。

b.def aa(x:Int):Int {3}这样写,没有等号,就会抛异常。异常原因就是返回值是Unit,但是你却让他返回Int。

c.一旦有=号,则可以进行推测执行,由方法的最后一行代码进行推测返回值

scala> def aa(x:Int) = {3}

aa: (x: Int)Int

第2节 Semicolon inference 分号推论

1.scala中分号在结尾是可以可选择的.

2.除非一行中有多个表达语句的时候,才会必须使用分号

例如 val s = "hello"; println(s)

3.如何写(x+y)

a.用回车的错误写法

x

+ y这样的语法就表示单独的两行数据,因此是有我问题的.因为会默认取消分号,因此一行就代表一个代码段

b.修改

(x

+ y) 才能保证是(x+y),因为你写到多行里面了.

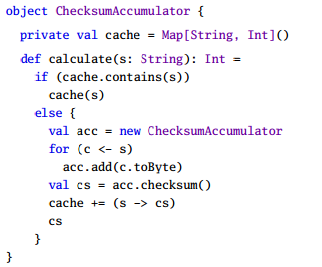
c.第二种修改方法,用+在一行的最后

x +

y

第3节 Singleton objects

1. 使用关键字object代替class,剩下的其实所有的代码更像是定义class,并且单例的名字要与class名字相同,即也成class的伴随对象



2. 必须定义class和伴随对象在同一个文件中

3. class和伴随对象能互相访问彼此的私有private成员

例如ChecksumAccumulator这个object有方法calculate,也有一个私有field

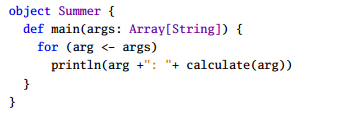
4. 伴随对象中的方法可以像java的静态方法一样,直接class.method即可调用

5.单例模式是不能有构造函数的参数的,class是可以的,原因是不能使用new去实例化一个单例模式object,因此就没有办法去传递参数了

6.单例模式是可以初始化的。

第4节 A Scala application

1.执行scala程序入口



2.执行顺序

object Test {

var aa = Seq(1,2,3,4,5,6)

aa.foreach(println(\_))

println("-----")

def main (args: Array[String]) {

test4()

}

}

虽然要执行main,但是是从上到下执行的,先会执行循环aa的代码,因为初始化的时候就执行了。只是其他方法不会被执行,会执行method而已。

3.scala默认会隐式导入java.lang and scala包,Predef是在scala包下的,包含许多有用的方法,比如println,即真正执行的是Predef.println,或者assert执行的是Predef.assert.

4.编译原是代码中两个文件

$ scalac ChecksumAccumulator.scala Summer.scala

5.运行程序

scala Summer of love

6.fsc编译的优势以及原理

上面的方式编译代码,会有一段时间的延迟.因为scala每次编译的时候要启动扫面所有的jar文件以及初始化工作

因此使用fsc进行编译

$ fsc ChecksumAccumulator.scala Summer.scala

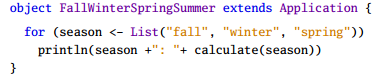
原理是:在本地上开启一个端口,将文件列表发送给端口进行编译,因此第一次启动的时候会花费时间,但是后续编译都是不需要启动了,直接调去服务,因此剩下时间。

fsc -shutdown. 停止本地编译服务

第5节 The Application trait

1.scala提供的trait, scala.Application

2.例子



3.相当于该object的{}内部的代码都存放在main方法中了

4.注意事项

a.如果你需要访问命令行参数的时候,不能使用这种方式,因为args参数数组不可用

b.因为JVM线程模型的一些限制,如果程序是多线程的话,需要使用mian函数方式

c.JVM没有优化,因此trait方式只能适用于简单的单线程程序,剩下的还是要用main函数方式