**[Java中的<< 和 >> 和 >>> 详细分析](https://www.cnblogs.com/chuijingjing/p/9405598.html)**

**<<表示左移移，不分正负数，低位补0；**

注：以下数据类型默认为byte-8位

左移时不管正负，低位补0

正数：**r = 20 << 2**

　　20的二进制补码：**0001 0100**

　　向左移动两位后：**0101 0000**

　　　　   　　结果：**r = 80**

负数：**r = -20 << 2**

　　-20 的二进制原码 ：**1001 0100**

　-20 的二进制反码 **：1110 1011**

　　-20 的二进制补码 ：**1110 1100**

　　左移两位后的补码：**1011 0000**

　　　　　　　　反码：**1010 1111**

　　　　　　　　原码：**1101 0000**

　　　　　　　　结果：**r = -80**

**>>表示右移，如果该数为正，则高位补0，若为负数，则高位补1；**

注：以下数据类型默认为byte-8位

正数：**r = 20 >> 2**

　　20的二进制补码：**0001 0100**

　　向右移动两位后：**0000 0101**

　　　　　　　结果：**r = 5**

负数：**r = -20 >> 2**

　　-20 的二进制原码 ：**1001 0100**

-20 的二进制反码**：1110 1011**

　　-20 的二进制补码 ：**1110 1100**

　　右移两位后的补码：**1111 1011**

　　　　　　　　反码：**1111 1010**

　　　　　　　　原码：**1000 0101**

　　　　　　　　结果：**r = -5**

**>>>表示无符号右移，也叫逻辑右移，即若该数为正，则高位补0，而若该数为负数，则右移后高位同样补0**

正数：　**r = 20 >>> 2**

　　　　的结果与 r = 20 >> 2 相同；

负数：　**r = -20 >>> 2**

注：以下数据类型默认为int 32位

　　-20:源码：**10000000 00000000 00000000 00010100**

　　　　反码：**11111111  11111111   11111111   11101011**

　　　　补码：**11111111  11111111   11111111   11101100**

　　　　右移：**00111111  11111111   11111111   11111011**

　　　　结果：**r = 1073741819**

& 按位与，判断某一位是否为1

| 按位或

^ 异或