Scala核心编程

-变量

尚硅谷研究院

- 为什么需要变量

一个程序就是一个世界，在scala中一切都是对象

- 变量是程序的基本组成单位

不论是使用哪种高级程序语言编写程序,变量都是其程序的基本组成单位，比如:

```
package com.atguigu.chapter02

object ScalaFunDemo01 {

  def main(args: Array[String]): Unit = {
      var a : Int = 1 //定义一个整型变量,取名a,并赋初值1
      var b : Int = 3 //定义一个整型变量,取名b,并赋初值3
      b = 89 //给变量b 赋 89
      println("a=" + a) //输出语句,把变量a的值输出
      println("b=" + b) //把变量b的值输出
  }
}
```

- 变量的介绍

概念

变量相当于内存中一个数据存储空间的表示，你可以把变量看做是一个房间的门牌号，通过门牌号我们可以找到房间，而通过变量名可以访问到变量(值)。

变量使用的基本步骤

1) 声明/定义变量 (scala要求变量声明时初始化)
2) 使用

- 变量基本使用

Scala变量使用案例入门

看演示并对代码进行说明

```
//1.声明变量【告诉计算机要开一个房间,并赋值】
var num : Int = 0
var score : Double = 1.0
var gender : Char  = 'N'
var name : String = "scott"
```

变量内存&

- **Scala变量使用说明**

变量声明基本语法

var | val 变量名 [: 变量类型] = 变量值

注意事项

1) 声明变量时，类型可以省略（编译器自动推导,即类型推导）
2) 类型确定后，就不能修改，说明Scala 是强数据类型语言.
3) 在声明/定义一个变量时，可以使用var 或者 val 来修饰， var 修饰的变量可改变，val 修饰的变量不可改 [案例].

4) val修饰的变量在编译后，等同于加上final，通过反编译看下底层代码。

```
object ScalaFunDemo01 {

  var num1 = 10
  val num2 = 20
```

反编译看val和var的

- **Scala变量使用说明**

注意事项

5) var 修饰的对象引用可以改变，val 修饰的则不可改变，但对象的状态(值)却是可以改变的。(比如: 自定义对象、数组、集合等等) [分析val好处]

```
class Dog {
  var age  = 100
}
```

6) 变量声明时，需要初始值。

- 程序中 +号的使用

1) 当左右两边都是数值型时，则做加法运算
2) 当左右两边有一方为字符串，则做拼接运算

```
var name : String = "kristina"
var score = 89.9
```

- 数据类型

scala数据类型介绍

- Scala 与 Java有着相同的数据类型，**在Scala中数据类型都是对象**，也就是说scala没有java中的原生类型

- Scala数据类型分为两大类 AnyVal(值类型) 和 AnyRef(引用类型)， 注意：**不管是AnyVal还是AnyRef 都是对象**。[案例演示 Int , Char]

```
var num1 : Int = 10
println("num1" + num1)
var char1 : Char = 'a'
println("char1的code= " + char1.toInt)
```

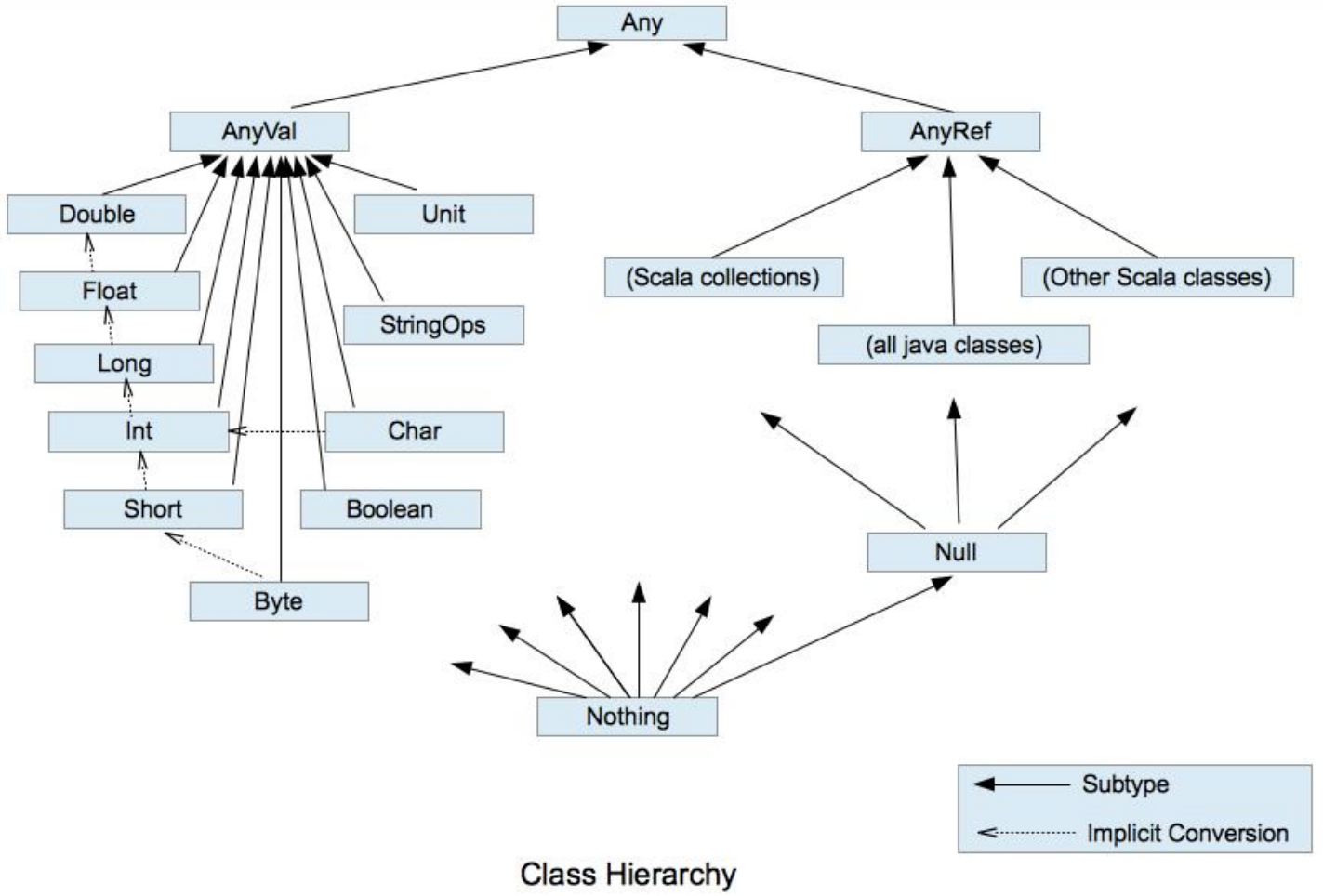
- 相对于java的类型系统，scala要复杂些！也正是这复杂多变的类型系统才让面向对象编程和函数式编程完美的融合在了一起

- 数据类型

scala数据类型体系一览图

- 小结

1) xx
2) yy



- 数据类型

scala数据类型列表

数据类型	描述
Byte	8位有符号补码整数。数值区间为 -128 到 127
Short	16位有符号补码整数。数值区间为 -32768 到 32767
Int	32位有符号补码整数。数值区间为 -2147483648 到 2147483647
Long	64位有符号补码整数。数值区间为 -9223372036854775808 到 9223372036854775807
Float	32 位, IEEE 754标准的单精度浮点数
Double	64 位 IEEE 754标准的双精度浮点数
Char	16位无符号Unicode字符, 区间值为 U+0000 到 U+FFFF
String	字符序列
Boolean	true或false
Unit	表示无值，和其他语言中void等同。用作不返回任何结果的方法的结果类型。Unit只有一个实例值，写成()。
Null	null
Nothing	Nothing类型在Scala的类层级的最低端；它是任何其他类型的子类型。
Any	Any是所有其他类的超类
AnyRef	AnyRef类是Scala里所有引用类(reference class)的基类

- **整数类型**

基本介绍

Scala的整数类型就是用于存放整数值的，比如 12 , 30, 3456等等

整型的类型

数据类型	描述
Byte [1]	8位有符号补码整数。数值区间为 -128 到 127
Short [2]	16位有符号补码整数。数值区间为 -32768 到 32767
Int [4]	32位有符号补码整数。数值区间为 -2147483648 到 2147483647
Long [8]	64位有符号补码整数。数值区间为 -9223372036854775808 到 9223372036854775807 = 2的(64-1)次方-1

案例演示：

```
("min =" + Byte.MinValue
("max =" + Long.MinValue
```

- 整数类型

整型的使用细节

1) Scala各整数类型有固定的表数范围和字段长度，不受具体OS的影响，以保证Scala程序的可移植性。

2) Scala的整型 **常量/字面量** 默认为 Int 型，声明Long型 **常量/字面量** 须后加‘l’’或‘L’ [反编译看]
3) Scala程序中变量常声明为Int型，除非不足以表示大数，才使用Long

```
var c = 11 // c 就是Int类型
println("c=" + c)
var d = 12l // d 就是 Long 类型 或者 var d = 12L
println("d=" + d)

var e = 9223372036854775807 // 正确吗? 如何解决
```

- 浮点类型

基本介绍

Scala的浮点类型可以表示一个小数，比如 123.4f，7.8 ，0.12等等

浮点型的分类

Float [4]	32 位, IEEE 754标准的单精度浮点数
Double [8]	64 位 IEEE 754标准的双精度浮点数

- 浮点类型

浮点型使用细节

1) 与整数类型类似，Scala 浮点类型也有固定的表数范围和字段长度，不受具体OS的影响。
2) Scala的浮点型常量默认为Double型，声明Float型常量，须后加‘f’或‘F’。

```
var f1 : Float = 1.1   // double->float 错误
var f2 = 1.2 //  ok 类型推断
var f3 : Double = 1.3 //  ok
var f4 : Float = 1.4f // ok
var f5 : Double = 1.5f  // float->double , ok
```

3) 浮点型常量有两种表示形式
   十进制数形式：如：5.12　　512.0f　　.512　(必须有小数点）
   科学计数法形式:如：5.12e2 = 5.12乘以10的2次方　5.12E-2 = 5.12除以10的2次方
4) 通常情况下，应该使用Double型，因为它比Float型更精确(小数点后大致7位)
   //测试数据：2.2345678912f , 2.2345678912

- 字符类型(Char)

基本介绍

字符类型可以表示**单个字符**,字符类型是Char， 16位无符号Unicode字符(2个字节),区间值为 U+0000 到 U+FFFF

案例演示：

```
var c1 : Char = 'a'
var c2 : Char = '\t'
var c3 : Char = '你'
var c4 : Char = 97
```

- 字符类型(Char)

字符类型使用细节

1) 字符常量是用单引号(‘ ’)括起来的单个字符。例如：var c1 = 'a‘ var c2 = '中‘ var c3 = '9'

2) Scala 也允许使用转义字符 ‘\’来将其后的字符转变为特殊字符型常量。例如：var c3 = ‘\n’ // '\n'表示换行符

3)可以直接给Char赋一个整数，然后输出时，会按照对应的unicode 字符输出 ['\u0061' 97]

4) Char类型是可以进行运算的，相当于一个整数，因为它都对应有Unicode码.

转义字符	说明
\b	退格符
\n	换行符
\r	回车符
\t	制表符
\"	双引号
\'	单引号
\\	反斜线

```
var c1 : Char = 'a'
var num : Int = 10 + c1 + 'b'
```

问题：

var c2 : Char = ‘a’ + 1 正确吗?

- 字符类型(Char)

字符类型本质探讨

1) 字符型 存储到 计算机中，需要将字符对应的码值（整数）找出来
存储：字符——>码值——>二进制——>存储
读取：二进制——>码值——> 字符——>读取

2) 字符和码值的对应关系是通过字符编码表决定的(是规定好)， 这一点和Java一样。

```
var c3 : Char = '国'
println("c3=" + c3 + "c3对应的码值=" + c3.toInt)
```

- **布尔类型：Boolean**

基本介绍

1) 布尔类型也叫Boolean类型，Booolean类型数据**只允许取值**true和false
2) boolean类型占1个字节。
3) boolean 类型适于**逻辑运算**，一般用于程序流程控制[后面详解]：

- if条件控制语句；
- while循环控制语句；
- do-while循环控制语句；
- for循环控制语句

案例演示：

- **Unit类型、Null类型和Nothing类型**

基本说明

Unit	表示无值，和其他语言中void等同。用作不返回任何结果的方法的结果类型。Unit只有一个实例值，写成()。
Null	null , Null 类型只有一个实例值 null
Nothing	Nothing类型在Scala的类层级的最低端；它是任何其他类型的子类型。当一个函数，我们确定没有正常的返回值，可以用Nothing 来指定返回类型，这样有一个好处，就是我们可以把返回的值（异常）赋给其它的函数或者变量（兼容性）

- **Unit类型、Null类型和Nothing类型**

使用细节和注意事项

1) Null类只有一个实例对象，null，类似于Java中的null引用。null可以赋值给任意**引用类型(AnyRef)**，但是不能赋值给**值类型(AnyVal: 比如 Int, Float, Char, Boolean, Long, Double, Byte, Short)**

2) Unit类型用来标识过程，也就是没有明确返回值的函数。由此可见，Unit类似于Java里的void。Unit只有一个实例，()，这个实例也没有实质的意义

```
def sayOk(): Unit = { //
  println("sayok被调用")
}
```

3) Nothing，可以作为没有正常返回值的方法的返回类型，非常直观的告诉你这个方法不会正常返回，而且由于Nothing是其他任意类型的子类，他还能跟要求返回值的方法兼容。

```
def test() : Nothing = {
    throw new Exception()
}
```

没有正常返回值

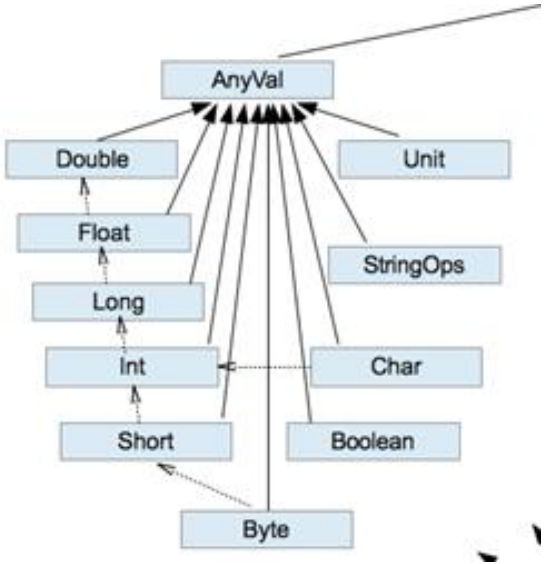
- 值类型转换

值类型隐式转换

➢ 介绍

当Scala程序在进行赋值或者运算时，精度小的类型自动转换为精度大的数据类型，这个就是自动类型转换(**隐式转换**)。

➢ 数据类型按精度(容量)大小排序为



- 值类型转换

值类型隐式转换

➢ 案例演示

演示一下基本数据类型转换的基本情况。

```
var a : Int = 'c'
var d : Double = 80
```

图片1.z

➢ 自动类型转换细节说明

1) 有多种类型的数据混合运算时，系统首先自动将所有数据转换成容量最大的那种数据类型，然后再进行计算。 5.6 + 10 = 》double
2) 当我们把精度(容量)大 的数据类型赋值给精度(容量)小 的数据类型时，就会报错，反之就会进行自动类型转换。
3) (byte, short) 和 char之间不会相互自动转换。

```
var b : Byte = 10
var c : Char = b
```

图片2.z

4) byte，short，char 他们三者可以计算，在计算时首先转换为int类型。
5) 自动提升原则： 表达式结果的类型自动提升为 操作数中最大的类型

```
var b : Byte  = 10
var c : Char = 90
var s : Short = b + c
var s2 : Short = 10 + 90
var s3 : Short = 100
```

图片3.z

- 值类型转换

高级隐式转换和隐式函数

scala还提供了非常强大的**隐式转换机制**(隐式函数，隐式类等等)，我们放在高级部分专门用一个章节来讲解

- 值类型转换

强制类型转换

➢ 介绍

自动类型转换的逆过程，将容量大的数据类型转换为容量小的数据类型。使用时要加上强制转函数，**但可能造成精度降低或溢出**,格外要注意。

➢ 案例演示

java ： int num = (int)2.5
scala： var num : Int = 2.7.toInt //对象

- 值类型转换

强制类型转换

➢ 强制类型转换细节说明

1) 当进行数据的 从 大——>小，就需要使用到强制转换
2) **强转符号只针对于最近的操作数有效**，往往会使用小括号提升优先级

```
val num1: Int = 10 * 3.5.toInt + 6 * 1.5.toInt  // 36
val num2: Int = (10 * 3.5 + 6 * 1.5).toInt // 44
println(num1 + " " + num2)
```


图片4.z

3) Char类型可以保存 Int的常量值，但不能保存Int的变量值，需要强转
4) Byte和Short类型在进行运算时，当做Int类型处理。

- 值类型转换-课堂练习题

判断是否能够通过编译,并说明原因

1）var s : Short = 5 // ok
    s = s-2   // error Int -> Short

2） var b : Byte = 3   // ok
     b = b + 4   // error Int ->Byte
     b = (b+4).toByte   // ok ， 使用强制转换

3）var c : Char = 'a' //ok
    var i : Int = 5 //ok
    var d : Float = .314F //ok
    var result : Double = c+i+d   //ok Float->Double

4） var b : Byte = 5 // ok
    var s : Short = 3 //ok
     var t : Short = s + b // error Int->Short
     var t2 = s + b   // ok, 使用类型推导

- 值类型和String类型的转换

介绍

在程序开发中，我们经常需要将基本数据类型转成String 类型。
或者将String类型转成基本数据类型。

基本类型转String类型

语法： 将基本类型的值+"" 即可
案例演示：

```
String str1 = true + "";
String str2 = 4.5 + "";
String str3 = 100 + "";
```

String类型转基本数据类型

语法：通过基本类型的**String的 toXxx**方法即可
案例演示：
MFC

"12"
s1.toInt
s1.toFloat
s1.toDouble
s1.toByte
s1.toLong
s1.toShort

- 值类型和String类型的转换

注意事项

1) 在将String 类型转成 基本数据类型时，**要确保String类型能够转成有效的数据**，比如 我们可以把 "123" , 转成一个整数，但是不能把 "hello" 转成一个整数

2) 思考就是要把 "12.5" 转成 Int //?

- 标识符的命名规范

标识符概念

1) Scala 对各种变量、**方法**、**函数**等命名时使用的**字符序列**称为标识符
2) 凡是自己可以起名字的地方都叫**标识符**

标识符的命名规则(记住)

Scala中的标识符声明，基本和Java是一致的，但是细节上会有所变化。

1) 首字符为字母，后续字符任意字母和数字，美元符号，可后接下划线_
2) 数字不可以开头。
3) **首字符为操作符(比如+ - * /)，后续字符也需跟操作符 ,至少一个(反编译)**
4) 操作符(比如+-*/)不能在标识符中间和最后.
5) 用**反引号`....`**包括的任意字符串，即使是关键字(39个)也可以 [true]

- 标识符的命名规范

标识符举例说明

```
hello    // ok
hello12 // ok
1hello  // error
h-b   // error
x h   // error
h_4   // ok
_ab   // ok
Int    // ok, 在scala中，Int 不是关键字，而是预定义标识符,可以用，但是不推荐
Float  // ok
_    // 不可以，因为在scala中，_ 有很多其他的作用，因此不能使用
Abc    // ok
+*-   // ok
+a  // error
```

- 标识符的命名规范

标识符命名注意事项

1) **包名：**尽量采取有意义的包名，简短，有意义

2) 变量名、函数名 、方法名 采用驼峰法。

- **Scala关键字**

关键字介绍

Scala有39个关键字：

- package, import, class, object, trait, extends, with, type, forSome
- private, protected, abstract, sealed, final, implicit, lazy, override
- try, catch, finally, throw
- if, else, match, case, do, while, for, return, yield
- def, val, var
- this, super
- new
- true, false, null

谢谢！ 欢迎收看！