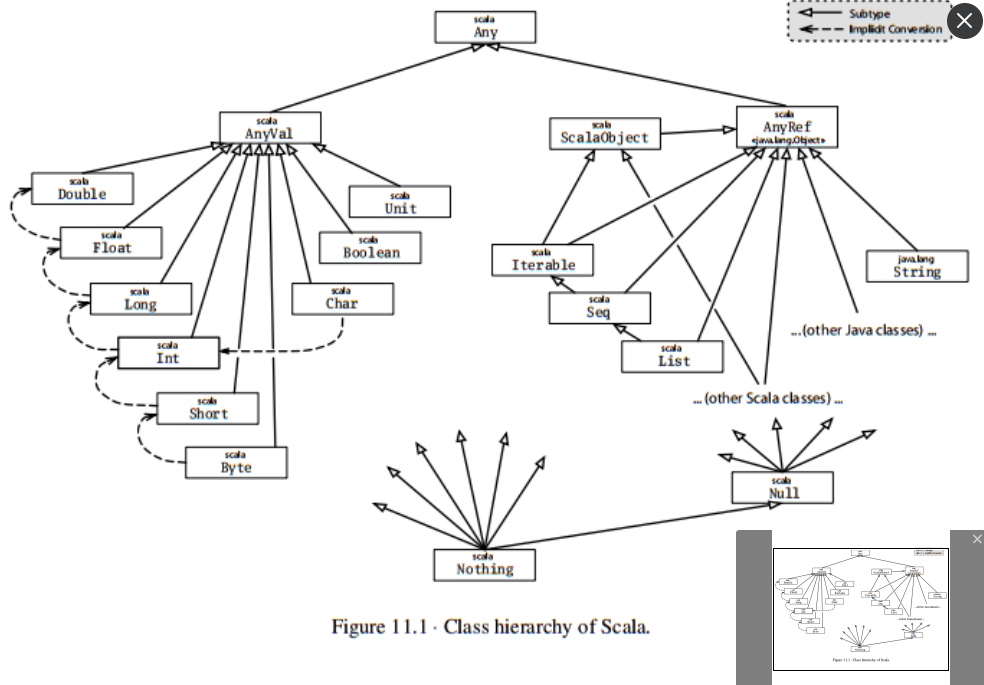
1. Scala每一个类都继承自Any,相当于java的Object。

因此Any中定义了公共方法,每一个子类都可以使用这些方法。

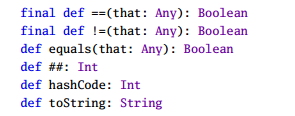
1. Scala也在继承关系的最底部定义了一些有趣的class,比如Null and Nothing

这两个类是所有类的子类,即是最底层的类了,因为是所有类的子类,因此可以返回对象的时候,可以返回null也没问题,因为null是返回对象的子类。

1. Scala’s class hierarchy
2. 继承关系图



1. Any类定义



其中a.##也是一种hash

b. == and !=是final方法,因此子类是不能复写的

c.==方法基本上与equals相同,!=基本上与equals相反.

因此相当于独立的class通过复写equals就可以实现==和!=方法了。

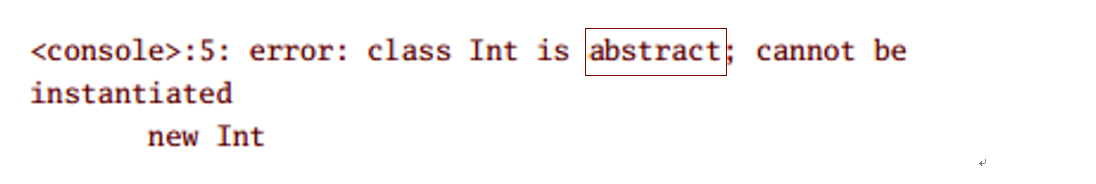
1. 跟节点Any有两个子类AnyVal and AnyRef.
2. AnyVal是每一个scala的内部基础类class的父类

a.一共9个基本类, Byte, Short, Char, Int, Long, Float, Double, Boolean, and Unit.

b.不能使用new创建这些类的实例

c.这些类都是abstract and final的类

因此new Int的时候会报错,



d.每一种类都是跟java相同的,因此可以跟java互相转换

1. AnyRef是所有引用类的基类

a.scala的类不同于java的类,因此scala的都继承自ScalaObject这个trait。

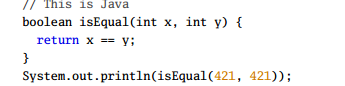
b.java的object可以使用AnyRef,基本上一个意思。

1. How primitives are implemented 原始类型如何实现的

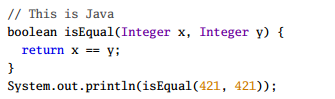
1．比如integer类型,scala与java相同的方式存储32位字节,这是为了JVM的效率,也是未来能与java类库可以互相操作的目的。

2．Scala使用java的Integer类用于int,当分配一个int变量的时候,就创建一个java的Integer类。

3．Java的自动装箱功能

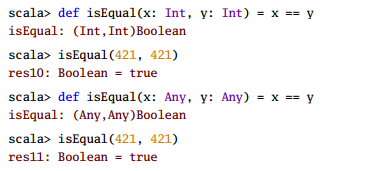


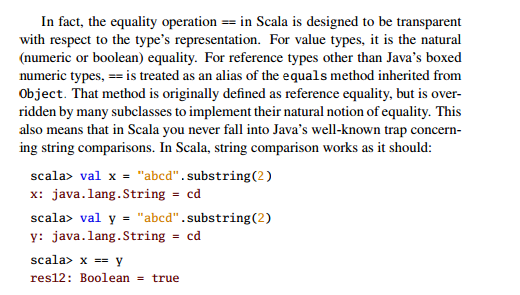
返回true,因为都是int类型的



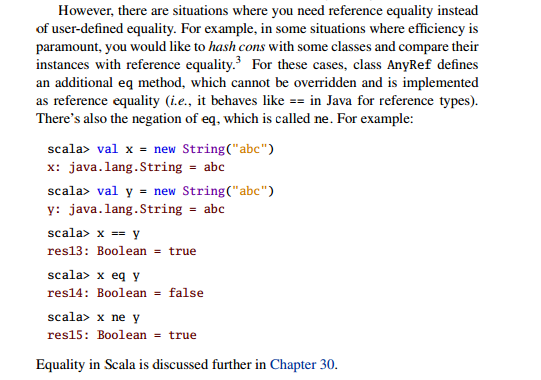
返回false,因为是Integer类型的,转换成对象了,因此不相同了

但是在scala中,都是返回true

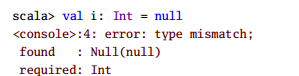




然后有时候也需要比较引用的,因此



1. Bottom types 最底部的类型
2. Class Null是null引用的一个类型
3. Null不是一个可以比较值的类型,因此不能让一个null和int比较。



1. Nothing,也是最底层的类,但是他没有值存在

第7.4节会讲述为什么他有类型,但是却没有值的原因。

一种使用场景表示异常终止



返回错误类型是Nothing,告诉用户该方法没有返回一般类型,因为Nothing是所有类的子类,因此你可以使用该方法返回各种类型

例如:



虽然要求返回Int,但是else返回的是Nothing,也是Int的一个子类,因此编译不会有问题。