第1节 A specification for class Rational

1.什么是分数对象

分数对象,表示分子/分母,并且分母不是0,分数和浮点数是有区别的,因为浮点数是有预估的,因此分数更精准。

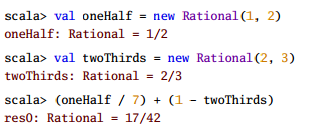
提供+ - \* /方法

+方法必须进行通分,然后分子和分母分别相加即可。

\*方法分子和分母进行相乘即可,然后进行约分.

/方法就是\*第二个数的倒数

比如



2.本节要构建一个不可变的分数对象,每次add或者乘法等等方法,产生一个新的分数对象,因此每一个原始的分数对象都是不可变的

第2节 Constructing a Rational 构建一个分数对象

1. class Rational(n: Int, d: Int) 客户端去构建一个分数对象,提供分子和分母

2.注意

scala> class Rational(n: Int, d: Int)

defined class Rational

scala> new Rational(5,7)

res12: Rational = Rational@1d4a3d2e

a.如果一个class没有任何body内容的时候,因此是可以省略{}的。

b.构造函数的时候会执行很多代码

scala的编译器可以编译在body中的任意代码,只要该代码不是field或者method,即在new的时候会执行非field和非method的代码块

class Rational(n: Int, d: Int) {

println("Created "+ n +"/"+ d)

}

打印结果

scala> new Rational(1, 2)

Created 1/2

res0: Rational = Rational@90110a

c. java与scala的不同在于,scala的构造函数中的参数,可以直接在body内使用,不需要像java一样,再次赋值才能使用

3.不可变对象编程的优点

a.不可变对象更容易变成,因为他们没有复杂的状态,不会随着时间变化

b.传递不可变对象非常自由,因为避免像可变对象一样要做一个副本,不可变对象不需要副本的概念

c.多线程变成的时候,更容易编程,因为没有线程可以改变该不可变对象的状态。

d.在hash中,不可变对象做key更安全,因为key要是变化了,可能就找不到原来存储的数据了

第3节 Reimplementing the toString method 重新实现toString方法

class Rational(n: Int, d: Int) {

override def toString = n +"/"+ d

}

第4节 Checking preconditions 校验前置条件

如何确保构造函数的数据的合法性

class Rational(n: Int, d: Int) {

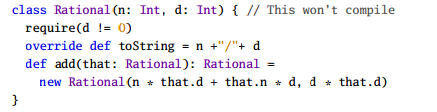
require(d != 0)

override def toString = n +"/"+ d

}

第5节 Adding fields 添加属性

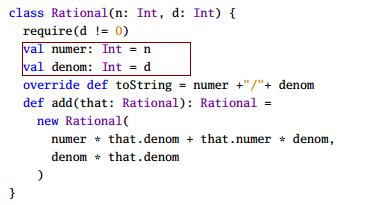
1.添加一个add方法,产生一个新的分数对象,分母是通分的结果,分子是通分后的数据加和



2.以上代码会编译出错

虽然该方法可以拿到n和d,但是却拿不到that的n和d,因此会出现编译异常。

3.修改代码



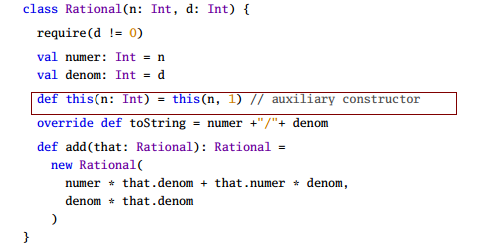
因为n和d只是该类的body可以看到,因此必须定义属性,该属性默认是public的,因此可以让其他类访问到,因此可以有that.number,以前是外部不能访问that.n的。

第6节 Self references 自引用

This表示该对象自己,因此使用this可以获取该对象本身的属性和方法,当然this也可以省略。

第7节 Auxiliary constructors 辅助构造函数

1. 有时候需要多个构造函数,scala中非主的构造函数都叫做辅助构造函数



调用val y = new Rational(3)

辅助构造函数会调用主构造函数

辅助函数以this作为方法名

2.scala和java构造函数的不同

java中构造函数第一个执行的必须是父类的super,或者执行其他的构造函数;scala中仅仅主函数能被执行,并且主函数才能执行super的构造器