**Java 运算符**

计算机的最基本用途之一就是执行数学运算，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。我们可以把运算符分成以下几组：

* 算术运算符
* 关系运算符
* 位运算符
* 逻辑运算符
* 赋值运算符
* 其他运算符

**算术运算符**

算术运算符用在数学表达式中，它们的作用和在数学中的作用一样。下表列出了所有的算术运算符。

表格中的实例假设整数变量A的值为10，变量B的值为20：

|  |  |  |
| --- | --- | --- |
| **操作符** | **描述** | **例子** |
| + | 加法 - 相加运算符两侧的值 | A + B 等于 30 |
| - | 减法 - 左操作数减去右操作数 | A – B 等于 -10 |
| \* | 乘法 - 相乘操作符两侧的值 | A \* B等于200 |
| / | 除法 - 左操作数除以右操作数 | B / A等于2 |
| ％ | 取余 - 左操作数除以右操作数的余数 | B%A等于0 |
| ++ | 自增: 操作数的值增加1 | B++ 或 ++B 等于 21（区别详见下文） |
| -- | 自减: 操作数的值减少1 | B-- 或 --B 等于 19（区别详见下文） |

**实例**

下面的简单示例程序演示了算术运算符。复制并粘贴下面的 Java 程序并保存为 Test.java 文件，然后编译并运行这个程序：

**实例**

public class Test { public static void main(String[] args) { int a = 10; int b = 20; int c = 25; int d = 25; System.out.println("a + b = " + (a + b) ); System.out.println("a - b = " + (a - b) ); System.out.println("a \* b = " + (a \* b) ); System.out.println("b / a = " + (b / a) ); System.out.println("b % a = " + (b % a) ); System.out.println("c % a = " + (c % a) ); System.out.println("a++ = " + (a++) ); System.out.println("a-- = " + (a--) ); // 查看 d++ 与 ++d 的不同 System.out.println("d++ = " + (d++) ); System.out.println("++d = " + (++d) ); } }

[**运行实例 »**](https://www.runoob.com/try/showjava.php?filename=test_operator)

以上实例编译运行结果如下：

a + b = 30

a - b = -10

a \* b = 200

b / a = 2

b % a = 0

c % a = 5

a++ = 10

a-- = 11

d++ = 25

++d = 27

**自增自减运算符**

**1、自增（++）自减（--）运算符**是一种特殊的算术运算符，在算术运算符中需要两个操作数来进行运算，而自增自减运算符是一个操作数。

**实例**

public class selfAddMinus{ public static void main(String[] args){ int a = 3;//定义一个变量； int b = ++a;//自增运算 int c = 3; int d = --c;//自减运算 System.out.println("进行自增运算后的值等于"+b); System.out.println("进行自减运算后的值等于"+d); } }

运行结果为：

进行自增运算后的值等于4

进行自减运算后的值等于2

解析：

* int b = ++a; 拆分运算过程为: a=a+1=4; b=a=4, 最后结果为b=4,a=4
* int d = --c; 拆分运算过程为: c=c-1=2; d=c=2, 最后结果为d=2,c=2

**2、前缀自增自减法(++a,--a):**先进行自增或者自减运算，再进行表达式运算。

**3、后缀自增自减法(a++,a--):**先进行表达式运算，再进行自增或者自减运算 实例：

**实例**

public class selfAddMinus{ public static void main(String[] args){ int a = 5;//定义一个变量； int b = 5; int x = 2\*++a; int y = 2\*b++; System.out.println("自增运算符前缀运算后a="+a+",x="+x); System.out.println("自增运算符后缀运算后b="+b+",y="+y); } }

运行结果为：

自增运算符前缀运算后a=6，x=12

自增运算符后缀运算后b=6，y=10

**关系运算符**

下表为Java支持的关系运算符

表格中的实例整数变量A的值为10，变量B的值为20：

|  |  |  |
| --- | --- | --- |
| **运算符** | **描述** | **例子** |
| == | 检查如果两个操作数的值是否相等，如果相等则条件为真。 | （A == B）为假。 |
| != | 检查如果两个操作数的值是否相等，如果值不相等则条件为真。 | (A != B) 为真。 |
| > | 检查左操作数的值是否大于右操作数的值，如果是那么条件为真。 | （A> B）为假。 |
| < | 检查左操作数的值是否小于右操作数的值，如果是那么条件为真。 | （A <B）为真。 |
| >= | 检查左操作数的值是否大于或等于右操作数的值，如果是那么条件为真。 | （A> = B）为假。 |
| <= | 检查左操作数的值是否小于或等于右操作数的值，如果是那么条件为真。 | （A <= B）为真。 |

**实例**

下面的简单示例程序演示了关系运算符。复制并粘贴下面的Java程序并保存为Test.java文件，然后编译并运行这个程序：

**Test.java 文件代码：**

public class Test { public static void main(String[] args) { int a = 10; int b = 20; System.out.println("a == b = " + (a == b) ); System.out.println("a != b = " + (a != b) ); System.out.println("a > b = " + (a > b) ); System.out.println("a < b = " + (a < b) ); System.out.println("b >= a = " + (b >= a) ); System.out.println("b <= a = " + (b <= a) ); } }

以上实例编译运行结果如下：

a == b = false

a != b = true

a > b = false

a < b = true

b >= a = true

b <= a = false

**位运算符**

Java定义了位运算符，应用于整数类型(int)，长整型(long)，短整型(short)，字符型(char)，和字节型(byte)等类型。

位运算符作用在所有的位上，并且按位运算。假设a = 60，b = 13;它们的二进制格式表示将如下：

A = 0011 1100

B = 0000 1101

-----------------

A&B = 0000 1100

A | B = 0011 1101

A ^ B = 0011 0001

~A= 1100 0011

下表列出了位运算符的基本运算，假设整数变量 A 的值为 60 和变量 B 的值为 13：

|  |  |  |
| --- | --- | --- |
| **操作符** | **描述** | **例子** |
| ＆ | 如果相对应位都是1，则结果为1，否则为0 | （A＆B），得到12，即0000 1100 |
| | | 如果相对应位都是 0，则结果为 0，否则为 1 | （A | B）得到61，即 0011 1101 |
| ^ | 如果相对应位值相同，则结果为0，否则为1 | （A ^ B）得到49，即 0011 0001 |
| 〜 | 按位取反运算符翻转操作数的每一位，即0变成1，1变成0。 | （〜A）得到-61，即1100 0011 |

**逻辑运算符**

下表列出了逻辑运算符的基本运算，假设布尔变量A为真，变量B为假

|  |  |  |
| --- | --- | --- |
| **操作符** | **描述** | **例子** |
| && | 称为逻辑与运算符。当且仅当两个操作数都为真，条件才为真。 | （A && B）为假。 |
| | | | 称为逻辑或操作符。如果任何两个操作数任何一个为真，条件为真。 | （A | | B）为真。 |
| ！ | 称为逻辑非运算符。用来反转操作数的逻辑状态。如果条件为true，则逻辑非运算符将得到false。 | ！（A && B）为真。 |

**实例**

下面的简单示例程序演示了逻辑运算符。复制并粘贴下面的Java程序并保存为Test.java文件，然后编译并运行这个程序：

**实例**

public class Test { public static void main(String[] args) { boolean a = true; boolean b = false; System.out.println("a && b = " + (a&&b)); System.out.println("a || b = " + (a||b) ); System.out.println("!(a && b) = " + !(a && b)); } }

**短路逻辑运算符**

当使用与逻辑运算符时，在两个操作数都为true时，结果才为true，但是当得到第一个操作为false时，其结果就必定是false，这时候就不会再判断第二个操作了。

**实例**

public class LuoJi{ public static void main(String[] args){ int a = 5;//定义一个变量； boolean b = (a<4)&&(a++<10); System.out.println("使用短路逻辑运算符的结果为"+b); System.out.println("a的结果为"+a); } }

运行结果为：

使用短路逻辑运算符的结果为false

a的结果为5

***解析：****该程序使用到了短路逻辑运算符(&&)，首先判断 a<4 的结果为 false，则 b 的结果必定是 false，所以不再执行第二个操作 a++<10 的判断，所以 a 的值为 5。*

**赋值运算符**

下面是Java语言支持的赋值运算符：

|  |  |  |
| --- | --- | --- |
| **操作符** | **描述** | **例子** |
| = | 简单的赋值运算符，将右操作数的值赋给左侧操作数 | C = A + B将把A + B得到的值赋给C |

**条件运算符（?:）**

条件运算符也被称为三元运算符。该运算符有3个操作数，并且需要判断布尔表达式的值。该运算符的主要是决定哪个值应该赋值给变量。

variable x = (expression) ? value if true : value if false

**实例**

**Test.java 文件代码：**

public class Test {

public static void main(String[] args){

int a , b; a = 10;

// 如果 a 等于 1 成立，则设置 b 为 20，否则为 30

b = (a == 1) ? 20 : 30;

System.out.println( "Value of b is : " + b );

// 如果 a 等于 10 成立，则设置 b 为 20，否则为 30

b = (a == 10) ? 20 : 30;

System.out.println( "Value of b is : " + b );

}

}

**instanceof 运算符**

该运算符用于操作对象实例，检查该对象是否是一个特定类型（类类型或接口类型）。

instanceof运算符使用格式如下：

( Object reference variable ) instanceof (class/interface type)

如果运算符左侧变量所指的对象，是操作符右侧类或接口(class/interface)的一个对象，那么结果为真。

下面是一个例子：

String name = "James";

boolean result = name instanceof String; // 由于 name 是 String 类型，所以返回真

如果被比较的对象兼容于右侧类型,该运算符仍然返回true。

看下面的例子：

class Vehicle {} public class Car extends Vehicle { public static void main(String[] args){ Vehicle a = new Car(); boolean result = a instanceof Car; System.out.println( result); } }