Java的运算符，分为四类：

算数运算符、关系运算符、逻辑运算符、位运算符。

算数运算符(7)：+ - * / % ++ --

关系运算符(6)：== != > >= < <=

逻辑运算符(6)：&& || ！^ & |

位运算符(7)：& | ~ ^ >> << >>>

Java基本数据类型：

数值类型：

整型：byte、short、int、long

非整型：double、float

非数值类型：

字符:char

布尔:boolean

其中：

1. boolean 类型变量的取值有：ture、false,1字节（8位）
2. char数据类型有：unicode字符,16位
3. byte：一个字节（8位）（-128~127）（-2的7次方到2的7次方-1）
4. short：两个字节（16位）（-32768~32767）（-2的15次方到2的15次方-1）
5. int：四个字节（32位）（一个字长）（-2147483648~2147483647）（-2的31次方到2的31次方-1）
6. long：八个字节（64位）（-9223372036854774808~9223372036854774807）（-2的63次方到2的63次方-1）

7. float：四个字节（32位）（3.402823e+38 ~ 1.401298e-45）（e+38是乘以10的38次方，e-45是乘以10的负45次方）
8. double：八个字节（64位）（1.797693e+308~ 4.9000000e-324）

对应java的基本数据类类型：Integer、Float、Boolean、Character、Double、Short、Byte、Long。

注意：

1. ①数值计算中语法现象——“晋升”，即：byte、short和char(低于int的数据类型)进行算术运算后，结果会自动提升成int类型

1. ②两个char型运算时，自动转换为int型；当char与别的类型运算时，也会先自动转换为int型的，再做其它类型的自动转换；

③算数运算可以加入小括号"()"提高优先级，优先小括号内运算，再其他运算符运算；

④算数运算前操作数变量必须赋值，反之，报语法错误。

一、算数运算符

注：算数运算符操作数必须是数值类型。

分为一元运算符和二元运算符；

一元运算符，只有一个操作数；

二元运算符有两个操作数，运算符在两个操作数之间。

一元运算符：正‘+’，负‘-’，自加‘++’，‘自减‘--‘这四个。

①”++“和”--“运算符，只允许用于数值类型的变量，不允许用于表达式中；

"++"和”--“可以用于数值变量之前或者之后；

两处使用差别：

”++“和”--“用于数值变量之前，在赋值操作中，先对被”++“或”--“操作变量值先加1或者先减1，然后在进行其他的操作；

"++"和”--“用于数值变量之后，在赋值操作中，先用被”++“或”--“的操作变量值进行其他的操作，然后在对其值加1或者减1。

----------见DEMO----------

②二元运算符，加'+'，减'-'，乘'*'，除'/'，求余'%'。

在算数运算符中，”+“，”-“，”*“，”/“完成加减乘除四则运算，%是求两个操作数相除后的余数。

运算规则和数学运算基本相同，在算数运算中，计算时按照从左向右的顺序计算，乘除和求余优先于加减，不同的是，程序中的乘运算符不可省略，在数学中可写为”y=2x“而程序中必须写为”y=2*x“。

当二元运算的两个操作数的数据类型不同时，运算结果的数据类型和参与运算的操作数的数据类型中精度较高(或位数较长)一致。

----------见DEMO----------

转换原则：
从低精度向高精度转换 byte 、short、int、long、float、double。

低精度到高精度会自动转换，而高精度到低精度则要类型强制转换。

二：关系运算符

关系运算符用于比较两个数值之间的大小，其运算结果为一个逻辑类型(boolean布尔类型)的数值。

等于'==',不等于'!=',大于'>',大于等于'>=',小于'<',小于等于'<='

----------代码demo

注： boolean类型只能比较相等和不相等，不能比较大小；

>=的意思是大于或等于，两者成立一个即可,结果为true，<=亦如此；

判断相等的符号是两个等号，而不是一个等号，这个需要特别小心。

实际代码中，数值、变量以及运算结果都可以直接参与比较，只是程序中为了增强可读性，有些时候需要将比较分开进行书写。

比较运算符是程序设计中实现数据比较的基础，也是很多逻辑实现的基础，在程序逻辑中，经常通过比较一定的条件，来判断后续的程序该如何执行。

三：逻辑运算符

逻辑运算符要求操作数的数据类型为逻辑型，其运算结果也是逻辑型值。

逻辑运算的数据和逻辑运算符的运算结果是boolean类型。

逻辑与'&&'，逻辑或'||'，逻辑非'!'，逻辑异或'^'，逻辑与'&'，逻辑或'|'

说明：

两种逻辑与（&&和&）的运算规则基本相同，两种逻辑或（||和|）的运算规则也基本相同。
&和|运算是把逻辑表达式全部计算完，而&&和||运算具有短路计算功能。
对于&来说，如果左侧条件为false，也会计算右侧条件的值，而对于

&&来说，如果左侧的条件为false，则不计算右侧的条件，这种现象被称作短路现象。
所谓短路计算，是指系统从左至右进行逻辑表达式的计算，一旦出现计算结果已经确定的情况，则计算过程即被终止。
对于&&运算来说，只要运算符左端的值为false，则因无论运算符右端的值为true或为false，其最终结果都为false。
所以，系统一旦判断出&&运算符左端的值为false，则系统将终止其后的计算过程；
对于 || 运算来说，只要运算符左端的值为true，则因无论运算符右端的值为true或为false，其最终结果都为true。
所以，系统一旦判断出|| 运算符左端的值为true，则系统将终止其后的计算过程。
利用短路现象：
在程序设计时使用&&和||运算符，不建议使用&和|运算符。

四：位运算符

位运算是以二进制位为单位进行的运算，其操作数和运算结果都是整型值。
位与'&'，位或'|'，位非'~'，位异或'^'，右移'>>'，左移'<<'，0填充的右移'>>>'
位运算的位与'&'，位或'|'，位非'~'，位异或'^'与逻辑运算的相应操作的真值表完全相同，其差别只是位运算操作的操作数和运算结果都是二进制整数，而逻辑运算相应操作的操作数和运算结果都是逻辑值boolean型。

右移是将一个二进制数按指定移动的位数向右移位，移掉的被丢弃，左边移进的部分或者补0（当该数为正时），或者补1（当该数为负时）。这是因为整数在机器内部采用补码表示法，正数的符号位为0，负数的符号位为1。

将一个数左移"<<"会使该值乘以2的幂。

将一个数右移>>"会使该值除以2的幂。

右移（补零）运算符，即无符号右移，">>>"永远不会产生负号，因为其符号位总是被补零。 不论被移动数是正数还是负数，左边移进的部分一律补0。

----------demo----------

五：赋值运算符与其他运算符的简捷使用方式

① 赋值运算符可以与二元算术运算符、逻辑运算符和位运算符组合成简捷运算符，从而可以简化一些常用表达式的书写。

赋值运算符与其他运算符的简捷使用方式

运算符	用法	等价于	说明
+=	s+=i	s=s+i	s，i是数值型
-=	s-=i	s=s-i	s，i是数值型
*=	s*=i	s=s*i	s，i是数值型
/=	s/=i	s=s/i	s，i是数值型
%=	s%=i	s=s%i	s，i是数值型
&=	a&=b	a=a&b	a，b是逻辑型或整型
|=	a|=b	a=a|b	a，b是逻辑型或整型
^=	A^=b	a=a^b	a，b是逻辑型或整型
<<=	s<<=i	s=s<<i	s，i是整型
>>=	s>>=i	s=s>>i	s，i是整型
>>>=	s>>>=i	s=s>>>i	s，i是整型

在程序开发中，大量使用“一元运算符或移位运算符等”该区别简化代码的书写，这样做，因为这样将增加阅读代码的难度，尽量注释。

----------demo----------

② 方括号[]和圆括号（）运算符

方括号[]是数组运算符，方括号[]中的数值是数组的下标，整个表达式就代表数组中该下标所在位置的元素值。

圆括号（）运算符用于改变表达式中运算符的优先级。

③ 字符串加（+）运算符
当操作数是字符串时，加（+）运算符用来合并两个字符串；当加（+）运算符的一边是字符串，另一边是数值时，机器将自动将数值转换为字符串，并连接为一个字符串。

④ 条件运算符(三目运算符)
<表达式1>？<表达式2>：<表达式3>
先计算<表达式1>的值，
当<表达式1>的值为true时，则将<表达式2>的值作为整个表达式的值；
当<表达式1>的值为false时，则将<表达式3>的值作为整个表达式的值。

⑤强制类型转换符
强制类型转换符能将一个表达式的类型强制转换为某一指定数据类型

⑥对象运算符instanceof
对象运算符instanceof用来测试一个指定对象是否是指定类（或它的子类）的实例，若是则返回true，否则返回false。

⑦点运算符
点运算符"."的功能有两个：一是引用类中成员，二是指示包的层次等级。

在实际的开发中，可能在一个运算符中出现多个运算符，计算时，就按照优先级级别的高低进行计算，级别高的运算符先运算，级别低的运算符后计算.

运算符优先级表

类	运算符	关联性
后缀	() [] . (dot operator)	从左到右
一元	++ -- ! ~	从右到左
乘法的	* / %	从左到右
加法的	+ -	从左到右
移位	>> >>> <<	从左到右
关系的	> >= < <=	从左到右
相等	== !=	从左到右
位与	&	从左到右
位异或	^	从左到右
位或	|	从左到右
逻辑与	&&	从左到右
逻辑或	||	从左到右
有条件的	?:	从右到左
赋值	= += -= *= /= %= >>= <<= &= ^==	从右到左
逗号	,	从左到右

优先级自上而下，级别由高到低。

六、demo代码

```
1  package com.cupdata.operator;
2
3  public class Operators {
4      public static void main(String[] args) {
5          Operators.testArithmetic();     //算数一元运算
   符
6          System.out.println("----------------------
   -");
7          Operators.testArithmetic2();     //算数二元运
   算符
8          System.out.println("----------------------
   -");
9          Operators.testRelation();     //关系运算符
10         System.out.println("----------------------
   -");
11         Operators.testBit();     //位运算符
```

```
        System.out.println("----------------------");
        Operators.testPlus();    //加号拼接字符串
        System.out.println("----------------------");

    }

    //算数一元运算符
    public static void testArithmetic() {
        int a = 5;
        int b,c,d,f,g,h;
        b = +a; //正值
        System.out.println("b="+b+",a="+a);
        c = -a; //负值
        System.out.println("c="+c+",a="+a);
        int l = 2;
        d = ++l; //先l=l+1;再d=l
        System.out.println("d="+d+",l="+l);
        int m = 3;
        f = m++;//先f=m;再m=m+1
        System.out.println("f="+f+",m="+m);
        int n = 4;
        g = --n;//先n=n-1;再g=n
        System.out.println("g="+g+",n="+n);
        int o = 6;
        h = o--;//先h=o;再o=o-1
        System.out.println("h="+h+",o="+o);
    }

    //算数二元运算符
    public static void testArithmetic2(){
        int a = 3;
        double b = 3.53;//或者3.53d
        float c = 1.7f;
        int d = 1;

```

```
47          System.out.println("a=" + a +" , b="+b+" , c="+c+" , d="+d);
48          System.out.println("----------------------");
49          System.out.println("int/int : a/d=" + a/d);
50          System.out.println("double/int : b/a=" + b/a);
51          System.out.println("float/int : c/a=" + c/a);
52
53          System.out.println("----------------------");
54          //强制类型转换
55          System.out.println("(int)double/int : b/a=" + (int)(b/a));
56      }
57
58      //关系运算符
59      public static void testRelation(){
60          System.out.println("9.5<8 :" + (9.5<8));
61          System.out.println("8.5<=8.5:" + (8.5<=8.5));
62          System.out.println("'A' < 'a':" + ('A' < 'a'));//字符'A'的Unicode编码值小于字符'a'
63
64          System.out.println("a~z: " + ((int)'a') + "~"+ ((int)'z'));//查看小写字母的Unicode编码值
65          System.out.println("A~Z: " + ((int)'A') + "~"+((int)'Z'));//查看大写字母的Unicode编码值
66      }
67
68      //位运算符
69      public static void testBit(){
70          int a = 15;      //x等于二进制数的00001111
71          int b = 6;       //y等于二进制数的00000110
72          int c = a & b ;    //z等于二进制数的00000110
73          System.out.println("c: " + c);
74
```

```
        System.out.println("1<<3 : " + (1<<3));
        System.out.println("8>>3 : " + (8>>3));
    }

    //加号拼接字符串
    public static void testPlus(){
        String a = "aa";

        int c = 555;
        String b = a+"bbb"+c;
        System.out.println(b);
        System.out.println("----------------------
");

        String aa = "aa";
        String aaa = new String("aa");

        System.out.println("a == aa ? " + (a==aa));
        System.out.println("a == aaa ? " +
(a==aaa));
        System.out.println(a.equals(aaa));
    }

}

```

```
package com.cupdata.process;

public class Process {
    public static void main(String[] args) {
        System.out.println("----------------------
------------------");
        Process.whileDemo();
        System.out.println("----------------------
------------------");
        Process.doWhileDemo();

```

```
9         System.out.println("----------------------
----------------");
10        Process.switchDemo(1);
11        Process.switchDemo(2);
12        Process.switchDemo(3);
13        Process.switchDemo(9);
14        System.out.println("----------------------
----------------");
15    }
16
17
18    public static void whileDemo(){
19        int c = 0 ;
20        while(c < 3) {
21            if (c < 3) {
22                System.out.println("The value of c
is :" + c + " , The loop while continue.....");
23            }else{
24                System.out.println("The value of c
is :" + c + " , This is the last loop !");
25            }
26            c += 1 ;
27        }
28    }
29
30    public static void doWhileDemo(){
31        int a = 3 ;
32        do {
33            System.out.println("The value of c is
:" + a );
34            a--;
35        }while (a >= 0);
36
37        System.out.println("The value of c is :" +
a );
38    }
39
40    public static void switchDemo(int c){
```

```
41          switch (c){
42              case 1:
43                  System.out.println("输入变量值为:
    1");
44                  break;
45              case 2:
46                  System.out.println("输入变量值为:
    2");
47                  break;
48              case 3:
49                  System.out.println("输入变量值为:
    3");
50                  break;
51              default:
52                  System.out.println("别瞎输入！");
53          }
54      }
55  }
56
```