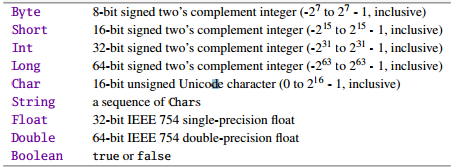
第1节 Some basic types

1.基本类型在scala包下



2. 精明的java程序员已经看到了scala的基本类型可以精准的字节范围与java类型是相应一致的,因此scala编译器能够转换scala的value type实例,转换成java的primitive types 需要的相同的字节码

第2节 Literals 字面量

1.基本类型对应的值都是字面量,可以直接使用,如果是java程序员.则这节可以跳过。

2.整数字面量

a.4种类型Int, Long, Short, and Byte

b.3种表现形式,10、16、8进制

c.0x or 0X开头,表示16进制, 例如 val hex = 0x5 val magic = 0xcafebabe

以0开头的表示8进制,例如 val oct = 035

d.如果以L或者l结尾,则表示long类型

e.可以强制转换成short和byte类型,比如 val little: Short = 367 val littler: Byte = 38

3. Floating point literals 浮点数字面量

a.可以使用e或者E表示科学计数法

比如val bigger = 1.2345e1 结果是 bigger: Double = 12.345

val biggerStill = 123E45 结果是 biggerStill: Double = 1.23E47

b.如果以F或者f结尾,表示是Float类型,否则都是double类型,默认是D或者d

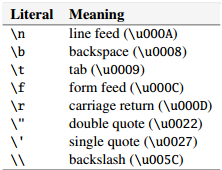
例如val littleBigger = 3e5f 结果是 littleBigger: Float = 300000.0

4. Character literals 字符字面量

a.表示任意一个Unicode编码的字节 例如val a = 'A' 结果 a: Char = A

b.使用8进制,必须使用\并且范围是0到377之间,比如 val c = '\101' 结果 c: Char = A

c.使用16进制,必须使用\u,比如val d = '\u0041' 结果是 d: Char = A



d.命名变量名字也可以用16进制,比如val B\u0041\u0044 = 1 结果是 BAD: Int = 1

5. String literals 字符串字面量

a.双引号包裹着

b.转义比较麻烦 比如 val escapes = "\\\"\'" 结果escapes: java.lang.String = \"'

即\\需要转义成\ \"转义成” \'转义成’

c.三个引号避免转义

比如

println("""Welcome to Ultamix 3000.

Type "HELP" for help.""")

结果是Welcome to Ultamix 3000.

Type "HELP" for help.

但是上面的结果没有办法上下两行数据对齐,我们的输出是看不出来的,如何对其呢,以下方式

println("""|Welcome to Ultamix 3000.

|Type "HELP" for help.""".stripMargin)

6. Symbol literals 符号字面 形式是'+任意字符,注意’表示的是单引号

a. symbol字面量表示’+任意字母/数字的形式,他都会映射到scala.Symbol上,比如'cymbal 转换成Symbol("cymbal")

b. 这种方式用于动态使用哪个属性中

比如:

def updateRecordByName(r: Symbol, value: Any) {} 返回值是updateRecordByName: (Symbol,Any)Unit

该方法表示两个参数,一个是数据库记录对应的属性name,另外一个参数是对应的值.

这样就可以动态的更新数据库的任意属性对应的值了,实现动态编程

但是scala会报错,比如updateRecordByName(favoriteAlbum, "OK Computer") ,报错说

<console>:6: error: not found: value favoriteAlbum

updateRecordByName(favoriteAlbum, "OK Computer")

即favoriteAlbum是不存在的,因为你没定义该变量对应的值是什么。

因此你可以如下写updateRecordByName('favoriteAlbum, "OK Computer")

c. symbol没有什么特别的方法,只有一个name方法

scala> val s = 'aSymbol

s: Symbol = 'aSymbol

scala> s.name

res20: String = aSymbol

d. 值得注意的是,symbols是被实习的,如果你写相同的symbol两次,他们两个引用的是相同的对象

第3节 Operators are methods操作方法

1.scala针对基本类型提供了丰富的操作集合

2.1 + 2 真正的调用是(1).+(2),即Int有一个+方法,该方法有一个Int的参数,返回值也是Int

3.+这样的操作符 表示中缀操作符(infix),

任意方法都可以这么执行,比如str indexOf 'o' 表示字符串有一个indexOf方法,参数是字符,返回值是int

注意如果方法是需要两个以上参数的时候,则参数要用括号,比如 s indexOf ('o', 5)

4.s.indexOf('o') 这样写,indexOf 就不是一个操作符,但是s indexOf 'o' 这样indexOf就是一个操作符

因此总结就是中缀操作符是作用在object和参数之间的形式

5.前缀(prefix)和后缀(postfix)操作符

6.prefix 前缀

在方法名在object前,比如‘-’ in -7 方法是-,作用在7这个int对象上,即int对象7调用了-方法

7.postfix

方法放在object之后,比如toLong 写成 7 toLong 方法是toLong,作用在7这个int对象上,即int对象7调用 toLong方法

8.postfix方式用在没有参数的方法上,因为在scala中没有参数的方法,可以省略(),比如s.toLowerCase,转换成s toLowerCase形式

9.前缀操作符

unary\_方法,-2.0 会转换成(2.0).unary\_- 这样的方法进行转换

+, -, !, and ~.这些可以使用前缀操作符

如果你定义了一个方法unary\_!,则可以执行!p

但是不能定义方法unary\_\*,此时\*p是不可用的.你必须执行p.unary\_\*才可以使用,主要是\*在很多语言中,比如c++中都是关键字

第5节 Arithmetic operations

+ - \* / %

第6节 Relational and logical operations > < || &&等

注意 || 和 && 是短路操作的

第7节 Bitwise operations 按位操作

1.bitwise-and (&), bitwise-or (|), and bitwise-xor (ˆ). (~, the method unary\_~)

1 & 2 结果是1 (0001) and 2 (0010),相同1的才是1,因此结果是0000,即0

1 | 2 结果是1 (0001) and 2 (0010),有1的就是1,因此结果是0011,即3

1 ˆ 3 结果是1 (0001) and 3 (0011),即一个0和一个1对应的结果才是1,相同的都是0,因此结果是0010,即2

~1 结果是1 (0001) 被反转,即11111111111111111111111111111110,即-2

2. shift left (<<),shift right (>>), and unsigned shift right (>>>).

第7节 Object equality 即两个对象的比较==或者!=

1.不仅仅用于基础对象,可以用于任意对象

比如 List(1, 2, 3) == List(1, 2, 3)

res34: Boolean = true

2. 原理是如果非null,就调用equals方法,因为equals方法依赖于类型,因此不同类型自然返回的是false

3.String的时候内容相同就返回true,跟java不同

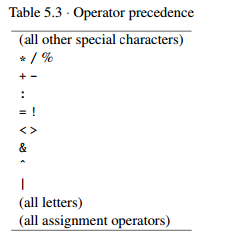
("he"+"llo") == "hello" res40: Boolean = true

第8节 Operator precedence and associativity 操作的优先级以及结合性

1.优先级决定了表达式的哪一部分先执行,但是scala没有提供解析的功能,仅仅通过操作符的优先级进行的执行

如果方法名字开始是\*,因此他有一个更高的优先级比+,即2 + 2 \* 7 = 2 + (2 \* 7),

a +++ b \*\*\* c 也是一样,方法以\*开头的比+优先级大,+++和\*\*\*是方法,因此执行顺序是a +++ (b \*\*\* c)



2.举例

比如2 << 2 + 2 结果是32

因为<<方法是<开头的,而<比+优先级小,因此先计算+,即2 << (2 + 2) 即 2 << 4 = 32

比如2 + 2 << 2 结果是16

即(2+2) << 2,即4 << 2 = 16

3.优先级规则的一个例外,如果一个操作符是以=号结束的,并且该操作符不是比较操作符,<=, >=, ==, or !=,那么该操作符的优先级等同于=号的优先级。

比如x \*= y + 1 中\*=的优先级等于=,即x \*= (y + 1)

4. 当多个操作符优先级一样的时候,则按顺序执行即可,可以进行结合律

:是right的操作,因此任意一个方法,以:结尾,则都执行right操作

比如a \* b = a.\*(b),但是a ::: b = b.:::(a)

如果优先级一样的时候,:::的执行顺序是从right到left,其他的则是left到right

例如:a ::: b ::: c 是 a ::: (b ::: c) 先 b是c的参数,然后a是结果的参数

而a \* b \* c顺序是(a \* b) \* c.

5.尽量用()替换优先级,因为优先级很容易理解错了,导致程序是错误的

第9节 Rich wrappers 包装类

