位运算符

- 位运算符 `<<`，`>>`与无符号位运算法 `<<<`，`>>>` 的区别

> `<<` 带符号左移，在低位补0，不论正负数结果相当于乘以2。
>
> `>>`带符号右移，如果为正数则在高位补0(一般情况下会被忽略)，然后往下取直至位数不变，网上说，右移一位相当于除2是因为`int`在结果为小数时会自动转为整数，
>
> `<<`带符号左移，如果为负数，因为负数在计算机存时，是整数取反码加1得到的，因此负数变成正数时，需要减1再取反码，高位补1 ,然后往下取直至位数不变
>
> `eg`：原来的编码为：`11111111111111111111111111111011`，共32位，高位补1后,从原来的编码处拿31位，即去除最后一位`1`,最终结果为`11111111111111111111111111111101`

```
 System.out.println("          9的二进制为 ： " + Integer.toBinaryString(9));
        int a3 = 9 >> 1;
        System.out.println("右移一位的二进制结果为 ： " +
Integer.toBinaryString(a3) + "  转换成十进制为： " + a3);


        System.out.println("          7的二进制为 ： " +
Integer.toBinaryString(7));
        int a4 = 7 >> 1;
        System.out.println("右移一位的二进制结果为 ： " +
Integer.toBinaryString(a4) + "  转换成十进制为： " + a4);
        System.out.println("          5的二进制为  ： " +
Integer.toBinaryString(5));
        int a5 = 5 >> 1;
        System.out.println("右移一位的二进制结果为 ： " +
Integer.toBinaryString(a5) + "  转换成十进制为： " + a5);
        System.out.println("          3的二进制为  ： " +
Integer.toBinaryString(3));
        int a6 = 3 >> 1;
        System.out.println("右移一位的二进制结果为 ： " +
Integer.toBinaryString(a6) + "  转换成十进制为： " + a6);
```

结果为：（根据结果你会发现，高位0被忽略显示了）

```
          9的二进制为 ： 1001
右移一位的二进制结果为 ： 100  转换成十进制为： 4
          7的二进制为 ： 111
右移一位的二进制结果为 ： 11  转换成十进制为： 3
          5的二进制为  ： 101
右移一位的二进制结果为 ： 10  转换成十进制为： 2
          3的二进制为  ： 11
右移一位的二进制结果为 ： 1  转换成十进制为： 1
```

```
        int a1 = -5 >> 1;
        System.out.println(Integer.toBinaryString(-5));
        System.out.println(Integer.toBinaryString(a1));
        System.out.println(a1);
```

结果为

```
11111111111111111111111111111011
11111111111111111111111111111101
-3
```

> `>>>` 无符号右移。无论是正数还是负数，高位通通补0。