# Day02 JAVA语言基础

## 计算机常量

### 常量的定义：

本身不可再改变的量叫做常量

### 常见的几个常量类型：

#### 整数常量：

所有的整数 3，4，100。

#### 小数常量：

所有的小数 1.02,5.36，8.7209。

1.2.3字符常量：将一个符号用 ‘ ’ 标志起来就构成一个字符常量 ‘a’ ‘+’ ‘1’ ‘ ’ 。

1.2.4字符串常量：将多个字符用 “ ”标志起来就构成了一个字符串常量 “abc” “12w”

1.2.5布尔常量：只有两个值---true/false---用于表示逻辑值

1.2.6空常量：null

## 进制及其转换

### 二进制:

用0-1来表示一个数，满2进1 ，在JDK1.7以前Java代码中不允许直 接表示二进制数字，从jdk1.7开始允许在代码中使用二进制数字，要求以0b/0B开头， 例如：1+1=10 11+1=100 0b100100 0B10111

### 八进制：

用0-7来表示一个数，满8进1 ，以0开头标志八进制数字 07+01=10 016+01=017 017+01=20 015 07 012 027

### 十进制：

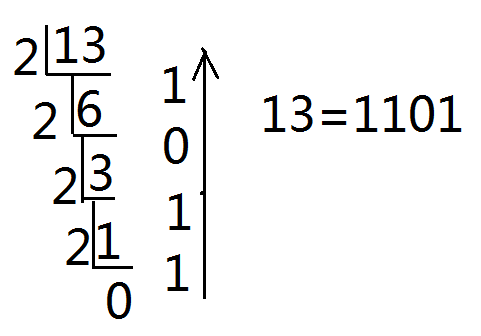
用0-9来表示一个数，满10进1

### 十六进制：

0-9，A-F/a-f，满16进1 以0x/0X开头标志十六进制数字 9+1=a a+1=b f+1=10 0x23 0xa 19+1=1a 1f+1=20

### 十进制转换成二进制：

不断地除以2取余数，将余数倒排



### 二进制转化成十进制：

从低位次开始，每一位乘以2的位次之幂，然后将积求和



### 十进制转化为其他进制：

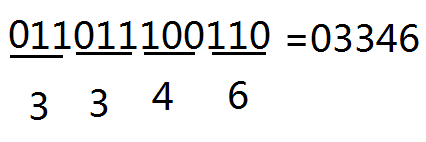
就是除以对应的进制然后取余倒排

### 其他进制转化为十进制：

乘以对应进制的位次次幂然后求和（参照二级制来做）

### 二进制转换成八进制：

从低位次开始，每三位化为一组，产生一个八进制数字，最 高位如果不足三位，则补0---三变一

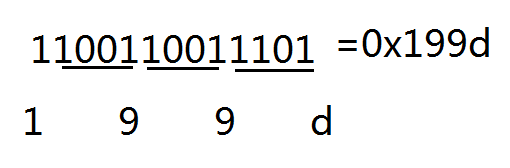


### 八进制转换为二进制：

一变三（将八进制每一位都用三位二进制数来表示）

### 二进制转化为十六进制：

四变一



### 十六进制转化为二进制：

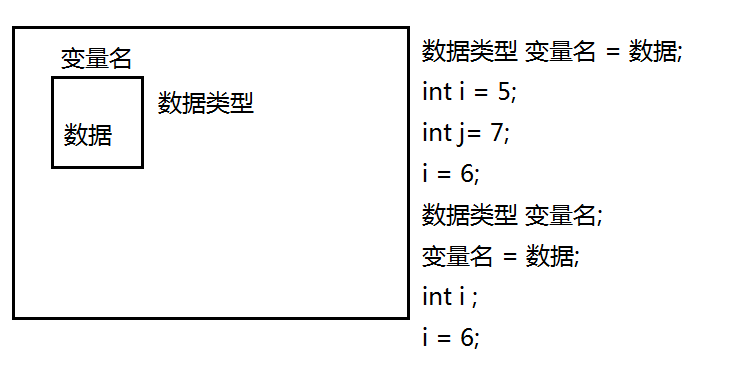
一变四（将十六进制每一位都用四位二进制数来表示）

## 变量

### 变量的定义：

我们在类中定义的用来存储数据的可以改变的量。

定义格式：



注意：变量名是不可以重复的。

## 数据类型及其转换

### 基本数据类型

#### 整数型：

* byte---字节型---占1个字节 --- 表示范围是-27~27-1 - -128~127

byte b = 25; byte b2 = 128;---false（超出范围）

* short---短整型---占2个字节 --- 表示范围是-215~215-1 --- -32768~32767

short s = 3; short s = -109;---几乎不用

* int---整型---占4个字节--- 表示范围是-231~231-1 --- 2.1\*1010~2.1\*1010---整数默认为 int类型System.out.println(6); int i = 129;从JDK1.7开始int j = 3\_186\_485\_318; ->编译 完成之后就是int j = 3186485318;
* long---长整型---占8个字节---表示范围 -263~263-1 --- -9\*1018~9\*1018---后边添加L/l标志是一个长整型：long l = 31864853188L;

#### 浮点型：

* float---单精度---占4个字节 ---表示范围 -1038~1038---要求必须以F/f标志float类型 的数字，float f = 3.23f;
* double---双精度---占8个字节--- 表示范围-10308~10308 ---小数默认为double

System.out.println(3.4); double d = 4.2; double d2 = 5.6D; double d3 = 6.01d;

#### 字符型：

* char---字符型---占2个字节--- 0~216\_1 --- 0~65535

char a= ’a’; char b=’中’;(对于中文而言，采用gbk编码，一个汉字占两个字节，用 utf-8，占用三个字节。)

转义字符：‘\t’ ---table 制表符 ‘\r’---return 回车 ‘\n’---next/newLine换行

‘\\’ ---\ ‘\’’ --- 单引号 ‘\”’ --- 双引号

#### 布尔类型:

* boolean ---布尔型---只能表示true/false---内存大小根据jdk版本和操作系统来确定

boolean b = true;

### 引用数据类型：

数组、类、接口。

#### 数据类型的转换:

* 自动类型转换/隐式转换：

规律一：小的类型可以自动转化为大的类型

byte--->short--->int--->long

float--->double

规律二：整数能够自动转化为小数，但是会产生精度损失

规律三：字符可以转化成整数

* 强制类型转换：

double d=3; int a =(int)d;

Int a=5；byte b= (int)a;

注意，小数类型转化为整数的时候，小数部分会舍掉。

## 数据的原反补三码

任意一个数据都有原码、反码、补码这么三种形式。

数据在计算机中是以补码形式来存储的，因此实际上计算的也是数据的补码。

对于正数，原反补三码一致。

int i = 5;

原码：00000000 00000000 00000000 00000101

反码：00000000 00000000 00000000 00000101

补码：00000000 00000000 00000000 00000101

对于负数，反码是在原码的基础上最高位不变，其余位1变0，0变1;

补码在反码的基础上+1

注意：最高位是符号位，0表示正数，1表示负数

int i = -3;

原码：10000000 00000000 00000000 00000011

反码：11111111 11111111 11111111 11111100

补码：11111111 11111111 11111111 11111101

## 算术运算符

常用运算符：+ - \* / % ++ --

* 关于+ - \* /：

byte/short/char在运算的时候会自动的提升为int类型

整数运算完成之后的结果一定是整数

当小类型和大类型一起运算的时候结果一定是大的类型

byte b = 4 + 1; --- 4 和1都是常量，值确定，所以在编译的时候进行优化，编 译完成之后结果就是byte b = 5;

整数/0 -- ArithmeticException---算术异常 任意一个非零小数/0;任意一个非 零数字/0.0---Infinity 0.0/0 ，0/0.0 ，0.0/0.0---NaN---Not a Number---非数字

* 关于%

% 取模---取余

7%3=1 2%5=2

-18%-5=-3 --- 对于负数的取余，先忽略符号按照正数取余来运算，运算完成之后看% 左边数字的符号，如果左边为正，结果为正，左边为负，结果为负。

注意：小数无法精确运算，绝大部分小数在内存中无法精确存储。

* 关于++/-- ：自增/自减

如果++/--在前，先自增/自减然后参与后续运算

如果++/--在后，先将值取出来参与后续运算然后再自增/自减

int i = 5;

int j = ++i; -> 先将i自增为6，然后将6赋值给j，所以j的值也是6；

int i = 5;

int j = i++;-> 先将i的值5取出来，然后i自增为6，最后将5赋值给j

int i = 3;

int j = ++i + 1; -> 先将i自增为4，然后将i的值4取出来参与+1，最后将4+1赋值 给j，所以j的值为5

int i =3;

int j = i++ + 1; -> 先将i的值3取出来参与+1运算，i再自增为4，然后将3+1的值 赋值给j，所以j的值是4

注意：byte/short/char类型也可以参与++/--,结果依然是原来的类型，是在底层做了 强制类型的转换。

## 赋值运算符

常见的赋值运算符：= += -= \*= /= %= &= |= ^= <<= >>= >>>=

运算规则：

int i = 8;

i += 2; -> i = i + 2; -> i = 10

int i = 6;

i -= 4; -> i = 2;

注意：

除了=以外，其余的符号要求变量必须先有值才能使用

byte/short/char可以参与赋值运算

int i = 5;

i += i -= i\*= i++; -> i=-15

i = 5 + ( 5 - (5 \* 5));

int i = 5;

i = i++; ---先将i的值5取出来，然后i自增为6，最后将5赋值给i -> i = 5。

## 关系运算符

常见关系运算符：==等于 !=不等 > < >= <=

关系运算符的使用：

3 == 4 --false

3 != 4 --true

instanceof --- 关键字 --- 判断对象和类的关系的---不能判断基本类型

System.out.println(“abc” instanceof String); --- true

## 逻辑运算符

常见的逻辑运算符：

&与---And ，|或---Or ，!非---Not， ^异或---Xor， &&--短路与， ||--短路或

逻辑运算符的使用：

true&true=true， true&false=false ，false&true=false ，false&false=false（同为true，结 果才为true）

true|true=true， true|false=true ，false|true=true ，false|false=false（有一个为true， 结果就为true）

!true=false ，!false=true（！非，取反）

true^true=false， true^false=true ，false^true=true ，false^false==false（两个不同为true）

&&：如果&&左边的结果为false，那么&&右边的表达式就不再运算---短路

||：如果||左边的结果为true，那么||右边的表达式就不再运算

注意：||可以把&&短路掉，但是&&不能把||短路掉。

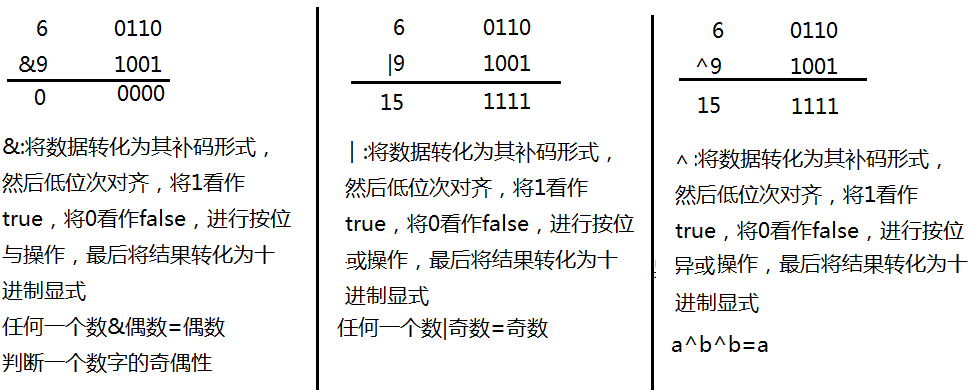
## 位运算符

注意：位运算符针对的是整数的补码。

常用的位运算符：&与， |或， ^异或 ，<<左移， >>右移 ，>>>无符号右移，

~取反

运算规则：



## 三元运算符

三元运算符的格式：逻辑值? 表达式1:表达式2

执行顺序：如果逻辑值为true，执行表达式1；反之，执行表达式2

注意事项：三元运算一定有结果，结果可以用变量来接住

double d = a > b ? 3 : 5.2;

double d2 = i > j ? 1.2 : ‘b’;

m > n ? true : 3; --- 不可以

两个表达式结果的类型要么一致，要么其中一个能够自动转化为另一个

i, j, k

int max = i > j ? (i > k ? i : k) : (j > k ? j : k);---三元表达式的嵌套