1. Java基础语法

山西优逸客科技有限公司官网

*http://www.sxuek.com/*

|  |  |  |
| --- | --- | --- |
| 编辑时间 | 编辑版本 | 编辑人 |
| 2019-7-10 | V2.0 | 优逸客大数据实训组 |
|  |  |  |

本章内容：

* 关键字与保留字
* 标识符
* 变 量

基本数据类型

基本数据类型转换

进制与进制间的转换

* 运算符
* 程序流程控制

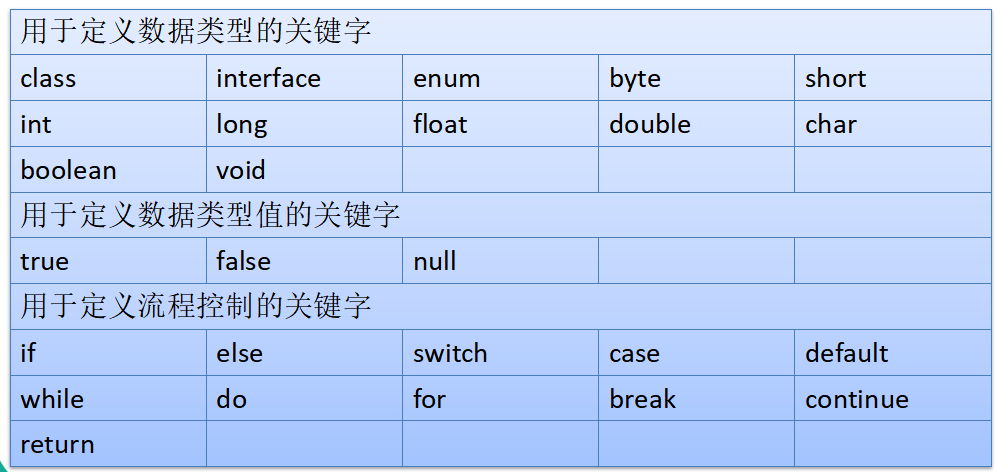
# 第一章 数据类型：

## 1.1 关键字与保留字：

关键字的定义和特点：

定义：被Java语言赋予了特殊含义，用做专门用途的字符串（单词）

特点：关键字中所有字母都为小写





保留字(reserved word)

Java保留字：现有Java版本尚未使用，但以后版本可能会作为关键字使用。自己命名标记符时要避免使用这些保留字 byValue、cast、future、 generic、 inner、 operator、 outer、 rest、 var 、 goto 、const；

## 1.2 标识符：

Java 对各种变量、方法和类等要素命名时使用的字符序列称为标识符；

凡是自己可以起名字的地方都叫标识符。

定义合法标识符规则：

* 由26个英文字母大小写，0-9 ，\_或 $ 组成
* 数字不可以开头。
* 不可以使用关键字和保留字，但能包含关键字和保留字。
* Java中严格区分大小写，长度无限制。
* 标识符不能包含空格。

注意：在起名字时，为了提高阅读性，要尽量有意义，“见名知意”。

Java中的名称命名规范：

* 包名：多单词组成时所有字母都小写：xxxyyyzzz
* 类名、接口名：多单词组成时，所有单词的首字母大写：XxxYyyZzz
* 变量名、方法名：多单词组成时，第一个单词首字母小写，第二个单词开始每个单词首字母大写：xxxYyyZzz
* 常量名：所有字母都大写。多单词时每个单词用下划线连接：XXX\_YYY\_ZZZ

## 1.3 变量：

变量的概念：

* 内存中的一个存储区域；
* 该区域的数据可以在同一类型范围内不断变化；
* 该区域有自己的名称（变量名）和类型（数据类型）；

变量的作用：

用于在内存中保存数据；

使用变量注意：

Java中每个变量必须先声明，后使用

变量的作用域：一对{ }之间有效

初始化值

使用变量名来访问这块区域的数据

声明变量：

语法：<数据类型> <变量名称>

例如：int var;

变量的赋值：

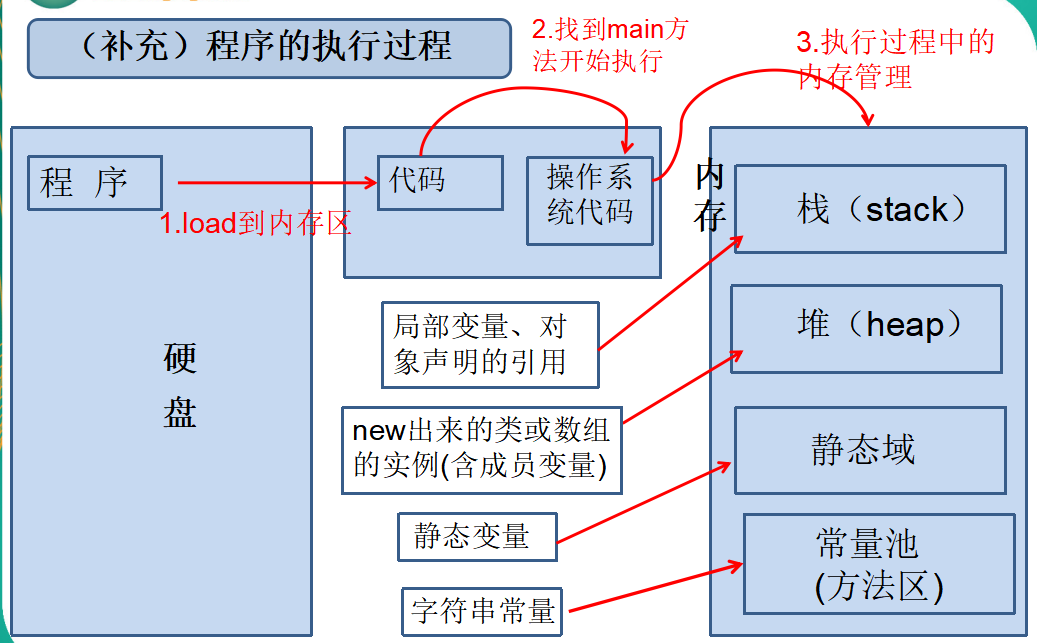
语法：<变量名称> = <值>

例如：var = 10;

声明和赋值变量：

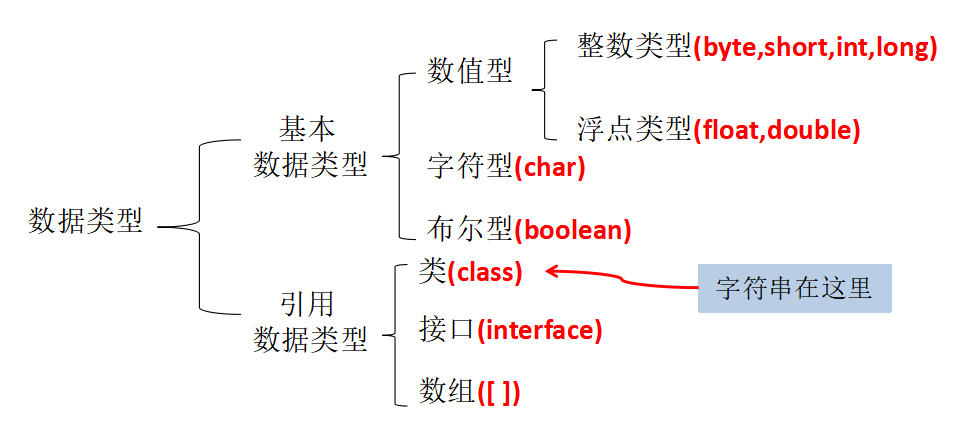
语法： <数据类型> <变量名> = <初始化值>

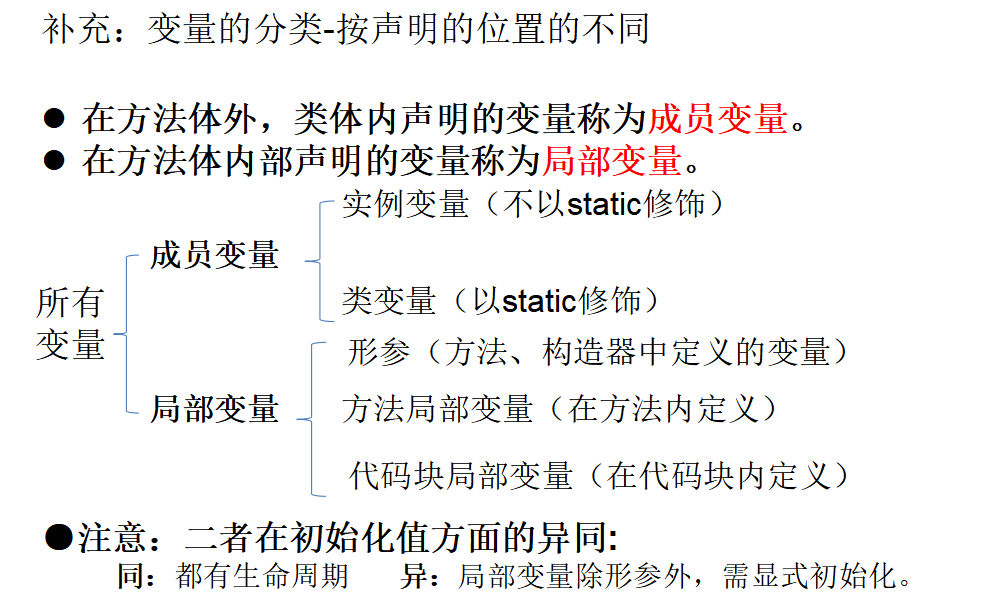
例如：int var = 10;



## 1.4 变量的分类-按数据类型：

对于每一种数据都定义了明确的具体数据类型，在内存中分配了不同大小的内存空间。



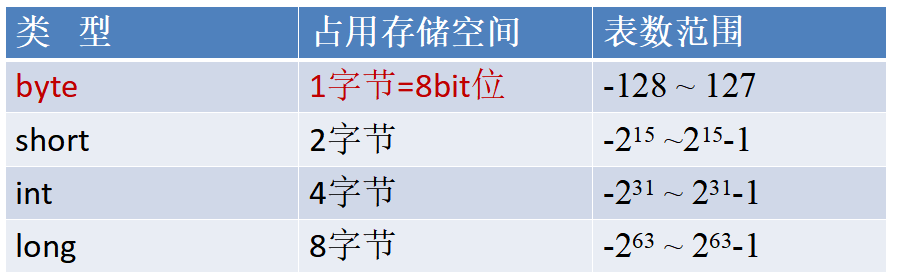


### 1.4.1 整数类型：byte、short、int、long：

Java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java程序的可移植性。

java的整型常量默认为 int 型，声明long型常量须后加‘l’或‘L’

java程序中变量常声明为int型，除非不足以表示大数，才使用long



500MB 1MB = 1024KB 1KB= 1024B B= byte ? bit?

bit: 计算机中的最小存储单位。byte:计算机中基本存储单元。

### 1.4.2 浮点类型：float、double：

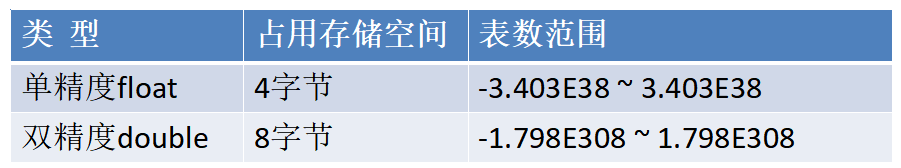
与整数类型类似，Java 浮点类型也有固定的表数范围和字段长度，不受具体OS的影响。

Java 的浮点型常量默认为double型，声明float型常量，须后加‘f’或‘F’。

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 512E2 100E-2



### 1.4.3 字符类型：char：

char 型数据用来表示通常意义上“字符”(2字节)；

注意：为了将字符存储到计算机中，需要引入一个字符编码集，将字符先通过字符编码集解析成对应的二进制，然后进行存储；

字符型常量的三种表现形式：

1）字符常量是用单引号(‘ ’)括起来的单个字符，涵盖世界上所有书面语的字符。例如：char c1 = 'a'; char c2 = '中'; char c3 = '9';

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



3）直接使用 Unicode 值来表示字符型常量：‘\uXXXX’。其中，XXXX代表一个十六进制整数。如：\u000a 表示 \n。

char类型是可以进行运算的。因为它都对应有Unicode码。

### 1.4.4 编码集介绍：

ASCII 码：

在计算机内部，所有数据都使用二进制表示。每一个二进制位（bit）有 0 和 1 两种状态，因此 8 个二进制位就可以组合出 256 种状态，这被称为一个字节（byte）。 一个字节一共可以用来表示 256 种不同的状态，每一个状态对应一个符号，就是 256 个符号，从 0000000 到 11111111。

ASCII码：上个世纪60年代，美国制定了一套字符编码，对英语字符与二进制位之间的关系，做了统一规定。这被称为ASCII码。ASCII码一共规定了128个字符的编码，比如空格“SPACE”是32（二进制00100000），大写的字母A是65（二进制01000001）。这128个符号（包括32个不能打印出来的控制符号），只占用了一个字节的后面7位，最前面的1位统一规定为0。

缺点：

不能表示所有字符。

相同的编码表示的字符不一样：比如，130在法语编码中代表了é，在希伯来语编码中却代表了字母Gimel (ג)

Unicode 编码：

Unicode码：可以包含世界上所有的字符，但固定的长度，有些浪费存储空间，Unicode 编码表中的字符占两个字节；

乱码：世界上存在着多种编码方式，同一个二进制数字可以被解释成不同的符号。因此，要想打开一个文本文件，就必须知道它的编码方式，否则用错误的编码方式解读，就会出现乱码。

Unicode：一种编码，将世界上所有的符号都纳入其中。每一个符号都给予一个独一无二的编码，使用 Unicode 没有乱码的问题。

Unicode 的缺点：Unicode 只规定了符号的二进制代码，却没有规定这个二进制代码应该如何存储：无法区别 Unicode 和 ASCII：计算机无法区分三个字节表示一个符号还是分别表示三个符号。另外，我们知道，英文字母只用一个字节表示就够了，如果unicode统一规定，每个符号用三个或四个字节表示，那么每个英文字母前都必然有二到三个字节是0，这对于存储空间来说是极大的浪费。

UTF-8：

UTF-8 ： 对Unicode的优化，可以包含世界上所有字符，字母占1个字节，汉字占3个字节；

UTF-8 是在互联网上使用最广的一种 Unicode 的实现方式。

UTF-8 是一种变长的编码方式。它可以使用 1-6 个字节表示一个符号，根据不同的符号而变化字节长度。

UTF-8的编码规则：

1）对于单字节的UTF-8编码，该字节的最高位为0，其余7位用来对字符进行编码（等同于ASCII码）。

2）对于多字节的UTF-8编码，如果编码包含 n 个字节，那么第一个字节的前 n 位为1，第一个字节的第 n+1 位为0，该字节的剩余各位用来对字符进行编码。在第一个字节之后的所有的字节，都是最高两位为"10"，其余6位用来对字符进行编码。

### 1.4.5 布尔类型：boolean：

boolean 类型适于逻辑运算，一般用于程序流程控制：

* 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；

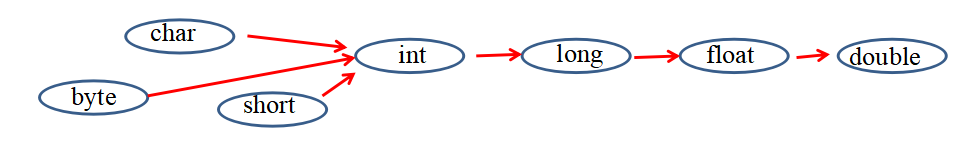
boolean类型数据只允许取值true和false，无null。

不可以0或非 0 的整数替代false和true，这点和C语言不同。

### 1.4.6 基本数据类型转换：

1）自动类型转换：

容量小的类型自动转换为容量大的数据类型。数据类型按容量大小排序为：



（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。

（2）byte,short,char之间不会相互转换，他们三者在计算时首先转换为int类型。

（3）当把任何基本类型的值和字符串(String)进行连接运算时(+)，基本类型的值将自动转化为字符串类型。

2）强制类型转换：

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换符（()），但可能造成精度降低或溢出,格外要注意。

通常，字符串不能直接转换为基本类型，但通过基本类型对应的包装类则可以实现把字符串转换成基本类型。

如： String a = “43”;

int i = Integer.parseInt(a);

boolean类型不可以转换为其它的数据类型。

3）案例练习：

### 1.4.7 字符串类型：String：

String不是基本数据类型，属于引用数据类型

使用方式与基本数据类型一致。例如：String str = “abcd”;

一个字符串可以串接另一个字符串，也可以直接串接其他类型的数据。例如：

str = str + “xyz” ; 或 str += “xyz” ; int n = 100; str += n;

# 第二章 运算符：

运算符是一种特殊的符号，用以表示数据的运算、赋值和比较等。包含有：

1）算术运算符

2）赋值运算符

3）比较运算符（关系运算符）

4）逻辑运算符

5）位运算符

6）三元运算符

### 2.1 算术运算符：

属于二元运算符，符号两边有两个操作数；



算术运算符的注意问题：

如果对负数取模，可以把模数负号忽略不记，如：5%-2=1。 但被模数是负数则不可忽略。此外，取模运算的结果不一定总是整数。

对于除号“/”，它的整数除和小数除是有区别的：整数之间做除法时，只保留整数部分而舍弃小数部分。 例如：int x=3510;x=x/1000\*1000; x的结果是？

“+”除字符串相加功能外，还能把非字符串转换成字符串.例如：System.out.println("5+5="+5+5); //打印结果是？

### 2.2 赋值运算符：

1）符号：=

当“=”两侧数据类型不一致时，可以使用自动类型转换或使用强制类型转换原则进行处理。支持连续赋值。

2）扩展赋值运算符(复合运算符)： +=, -=, \*=, /=, %=

*注意：运算的结果不会改变原变量的数据类型（在复合运算符中，隐藏了强制转换，会自动转换成前面的操作数的类型）；*

### 2.3 比较运算符(关系运算符)：



1）关系运算符用来进行比较运算，操作数与关系运算符组成关系运算表达式；

2）关系运算符的结果时布尔值；

3）> < >= <= ：操作数数值类型，包含整型，浮点型，字符型；

4）== != ：基本类型和引用类型都可以；

比较运算符的结果都是boolean型，也就是要么是true，要么是false。

比较运算符“==”不能误写成“=” 。

### 2.4 逻辑运算符：

1）逻辑运算符与布尔操作数一起使用，组成逻辑表达式，所以，操作数要求是boolean值，运算结果也是Boolean值；逻辑运算符用于连接布尔型表达式,例：a>90&&b<100&&c>80;

注意：在Java中不可以写成3<x<6，应该写成x>3 & x<6 。

& ：逻辑与，作为逻辑运算符时，两个操作数都是真时，为真，任意一个为假，则为假

| ：逻辑或，有一个为真，就为真；

^ ：逻辑异或，相同为假，不同为真；

！ ：逻辑反/逻辑非，非真即假；

&& ：短路与；

|| ：短路或；

2） & | 与 && || 的区别：

（1）运算规则相同；

（2） && || 只判断第一个操作数；

& 无论任务情况，它的两边的表达式都会参与计算；

&& 当它的左边为false时，则将不会计算其右边的表达式；即左false则false;

| 与 || 类似；

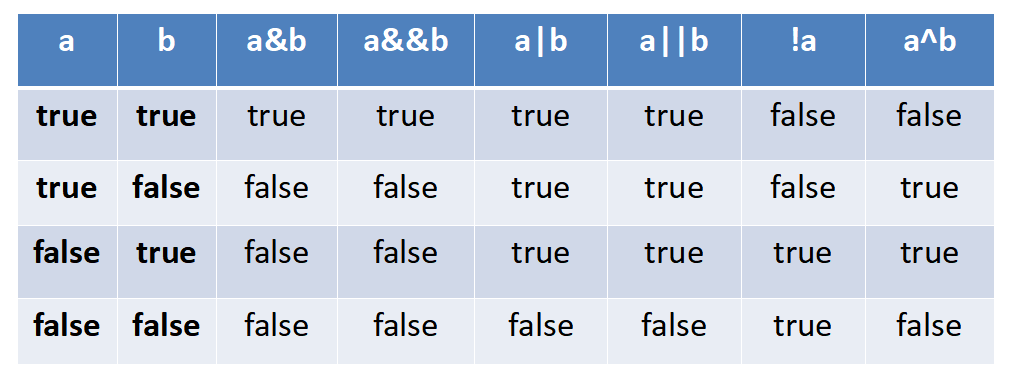
单&时，左边无论真假，右边都进行运算；

双&时，如果左边为真，右边参与运算，如果左边为假，那么右边不参与运算。

“|”和“||”的区别同理，||表示：当左边为真，右边不参与运算。

异或( ^ )与或( | )的不同之处是：当左右都为true时，结果为false。

理解：异或，追求的是“异”!

逻辑运算符的优先级：! > & > ^ > |

### 2.5 位运算符：

位运算符先将十进制数转成对应的二进制的补码形式进行运算，并且最高位参与运算；



& ： 按位与，采用二进制进行运算，同1，为1，其他为0；



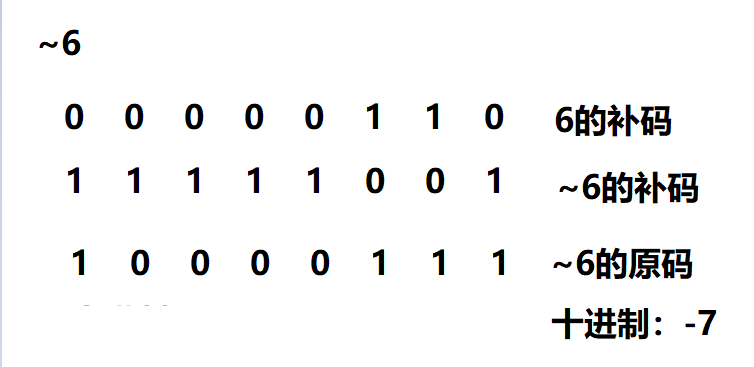
| ： 按位或，采用二进制进行运算，有一个1，就是1；



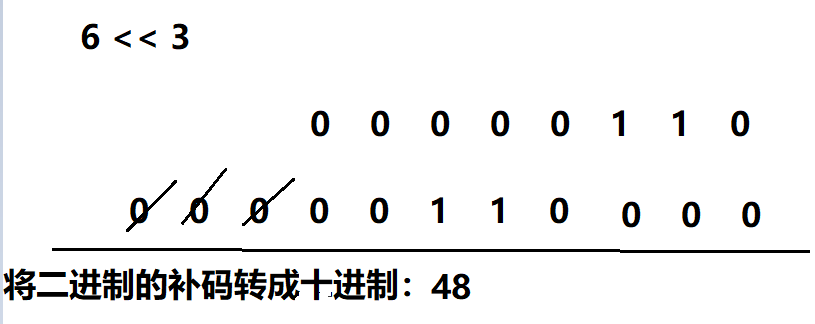
^ ： 按位异或，采用二进制进行运算，相同为0，不同为1；



~ ： 按位取反，数值位取反加1；



<< : 桉位左移，采用二进制进行运算，变大，直接左移，用0来补位，结尾移几位就乘以2的几次方； 相当于 ： a\*2^b (a乘以2的b次方)



>> : 桉位右移，采用二进制进行运算，变小，直接右移，补位取决于最高位（0补0，1补1），右移几位，就除以2的几次方；如果是正数，左边的高位用0补；如果是负数，左边的高位用1补；相当于： a/2^b (a除以2的b次方)



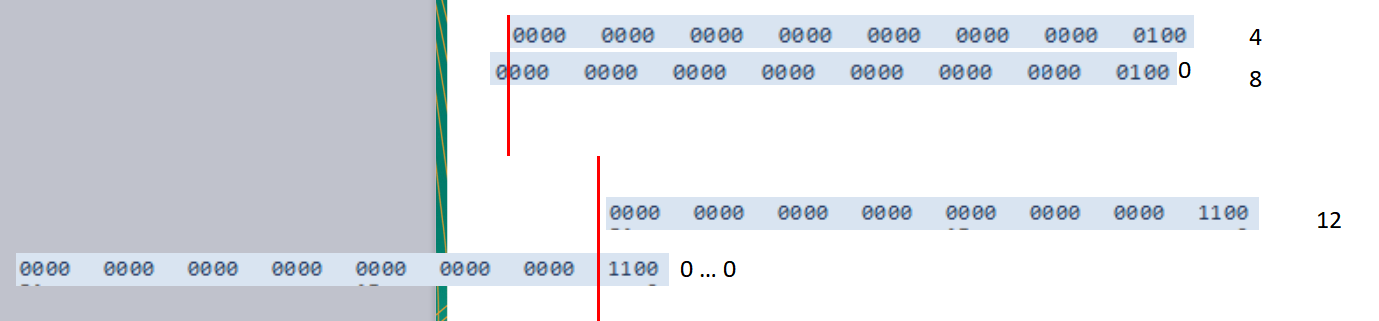
>>> : 无符号桉位右移，采用二进制进行运算，直接右移，最高位只补0。

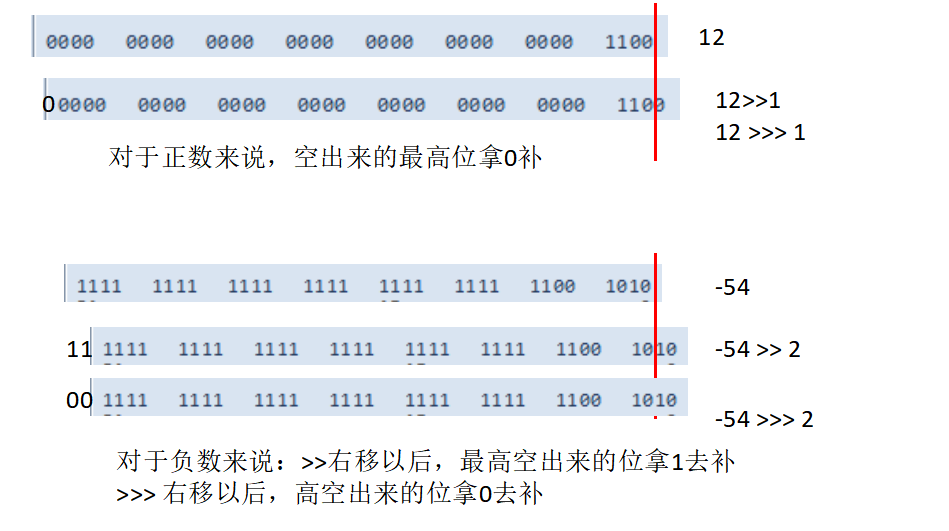


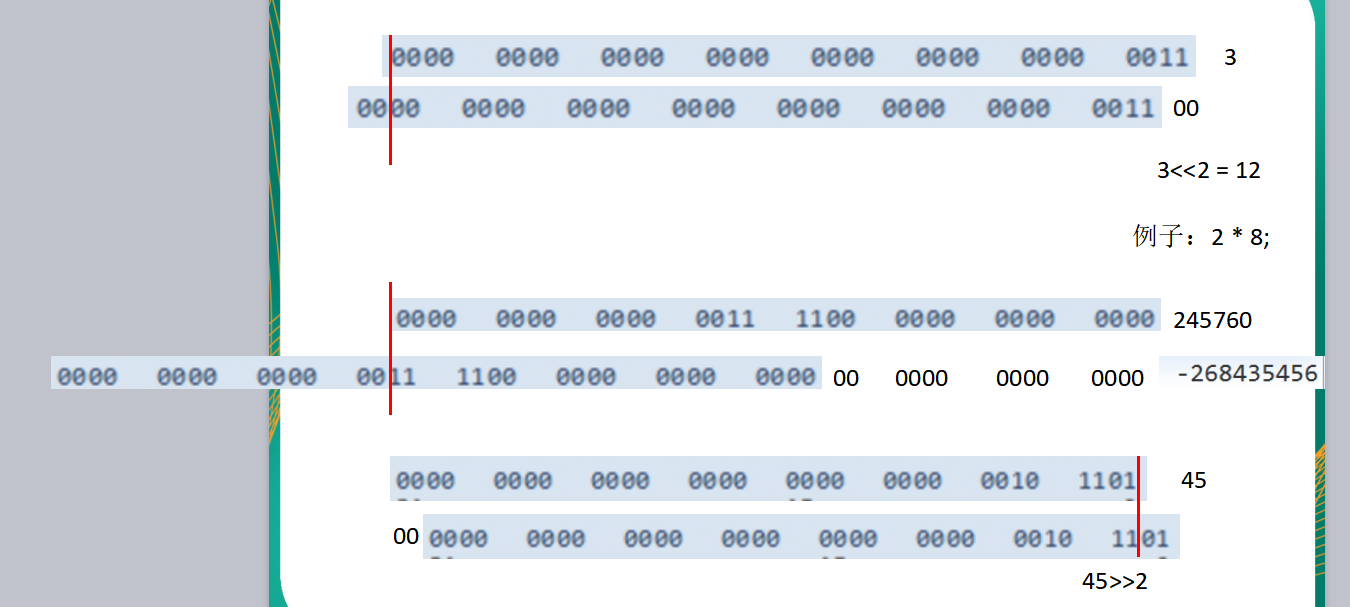
注意：

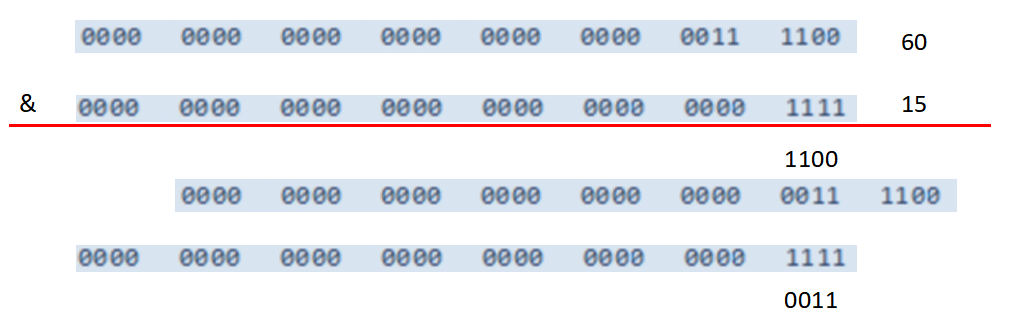
如果操作数是boolean,程序会认为是逻辑运算符，如果是数值型，则认为是位运算符；

如果操作数是char,byte,short时，在位移动前，先把该操作数自动转换成int, 运算结果也是int;









|  |
| --- |
| *21 << 1 : 42*    *21 << 26 : 1409286144*  *//在一定的范围内，每向左移一位，原有的数值 \* 2：*  *21 << 27 : -1476395008*  *//在一定的范围内，每向右移一位，原有的数值 / 2：*  *21 >> 1 : 10*      *//案例：最高效的计算：2 \* 8 ： 2 << 3* |

### 2.6 三元运算符：

1）语法格式:

操作数为3个，语法格式：x ? y : z

2）执行过程：

其中x为boolean类型表达式，先计算x的值，若为true,则整个三目运算的结果为表达式y的值，否则整个运算结果为表达式z的值。

3）作用：经常用来代替简单的if else判断：

4）三元运算符与if-else的联系与区别：

1）三元运算符可简化if-else语句

2）三元运算符要求必须返回一个结果。

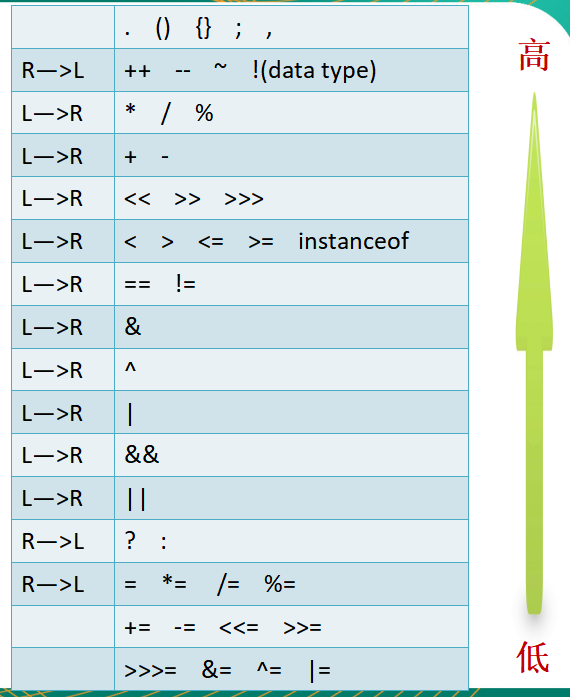
3）if后的代码块可有多个语句

### 2.7 运算符的优先级：

小括号>自增、自减>算术运算 > 位运算> 关系运算 > 逻辑运算 > 条件运算 > 赋值运算

运算符有不同的优先级，所谓优先级就是表达式运算中的运算顺序。如右表，上一行运算符总优先于下一行。

只有单目运算符、三元运算符、赋值运算符是从右向左运算的。



# 第三章 程序流程控制：

顺序结构：程序从上到下逐行地执行，中间没有任何判断和跳转。

分支结构：根据条件，选择性地执行某段代码。有if…else和switch-case两种分支语句。

循环结构：根据循环条件，重复性的执行某段代码。有while、do…while、for三种循环语句。

注：JDK1.5提供了foreach循环，方便的遍历集合、数组元素。

## 3.1 顺序结构：

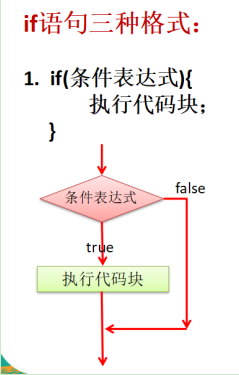
顺序结构：Java中定义成员变量时采用合法的前向引用。如：

|  |
| --- |
| *public class Test{*  *int num1 = 12;*  *int num2 = num1 + 2;*  *}*  错误形式：  *public class Test{*  *int num2 = num1 + 2；*  *int num1 = 12;*  *}* |

## 3.2 分支语句1：if-else结构：

程序从两条或者多条路径中选择一条执行，这种结构就是分支结构；

### 3.2.1 单分支if语句：



If语句中的表达式的形式：要求必须是boolean类型结果；

1）boolean变量： if(true){} if(false){}

2）boolean变量的赋值： if(a=true){} if(a=false){}

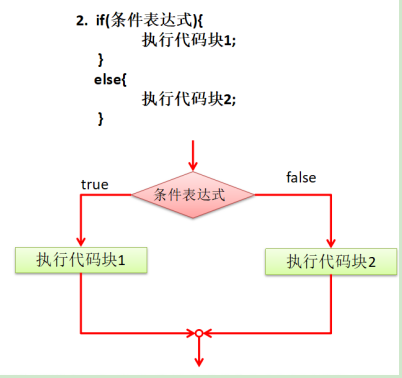
3）关系表达式：if(a > b){}

4）逻辑表达式：if(a > b && a >c){}

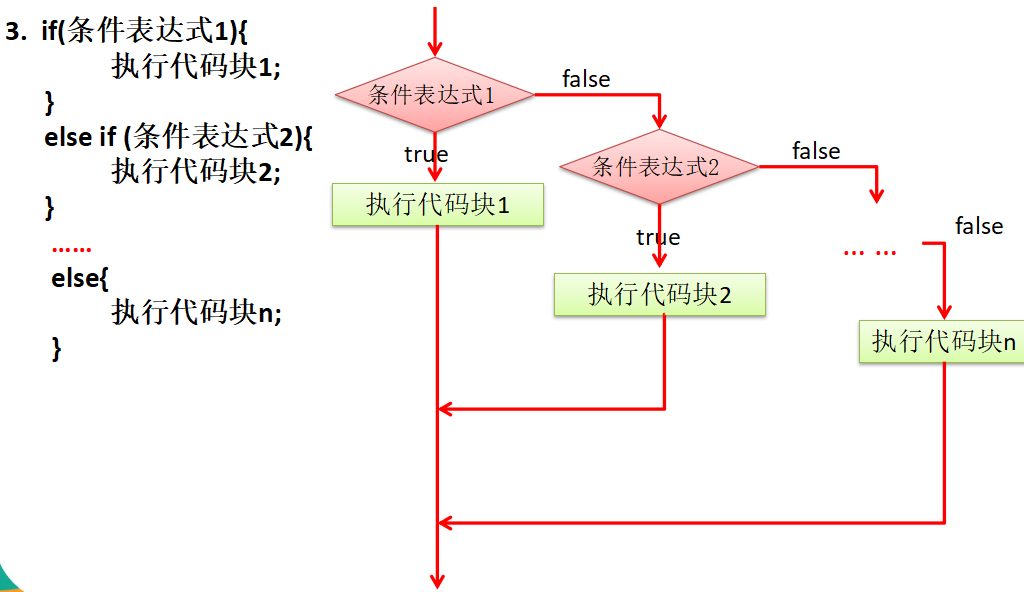
注意：大括号中的代码块可以是0行或者任意行代码；

当大括号中只有一行执行语句时，可以把大括号去掉：

### 3.2.2 双分支if else语句：



### 3.2.3 多分支if else if else语句：



1）从上往下依次判断if条件，如果某个条件成立，则下面的条件将不再执行；

2）else if块的顺序最好不要调换，因为可能影响效果；

3）else块可以省略，也可能影响效果；

4）区分开多重if结构和多个if结构；

### 3.2.4 嵌套分支语句：

一个分支结构中又完整的嵌套了另外一个分支结构：

## 3.3 分支语句2：switch语句：

### 3.3.1 语法结构：

|  |
| --- |
| *switch(表达式){*  *case 常量1:*  *语句1;*  *break;*  *case 常量2:*  *语句2;*  *break;*  *… …*  *case 常量N:*  *语句N;*  *break;*  *default:*  *语句;*  *break;*  *}* |

### 3.3.2 执行顺序：

先判断出switch括号中的变量或表达式的值，然后依次判断case的常量值是否与之相等；

如果相等，则执行对应case的语句块，直到遇见break为止

如果没有相等的，则执行default语句块，直到遇见break为止

### 3.3.3 语法注事项：

1、变量或表达式的类型：byte、short、int、枚举、String、char

2、case后只能是常量

3、case后的常量不能重复

4、break从语法上可以省略，但影响效果

5、default从语法上可以省略而且可以调换顺序，但影响效果

### 3.3.4 if和switch对比：

相同点：

都属于分支结构，很多时候都可以互相替换解决分支结构的题目

不同点：

能用switch结构的题目，肯定能用if结构解决

能用if结构解决的题目，不一定能用switch结构解决，必须满足以下条件：

①判断的类型有限制，必须是byte、short、int、枚举、String、char

②必须为等值判断；

## 3.4 循环结构：

*案例：打印100遍“我最帅”。*

*没有用循环*

*System.out.println(“我最帅”);*

*System.out.println(“我最帅”);*

*System.out.println(“我最帅”);*

*……*

*缺点：*

*1、代码量大*

*2、不易修改和扩展*

*使用循环：*

*Int i=1;*

*while(i<=100){*

*System.out.println(“我最帅”);*

*i++;*

*}*

*循环的好处：*

*1、提高代码的重用性，代码更加简洁*

*2、提高代码的维护性，使代码扩展性更强！*

循环语句 ： 在某些条件满足的情况下，反复执行特定代码的功能；

循环四要素：

1）循环变量初始化（循环变量赋初值）；

2）循环条件：满足该条件，将进入循环体执行操作（循环终止条件）；

3）循环操作：反复执行的代码（循环语句）；

4）循环变量更新（让循环趋向终止的条件，步进、迭代）；

循环语句分类：

1）for 循环

2）while 循环

3）do/while 循环

循环的使用场景：

1）知道循环的次数或知道一个范围

示例：打印100遍，统计10个学生，求100——200之间的和

1.需要先套用一个知道条件的框架

int i=初始值;

while(i<=次数或终止值){

i++;

}

2.再分析循环操作即可

3.检测是否四要素齐全或语法有没有改进的地方

2）不知道循环的次数：

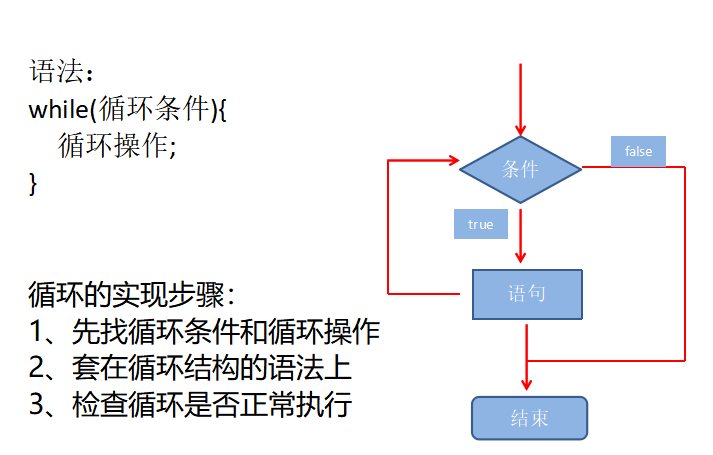
1、将while框架套上

while(){}

2、分析循环条件和循环操作

3、检测是否有四要素

### 3.4.1 while 循环语句：



语法格式：

|  |
| --- |
| *[初始化语句]*  *while( 布尔值测试表达式)｛*  *语句或语句块(如果表达式为真，则代码块会一致执行下去);*  *[更改语句;]*  *}* |

特点：

|  |
| --- |
| *1）在循环刚开始时，会计算一次“布尔表达式”的值，若条件为真，执行循环体。而对于后来每一次额外的循环，都会在开始前重新计算一次。*  *2）语句中应该有使用循环趋向于结束的语句，否则会出现无限循环----> "死"循环；*  *3）while : 未知循环次数； （迭代 = 步进）* |

### 3.4.2 do-while 循环语句：

语法格式：

|  |
| --- |
| *[初始化语句]*  *do｛*  *语句或语句块;*  *[更改语句;]*  *｝while(布尔值测试表达式);* |

特点：

没有初始化条件。必须先执行do一次，才有初始化条件(特点：至少执行一次)；

do while : 先执行，再判断；while后面加分号；

while : 先判断，再执行；

|  |
| --- |
| *int ii = 0;*  *while (ii < 0) {*  *System.out.println(ii);*  *ii++;*  *}*  *ii = 0;*  *do {*  *System.out.println(ii);*  *ii++;*  *}while(ii < 0);* |

while和do while对比：

相同点：

都能解决任何类型的循环题目

都具备循环的四要素

不同点：

执行顺序不同,do while先执行后判断；while 先判断后执行；

执行效果不同，如果第一次条件不成立，do while 至少执行一次；while执行0次

如果第一次条件成立，do while和while执行次数一样

语法不同;

### 3.4.3 for 循环语句：

for循环语句是支持迭代的一种通用结构，是最有效，最灵活的循环结构；

语法格式：执行顺序

|  |
| --- |
| *for(1.初始化表达式 ;2.布尔表达式；4.步进){*  *3.循环体；*  *}* |

注意事项：

1、for循环在执行条件测试后，先执行程序部分，再执行步进；

2、在for语句的初始化部分声明的变量，其作用域为整个for循环体；

3、“初始化” 和“循环条件表达式”，部分可以使用逗号来执行多个操作；

4、如果三个部分都为空语句（分号不能省），相当于一个无限循环；

### 3.4.4 嵌套循环(多重循环)：

将一个循环放在另一个循环体内，就形成了嵌套循环。其中，for ,while ,do…while均可以作为外层循环和内层循环。

实质上，嵌套循环就是把内层循环当成外层循环的循环体。当只有内层循环的循环条件为false时，才会完全跳出内层循环，才可结束外层的当次循环，开始下一次的循环。

设外层循环次数为m次，内层为n次，则内层循环体实际上需要执行m\*n=mn次。

### 3.4.5 跳转结构：

break 语句：

在任何循环语句的主体部分，均可用break控制循环的流程，break用于强行退出循环，不执行循环中剩余的语句。（break 只能与switch , 循环中使用，不能单独使用）

特点：

1. 不管是单层还是多层，break是终止本层循环；
2. Break后，不能加执行代码，因为程序不会往下执行；

1）语法：

|  |
| --- |
| 1. *switch :*   *Switch(){*  *Case 常量 ： 语句；break；*  *}*   1. *循环：跳出的是while循环*   *While(){*  *If(){*  *Break;*  *}*  *}* |

2）案例：

continue 语句：

用于在循环语句体中，用于终止某次循环过程，即跳过循环体中尚未执行的语句，接着进行下一次是否执行循环的判定；

特点：

1. 不管是单层或多层，continue是终止本次循环；
2. Continue后，不能加执行代码，因为程序不会往下执行；

return 语句：

return：并非专门用于结束循环的，它的功能是结束一个方法。当一个方法执行到一个return语句时，这个方法将被结束。

与break和continue不同的是，return直接结束整个方法，不管这个return处于多少层循环之内；