Java中的运算符和表达式

1. **表达式**： java中用运算符及操作元连接起来的复合Java规则的式子，称为一般表达式，简称表达式。
2. **运算符**主要包括：
3. 算术运算符：由算术运算符构成的表达式称为算术表达式。

**主要包括： + - \* / %**

**复合：+= -= \*= /= %=**

其中求余运算符 % ：-5%-4 为-1；-5%4为-1；5%-4为1；直接让两个数相除即可，被除数剩余几，结果就是几。

1. 关系运算符（比较运算符）：由关系运算符构成的表达式称为**关系表达式，关系表达式的值为boolean值，true或false**。

主要包括 < <= > >= == !=

1. **逻辑运算符**：用于连接两个Boolean类型的表达式，由逻辑运算符构成的新的表达式称为**逻辑表达式**。



与 或 非 ：**&&** || ！ 异或 **^**

**单： ：& |**

**&&** || 又被称为**短路运算符**。

**异或运算符 : true ^ false为true，true ^ true 为false 等**

**注意 ：异或运算符 ^ 不仅可以对boolean变量操作，也可以对二进制进行操作，即 异或运算符 ^ 既是逻辑运算符，又是位运算符。作为逻辑运算符时返回boolean变量，作为位运算符返回0或1。**

**boolean b1 = false; boolean b2 = false;**

**System.out.println(b1 ^ b2); //false**

**System.out.println(!b1 ^ !b2); //false**

**System.out.println(!b1 ^ b2); //true**

**System.out.println(b1 ^ !b2); //true**

**单位异或：与1异或相当于取反，与0异或相当于不变。**

**一个数同另外一个数异或两次，则结果还是这个数。因为**

**1 ^ 0 = 1;1^0 = 1; 不变 0^0 = 0,0^0 = 0;不变**

**1^1 = 0,0^1=1; 不变 1^0= 1,1^0=1;不变**

双与 ：&& ；单与 ：&，又叫 “位与”运算符。

双或” || ； 单或 | 又叫“位或”运算符

Java中单与、单或也可以用于逻辑运算，但是与双与、双或有些区别：

* 1. **单运算符的两边在任何情况下都会进行运算并判断，而双的运算符两边有可能只会进行一个，先进行左边，如 b1 && b2 ，若b1为false，b2 就不再进行判断了； b1 || b2 ，若b1为true，则b2 不再判断。双运算符正是由于这个特点，又被称为短路逻辑运算符。而 b1 & b2、b1 | b2无论b1为true或者false，b2 都会进行判断。**

**例子：boolean b1 = false;**

**boolean b2 = false;**

**System.out.println((b1 = (8 < 7))&&(b1 = (5 < 8)));**

**System.out.println((b2 = (8 < 7))&(b2 = (5 < 8)));**

**System.out.println(b1);// false**

**System.out.println(b2);// true**

* 1. **单运算符还可以进行位操作，而双运算符不可以进行位操作。**

如 6 | 3 =7；6 & 3 = 2；是合法的，但是6 && 3；6||3都是非法的，因为&&、||两边只能是Boolean变量，不能是int变量。

1. 位运算符：直接对**二进制数据**进行运算。

**位与 & 位或 | 位取反 ~ 异或 ^（见上述）**



**移位运算符: 左移 << 右移>> 无符号右移>>>**

**（箭头指向哪，就是哪移运算符）**

**正数左移一位相当于乘以2，左移一位相当于除以2.**

1. **移位运算符只有这三个，<< >> >>>；都是双目运算符；左边的操作元是被移位数，右边的是移位量，操作元必须是整型类型的数据。**
2. **左移运算符只有一个 << ：左移：无论是正数还是负数，右边补的都是0。**
3. **右移运算符有两个：>> >>> 。**

**>> 是带符号右移 ： 正数就补0，负数就补1。**

**>>> 是无符号右移：左边始终补0。**

**两者的区别是： >> 移位时，若为正数，则高位用0填充，若为负数，高位用1填充。而 >>> 无论是正数还是负数，高位都用0填充，因此称为无符号右移运算符。**

1. **共性：**

**对于byte和short类型数据：a<<n a>>n 或 a>>>n 的结果是int类型。在进行移位之前，系统先将被移位数升级为int类型，升级方法是：正数高位用0填充，负数高位用1填充。升级为int类型后，在进行移位运算。**

1. **对于byte、short、int类型数据，移位时都是4个字节32位的，在移位前都会对移位量进行一次求余运算即**n%32**，本质是移动余数位。故如果移位量是32，那么还是本身，相当于移位0。对于long类型数据，先计算n%64，然后进行移位**余数**位。**
2. 赋值运算符：由赋值运算符构成的表达式称为 赋值表达式。

**= （**注意与等号运算符 == 区别开。**）**

1. **特殊运算符**：
2. **instanceof ：双目运算符，返回boolean值。**
   1. 首先明确instanceof 只是一个运算符，都是小写字母构成的。
   2. 运算符instanceof是双目运算符，左边的操作元是一个**对象名**，右边是一个**类名或接口名**。
   3. **使用方法**：利用**空格**隔开，注意**不是**利用**点 .** ，**也不是**利用**括号**。

如s instanceof Student。

* 1. 当左边的对象是**右边类或者其子类或者接口的实现类的对象**时，该运算的结果为true，否则为false。
  2. 实例：

interface University{}

interface China{}

class Person1{ }

class Student extends Person1 implements University,China{}

public class InstanceofTest {

public static void main(String[] args) {

Student s = new Student();

Person1 p = new Person1();

System.out.println(s instanceof Student);//true

System.out.println(s instanceof Person1);//true

System.out.println(s instanceof Object);//true

System.out.println(p instanceof Object);//true

System.out.println(s instanceof University);//true

System.out.println(s instanceof China);//true

}

}

1. **条件运算符 ？ ：** （**三目运算符**）

**op1？op2：op3 即 （条件表达式）？（表达式1）：（表达式2）**

**要求第一个操作元op1的值必须是boolean类型的数据，若为true则返回表达式1的值，若为false则返回表达式2的值。**

**最常使用的例子是获取两者中的最大值：max = x > y ? x : y ;**

1. 练习题：

不借助第三方变量，如何实现两个变量的数值交换。

1. **方法 1**. int a = 3; int b= 6; a = a+b; b = a-b; a = a-b;

这种方式一般不要使用，因为两个数相加，可能会超出范围，需要强制类型转换。

1. **方法2** . a = a ^ b; b = a ^ b; b = a ^ b;

**原理**：与0异或不变，与1异或取反，故有**任何一个数与另外一个数异或两次，结果都不变**。因为异或满足交换律和结合律，一个数和自己**亦或**结果为0000000；任何一个数与0000000亦或都不变。

**实际开发中上述两种方法都不用，但是面试时一般会考察。**

1. 原码、反码、补码：

正数的原码、反码、补码都一样，不发生变化；

负数：**由原码到补码：先取反再加1；由补码到反码：先减1再取反。符号位保持不变。**

规律：一个数和它的补码之间的关系为**和为-1**。因为-1的补码为FFFFFFFF；对一个数取补码就是利用FFFFFFFF减去即可。

例子: 反码： ~6 = -7 。解释：6为int型：

6 ： 00000000 00000000 00000000 00000110 取反后为：

FF FF FF 11111001 这就是-7 的补码，计算机中负数以**补码形式**存在的。 负数则原码变为80 00 00 00000111 即-7。

1. 不同进制数据之间的转换：

JDK中提供了**进制转换**的方法，且一般都是**直接转换成字符串**的。例如Integer.toBinaryString(int);

**进制转换**：

private static void displayToBinary(int number)

{

transDecToOther(number, 1,1);

}

private static void displayToOctonary(int number)

{

transDecToOther(number, 7, 3);

}

private static void displayToHex(int number)

{

transDecToOther(number, 15, 4);

}

private static void transDecToOther(int number,int base,int offset)

{

if(number == 0) {

System.out.println('0');

return;

}

char[] disp = {'0','1','2','3','4',

'5','6','7','8',

'9','A','B','C',

'D','E','F'};

char[] disp1 = new char[32];

int pos = 32 ;

for(;number != 0;) {

int num1 = number & base;

disp1[--pos] = disp[num1];

number >>>= offset;

}

for(int i = pos;i<32;i++) {

System.out.print(disp1[i]);

}

System.out.println();

return;

}

1. char类型数据和int类型数据运算结果是**int类型**；**char类型**可以利用**int类型变量**进行接收，如：**int s = 's';** //合法的
2. 精度从低到高排列的顺序是：**byte，short，int，long，float，double**。