Java运算符大致分为逻辑运算符(&&,||,！)、算数运算符（+, -, \*, / ,+=）、位运算符（^,|,&）、其他运算符（三元运算符）.

### **&（按位与）**

&按位与的运算规则是将两边的数转换为二进制位，然后运算最终值，运算规则即(两个为真才为真)1&1=1 , 1&0=0 , 0&1=0 , 0&0=0

**int** i = 3&5;  
**int** y = 5&7;  
System.***out***.println(**"i="**+i +**"\ty="** +y); *// i=1 y=5*

3的二进制位是0000 0011 ， 5的二进制位是0000 0101 ， 那么就是011 & 101，由按位与运算规则得知，001 & 101等于0000 0001，最终值为1

7的二进制位是0000 0111，那就是111 & 101等于101，也就是0000 0101，故值为5

### **&&（逻辑与）**

&&逻辑与也称为短路逻辑与，先运算&&左边的表达式，一旦为假，后续不管多少表达式，均不再计算，一个为真，再计算右边的表达式，两个为真才为真。

String str = **null**;  
**if**(str != **null** && 1/0 == 0){  
 System.***out***.println(**" && 第一个表达式false，后续不在执行"**);  
}**else** {  
 System.***out***.println(**"执行这里"**);  
}

没有报错，执行else部分

### **|（按位或）**

|按位或和&按位与计算方式都是转换二进制再计算，不同的是运算规则(一个为真即为真)1|0 = 1 , 1|1 = 1 , 0|0 = 0 , 0|1 = 1

**int** x = 6|2;  
System.***out***.println(x); //6

6的二进制位0000 0110 , 2的二进制位0000 0010 , 110|010为110，最终值0000 0110，故6|2等于6

### **||（逻辑或）**

逻辑或||的运算规则是一个为真即为真，后续不再计算，一个为假再计算右边的表达式。

**^（异或运算符）**

^异或运算符顾名思义，异就是不同，其运算规则为1^0 = 1 , 1^1 = 0 , 0^1 = 1 , 0^0 = 0

System.***out***.println(5^9); //12

5的二进制位是0000 0101 ， 9的二进制位是0000 1001，也就是0101 ^ 1001,结果为1100 , 00001100的十进制位是12

### **<<（左移运算符）**

System.***out***.println(5 <<2); *// 20*

5<<2的意思为5的二进制位往左挪两位，右边补0，5的二进制位是0000 0101 ， 就是把有效值101往左挪两位就是0001 0100 ，正数左边第一位补0，负数补1，等于乘于2的n次方，十进制位是20

### **>>（右移运算符）**

System.***out***.println( 5 >>2); *// 1*

凡位运算符都是把值先转换成二进制再进行后续的处理，5的二进制位是0000 0101，右移两位就是把101左移后为0000 0001，正数左边第一位补0，负数补1，等于除于2的n次方，结果为1

### **~（取反运算符）**

System.***out***.println(~5); *// -6*

取反就是1为0,0为1,5的二进制位是0000 0101，取反后为1111 1010，值为-6

### **>>>（无符号右移运算符）**

正数无符号右移

System.***out***.println(15 >>>2); *// 3*

无符号右移运算符和右移运算符的主要区别在于负数的计算，因为无符号右移是高位补0，移多少位补多少个0。

15的二进制位是0000 1111 ， 右移2位0000 0011，结果为3

负数无符号右移

System.***out***.println(-6>>>3); *// 536870911*

-6的二进制是6的二进制取反再加1,6的二进制也就是0000 0000 0000 0000 0000 0000 0000 0110，取反后加1为1111 1111 1111 1111 1111 1111 1111 1010，右移三位0001 1111 1111 1111 1111 1111 1111 1111