3. 运算符

【本节目标】

1. 熟练掌握各种运算符

1. 什么是运算符

计算机的最基本的用途之一就是执行数学运算，比如：

```
int a = 10;
int b = 20;

a + b;
a < b;
```

上述`+`和`<`等就是运算符，即：对操作数进行操作时的符号，不同运算符操作的含义不同。

作为一门计算机语言，Java也提供了一套丰富的运算符来操纵变量。Java中运算符可分为以下：算术运算符(+ - * /)、关系运算符(< > ==)、逻辑运算符、位运算符、移位运算符以及条件运算符等。

2. 算术运算符

1. **基本四则运算符：加减乘除模(+ - * / %)**

```
int a = 20;
int b = 10;

System.out.println(a + b);    // 30
System.out.println(a - b);    // 10
System.out.println(a * b);    // 200
System.out.println(a / b);    // 2
System.out.println(a % b);    // 0 --->模运算相当于数学中除法的余数
```

注意：

- 都是二元运算符，使用时必须要有左右两个操作数
- int / int 结果还是int类型，而且会向下取整

比特就业课

```
int a = 3;
int b = 2;

// 在数学中应该是1.5 但是在Java中输出结果为1 会向下取整，即小数点之后全部舍弃掉了
System.out.println(a / b);

// 如果要得到数学中的结果，可以使用如下方式
double d = a*1.0 / b;
System.out.println(d);
```

- 做除法和取模时，右操作数不能为0

```
int a = 1;
int b = 0;
System.out.println(a / b)

// 运行结果
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at Test.main(Test.java:5)
```

- % 不仅可以对整型取模，也可以对double类型取模，但是没有意义，一般都是对整型取模的

```
System.out.println(11.5 % 2.0);

// 运行结果
1.5
```

- 两侧操作数类型不一致时，向类型大的提升

```
System.out.println(1+0.2);  // +的左侧是int，右侧是double，在加之前int被提升为double
                            // 故：输出1.2
```

2. **增量运算符** `+= -= *= %=`

该种类型运算符操作完成后，会将操纵的结果赋值给左操作数。

```
int a = 1;
a += 2;             // 相当于 a = a + 2
System.out.println(a);  // 输出3

a -= 1;             // 相当于 a = a - 1
System.out.println(a);  // 输出2

a *= 3;             // 相当于 a = a * 3
System.out.println(a);  // 输出6


a /= 3;             // 相当于 a = a / 3
```

比特就业课

```
System.out.println(a);  // 输出2

a %= 3;             // 相当于 a = a % 2
System.out.println(a);  // 输出2
```

注意：只有变量才能使用该运算符，常量不能使用。

3. **自增/自减运算符** `++ --`

++是给变量的值+1，--是给变量的值-1。

```
int a = 1;
a++;   // 后置++  表示给a的值加1，此时a的值为2
System.out.println(a++);  // 注意：后置++是先使用变量原来值，表示式结束时给变量+1，因此输出2
System.out.println(a);    // 输出3


++a;   // 前置++  表示给a的值加1
System.out.println(++a);  // 注意：前置++是先给变量+1，然后使用变量中的值，因此输出5
System.out.println(a);    // 输出5

// --操作符给操作-1，与++含义类似
```

注意：

- 如果单独使用，【前置++】和【后置++】没有任何区别
- 如果混合使用，【前置++】先+1，然后使用变量+1之后的值，【后置++】先使用变量原来的值，表达式结束时给变量+1
- 只有变量才能使用自增/自减运算符，常量不能使用，因为常量不允许被修改

3. 关系运算符

关系运算符主要有六个: `== != < > <= >=`，其计算结果是 true 或者 false 。

```
int a = 10;
int b = 20;
// 注意：在Java中 = 表示赋值，要与数学中的含义区分
//     在Java中 == 表示相等
System.out.println(a == b);   // false
System.out.println(a != b);   // true
System.out.println(a < b);    // true
System.out.println(a > b);    // false
System.out.println(a <= b);   // true
System.out.println(a >= b);   // false
```

注意：当需要多次判断时，不能连着写，比如：3 < a < 5，Java程序与数学中是有区别的

4. 逻辑运算符(重点)

逻辑运算符主要有三个: `&& || !`，运算结果都是 boolean类型。

1. **逻辑与 &&**

比特就业课

语法规则：表达式1 && 表达式2，左右表达式必须是boolean类型的结果。

相当于现实生活中的且，比如：如果是学生，并且 带有学生证 才可以享受半票。

两个表达式都为真，结果才是真，只要有一个是假，结果就是假。

表达式1	表达式2	结果
真	真	真
真	假	假
假	真	假
假	假	假

```
int a = 1;
int b = 2;

System.out.println(a == 1 && b == 2);  // 左为真 且 右为真 则结果为真
System.out.println(a == 1 && b > 100);  // 左为真 但 右为假 则结果为假
System.out.println(a > 100 && b == 2);  // 左为假 但 右为真 则结果为假
System.out.println(a > 100 && b > 100); // 左为假 且 右为假 则结果为假
```

2. **逻辑 ||**

语法规则：表达式1 || 表达式2，左右表达式必须是boolean类型的结果。

相当于现实生活中的或，比如：买房子交钱时，全款 或者 按揭都可以，如果全款或者按揭，房子都是你的，否则站一边去。

表达式1	表达式2	结果
真	真	真
真	假	真
假	真	真
假	假	假

```
int a = 1;
int b = 2;

System.out.println(a == 1 || b == 2);  // 左为真 且 右为真 则结果为真
System.out.println(a == 1 || b > 100);  // 左为真 但 右为假 则结果也为真
System.out.println(a > 100 || b == 2);  // 左为假 但 右为真 则结果也为真
System.out.println(a > 100 || b > 100); // 左为假 且 右为假 则结果为假
```

注意：左右表达式至少有一个位真，则结果为真

比特就业课

3. **逻辑非 !**

   语法规则：！表达式

   真变假，假变真。

表达式	
真	假
假	真

```
int a = 1;
System.out.println(!(a == 1));  // a == 1 为true，取个非就是false
System.out.println(!(a != 1)); // a != 1 为false，取个非就是true
```

4. **短路求值**

   && 和 || 遵守短路求值的规则.

```
System.out.println(10 > 20 && 10 / 0 == 0);         // 打印 false
System.out.println(10 < 20 || 10 / 0 == 0);         // 打印 true
```

   我们都知道, 计算 `10 / 0` 会导致程序抛出异常. 但是上面的代码却能正常运行, 说明 `10 / 0` 并没有真正被求值.

   注意:

   - 对于 &&, 如果**左侧表达式值为 false, 则表达式结果一定是 false, 无需计算右侧表达式**.
   - 对于 ||, 如果**左侧表达式值为 true, 则表达式结果一定是 true, 无需计算右侧表达式**.
   - & 和 | 如果表达式结果为 boolean 时, 也表示逻辑运算. 但与 && || 相比, 它们不支持短路求值.

```
System.out.println(10 > 20 & 10 / 0 == 0);         // 程序抛出异常
System.out.println(10 < 20 | 10 / 0 == 0);         // 程序抛出异常
```

5. 位运算符

Java 中数据存储的最小单位是字节，而数据操作的最小单位是比特位. 字节是最小的存储单位，每个字节是由8个二进制比特位组成的，多个字节组合在一起可以表示各种不同的数据。

位运算符主要有四个: `& | ~ ^`，除 `~` 是一元运算符外，其余都是二元运算符。

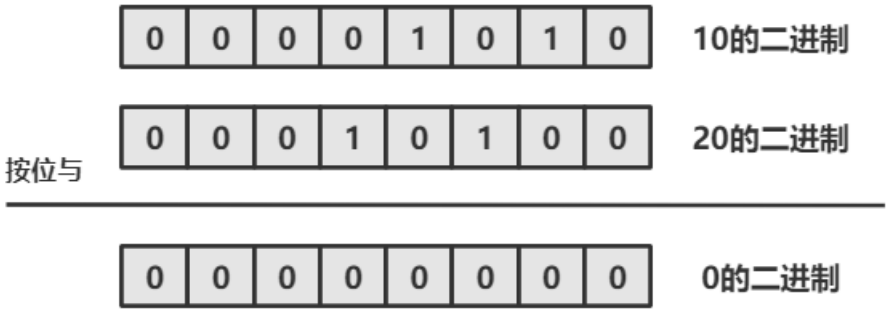
位操作表示 **按二进制位运算**. 计算机中都是使用二进制来表示数据的(01构成的序列), 按位运算就是在按照二进制位的每一位依次进行计算.

1. **按位与 &**: 如果两个二进制位都是 1, 则结果为 1, 否则结果为 0.

```
int a = 10;
int b = 20;
System.out.println(a & b);
```

比特就业课

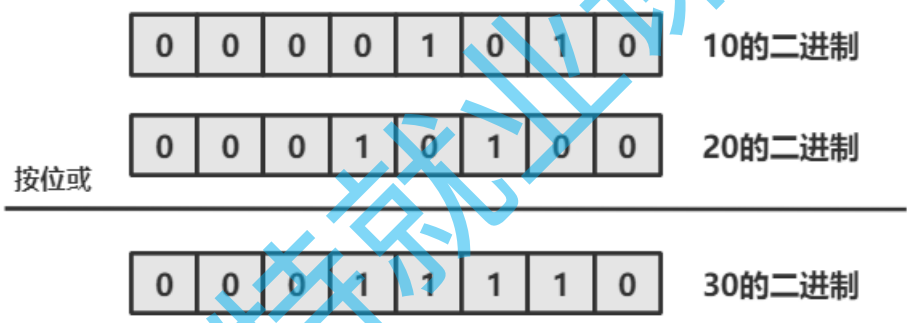
进行按位运算, 需要先把 10 和 20 转成二进制, 分别为 1010 和 10100



2. **按位或 |:** 如果两个二进制位都是 0, 则结果为 0, 否则结果为 1.

```
int a = 10;
int b = 20;
System.out.println(a | b);
```

运算方式和按位于类似.



注意: 当 & 和 | 的操作数为整数(int, short, long, byte) 的时候, 表示按位运算, 当操作数为 boolean 的时候, 表示逻辑运算.

3. **按位取反 ~:** 如果该位为 0 则转为 1, 如果该位为 1 则转为 0

```
int a = 0xf;
System.out.printf("%x\n", ~a)
```

注意:

- 0x 前缀的数字为 十六进制 数字. 十六进制可以看成是二进制的简化表示方式. 一个十六进制数字对应 4 个二进制位.
- 0xf 表示 10 进制的 15, 也就是二进制的 1111
- printf 能够格式化输出内容, %x 表示按照十六进制输出.
- \n 表示换行符

4. **按位异或 ^:** 如果两个数字的二进制位相同, 则结果为 0, 相异则结果为 1.

比特就业课

```
int a = 0x1;
int b = 0x2;
System.out.printf("%x\n", a ^ b);
```

注意：如果两个数相同，则异或的结果为0

6. 移位运算(了解)

移位运算符有三个: `<< >> >>>`，都是二元运算符，且都是按照二进制比特位来运算的。

1. **左移 <<**: 最左侧位不要了, 最右侧补 0.

```
int a = 0x10;
System.out.printf("%x\n", a << 1);

// 运行结果(注意, 是按十六进制打印的)
20
```

注意：向左移位时，丢弃的是符号位，因此正数左移可能会编程负数。

2. **右移 >>**: 最右侧位不要了, 最左侧补符号位(正数补0, 负数补1)

```
int a = 0x10;
System.out.printf("%x\n", a >> 1);

// 运行结果(注意, 是按十六进制打印的)
8

int b = 0xffff0000;
System.out.printf("%x\n", b >> 1);

// 运行结果(注意, 是按十六进制打印的)
ffff8000
```

3. **无符号右移 >>>:** 最右侧位不要了, 最左侧补 0.

```
int a = 0xffffffff;
System.out.printf("%x\n", a >>> 1);

// 运行结果(注意, 是按十六进制打印的)
7fffffff
```

注意:

1. 左移 1 位, 相当于原数字 * 2. 左移 N 位, 相当于原数字 * 2 的N次方.
2. 右移 1 位, 相当于原数字 / 2. 右移 N 位, 相当于原数字 / 2 的N次方.
3. 由于计算机计算移位效率高于计算乘除, 当某个代码正好乘除 2 的N次方的时候可以用移位运算代替.
4. 移动负数位或者移位位数过大都没有意义.

比特就业课

7. 条件运算符

条件运算符只有一个:

`表达式1 ? 表达式2 : 表达式3`

当 `表达式1` 的值为 true 时, 整个表达式的值为 `表达式2` 的值;

当 `表达式1` 的值为 false 时, 整个表达式的值为 `表达式3` 的值.

也是 Java 中唯一的一个 **三目运算符**, 是条件判断语句的简化写法.

```
// 求两个整数的最大值
int a = 10;
int b = 20;
int max = a > b ? a : b;
```

注意:

1. 表达式2和表达式3的结果要是同类型的，除非能发生类型隐式类型转换

```
int a = 10;
int b = 20;
int c = a > b? 1 : 2.0;
```

2. 表达式不能单独存在，其产生的结果必须要被使用。

```
int a = 10;
int b = 20;
a > b? a : b;    // 报错：Error:(15, 14) java: 不是语句
```

8. 运算符的优先级

在一条表达式中，各个运算符可以混合起来进行运算，但是运算符的优先级不同，比如：* 和 / 的优先级要高于 + 和 - ，有些情况下稍不注意，可能就会造成很大的麻烦。

```
// 求a和b的平均值
int a = 10;
int b = 20;
int c = a + (b - a) >> 1;
System.out.println(c);
```

上述表达式中，由于 + 的优先级要高于 >> , 因此a先和b-a的结果做加法，整体为20，最后再进行右移，因此结果为10。

注意：运算符之间是有**优先级**的. 具体的规则我们**不必记忆**. 在可能存在歧义的代码中加上括号即可.

比特就业课

```java
// 求a和b的平均值
int a = 10;
int b = 20;
int c = a + ((b - a) >> 1);
System.out.println(c);
```

比特就业课