Operators

<https://docs.oracle.com/javase/tutorial/java/nutsandbolts/opsummary.html>

**Simple Assignment Operator**

= Simple assignment operator

**Arithmetic Operators**

+ Additive operator (also used

for String concatenation)

- Subtraction operator

\* Multiplication operator

/ Division operator

% Remainder operator

**Unary Operators**

+ Unary plus operator; indicates

positive value (numbers are

positive without this, however)

- Unary minus operator; negates

an expression

++ Increment operator; increments

a value by 1

-- Decrement operator; decrements

a value by 1

! Logical complement operator;

inverts the value of a boolean

**Equality and Relational Operators**

== Equal to

!= Not equal to

> Greater than

>= Greater than or equal to

< Less than

<= Less than or equal to

**Conditional Operators**

&& Conditional-AND

|| Conditional-OR

?: Ternary (shorthand for

if-then-else statement)

**Type Comparison Operator**

instanceof Compares an object to

a specified type

**Bitwise and Bit Shift Operators**

~ Unary bitwise complement

<< Signed left shift

>> Signed right shift

>>> Unsigned right shift

& Bitwise AND

^ Bitwise exclusive OR

| Bitwise inclusive OR

Operator Precedence & Associativity

**Java Operator Precedence Table**

<http://www.cs.bilkent.edu.tr/~guvenir/courses/CS101/op_precedence.html>

|  |  |  |  |
| --- | --- | --- | --- |
| **Precedence** | **Operator** | **Type** | **Associativity** |
| 15 | () [] · | Parentheses Array subscript Member selection | Left to Right |
| 14 | ++ -- | Unary post-increment Unary post-decrement | Right to left |
| 13 | ++ -- + - ! ~ ( *type* ) | Unary pre-increment Unary pre-decrement Unary plus Unary minus Unary logical negation Unary bitwise complement Unary type cast | Right to left |
| 12 | \*  /  % | Multiplication Division Modulus | Left to right |
| 11 | + - | Addition Subtraction | Left to right |
| 10 | << >> >>> | Bitwise left shift Bitwise right shift with sign extension Bitwise right shift with zero extension | Left to right |
| 9 | < <= > >= instanceof | Relational less than Relational less than or equal Relational greater than Relational greater than or equal Type comparison (objects only) | Left to right |
| 8 | == != | Relational is equal to Relational is not equal to | Left to right |
| 7 | & | Bitwise AND | Left to right |
| 6 | ^ | Bitwise exclusive OR | Left to right |
| 5 | | | Bitwise inclusive OR | Left to right |
| 4 | && | Logical AND | Left to right |
| 3 | || | Logical OR | Left to right |
| 2 | ? : | Ternary conditional | Right to left |
| 1 | = += -= \*= /= %= | Assignment Addition assignment Subtraction assignment Multiplication assignment Division assignment Modulus assignment | Right to left |

*Larger number means higher precedence*.

bitwise operator example

<https://www.tutorialspoint.com/java/java_bitwise_operators_examples.htm>

/\*

 \* '-128' is '1000 0000', from there you count back to zero like this:

 \* '-127' is '1000 0001',

 \* '-126' = '1000 0010', '-125' = '1000 0011', ..

 \* '-1' is '1111 1111',

 \* '0' is '0000 0000'

 \* \*/

Java represent negative numbers as 2's complement and all primitive data types except char is signed

Read more: <http://javarevisited.blogspot.com/2015/02/difference-between-right-shift-and.html#ixzz4xBtIhx60>

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | sign- most significant bit |  |  |  |  |  |  |  |  |  |
| 2 to the powe of | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |  |  |
| place value | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| a=5 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 101 |  |
| b = 6 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 110 |  |
|  |  |  |  |  |  |  |  |  |  |  |
| ~a | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 |  |  |
| -1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |  |
| a << 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | multiplies by 2 |  |
| a>>1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |  |  |
| a>>> 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |  |  |
| a=-5 |  |  |  |  |  |  |  |  | 101 | 10 |
|  |  |  |  |  |  |  |  |  | 1111010 | 01111101 |
| a = 2 |  |  |  |  |  |  | 1 |  |  |  |
| b = 6 |  |  |  |  |  | 1 | 1 | 0 |  |  |
| a & b |  |  |  |  |  | 0 | 1 | 0 | a = 2 |  |
| a | b |  |  |  |  |  | 1 | 1 | 0 | return the highest number of the 2 |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
| a = 2 |  |  |  |  |  | 0 | 1 | 0 |  |  |
| b = 6 |  |  |  |  |  | 1 | 1 | 0 |  |  |
| a ^ b |  |  |  |  |  | 1 | 0 | 0 | gives difference between two numbers |  |