CAPÍTULO OPERADORES

Un operador lleva a cabo operaciones sobre uno (operador *unario*), dos (operador *binario*) o tres (operador *ternario*) datos u *operandos* de tipo primitvo devolviendo un valor determinado también de un tipo primitivo. El tipo de valor devuelto tras la evaluación depende del operador y del tipo de los operandos. Por ejemplo, los operadores *aritméticos* trabajan con operandos numéricos, llevan a cabo operaciones aritméticas básicas y devuelven el valor numérico correspondiente. Los operadores se pueden clasificar en distintos grupos según se muestra en los siguientes apartados.

1 OPERADOR ASIGNACIÓN

El operador asignación =, es un operador binario que asigna el valor del término de la derecha al operando de la izquierda. El operando de la izquierda suele ser el identificador de una variable. El término de la derecha es, en general, una expresión de un tipo de dato compatible; en particular puede ser una constante u otra variable. Como caso particular, y a diferencia de los demás operadores, este operador no se evalúa devolviendo un determinado valor.



**Figura 1** Operador asignación

No debe confundirse el operador asignación (=) con el operador relacional de igualdad (==) que se verá más adelante. Además Java dispone de otros operadores que combinan la asignación con otras operaciones (operadores aritméticos *combinados*).

En el siguiente código se muestran algunos ejemplos de uso del operador asignación con datos de distintos tipos:

public class OpAsignacion {

public static void main(String[] args) {

int i,j;

double x;

char c;

boolean b;

String s;

i = 15;

j = i;

x = 12.345;

c = 'A';

b = false;

s = "Hola";

System.out.println("i = " + i);

System.out.println("j = " + j);

System.out.println("x = " + x);

System.out.println("c = " + c);

System.out.println("b = " + b);

System.out.println("s = " + s);

}

}

Salida por pantalla del programa anterior:

$>javac OpAsignacion.java

$>java OpAsignacion

i = 15

j = 15

x = 12.345

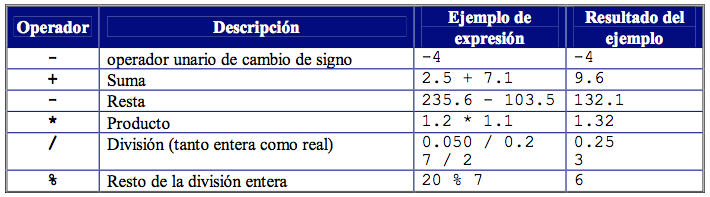
c = A

b = false

s = Hola

2 OPERADORES ARITMÉTICOS

El lenguaje de programación Java tiene varios operadores aritméticos para los datos numéricos enteros y reales. En la siguiente tabla se resumen los diferentes operadores de esta categoría.



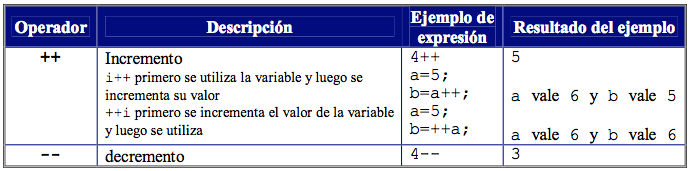
**Figura 2** Operadores aritméticos básicos

El resultado exacto depende de los tipos de operando involucrados. Es conveniente tener en cuenta las siguientes peculiaridades:

* El resultado es de tipo long si, al menos, uno de los operandos es de tipo long y ninguno es real (float o double).
* El resultado es de tipo int si ninguno de los operandos es de tipo long y tampoco es real (float o double).
* El resultado es de tipo double si, al menos, uno de los operandos es de tipo double.
* El resultado es de tipo float si, al menos, uno de los operandos es de tipo float y  ninguno es double.
* El formato empleado para la representación de datos enteros es el complemento a dos. En la aritmética entera no se producen nunca desbordamientos (*overflow*) aunque el resultado sobrepase el intervalo de representación (int o long).
* La división entera se trunca hacia 0. La división o el resto de dividir por cero es una operación válida que genera una excepción ArithmeticException que puede dar lugar a un error de ejecución y la consiguiente interrupción de la ejecución del programa.
* La aritmética real (en coma flotante) puede desbordar al infinito (demasiado grande, overflow) o hacia cero (demasiado pequeño, underflow).
* El resultado de una expresión inválida, por ejemplo, dividir infinito por infinito, no genera una excepción ni un error de ejecución: es un valor NaN (Not a Number).

3 OPERADORES ARITMÉTICOS INCREMENTALES

Los operadores aritméticos incrementales son operadores unarios (un único operando). El operando puede ser numérico o de tipo char y el resultado es del mismo tipo que el operando. Estos operadores pueden emplearse de dos formas dependiendo de su posición con respecto al operando.

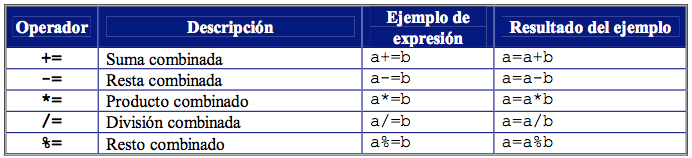


**Figura 3** Operadores aritméticos incrementales

Estos operadores suelen sustituir a veces al operador asignación y también suelen aparecer en bucles for.

4 OPERADORES ARITMÉTICOS COMBINADOS

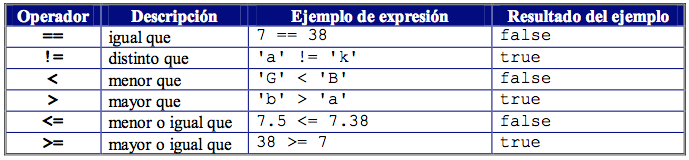
Combinan un operador aritmético con el operador asignación. Como en el caso de los operadores aritméticos pueden tener operandos numéricos enteros o reales y el tipo específico de resultado numérico dependerá del tipo de éstos. En la siguiente tabla se resumen los diferentes operadores de esta categoría.



**Figura 4** Operadores aritméticos combinados

5 OPERADORES DE RELACIÓN

Realizan comparaciones entre datos compatibles de tipos primitivos (numéricos, carácter y booleanos) teniendo siempre un resultado booleano. Los operandos booleanos sólo pueden emplear los operadores de igualdad y desigualdad.

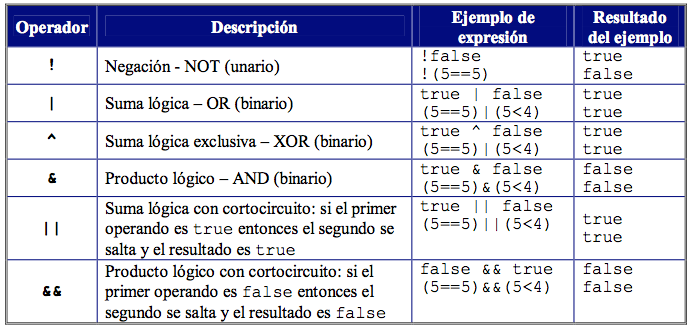


**Figura 5** Operadores de relación

Todos los valores numéricos que se comparan con NaN dan como resultado false excepto el operador != que devuelve true. Esto ocurre incluso si ambos valores son NaN.

6 OPERADORES LÓGICOS O BOOLEANOS

Realizan operaciones sobre datos booleanos y tienen como resultado un valor booleano. En la siguiente tabla se resumen los diferentes operadores de esta categoría.



**Figura 6** Operadores booleanos

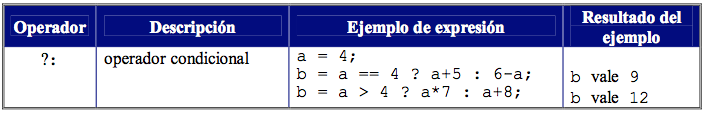
Para mejorar el rendimiento de ejecución del código es recomendable emplear en las expresiones booleanas el operador && en lugar del operador &. En este caso es conveniente situar la condición más propensa a ser falsa en el término de la izquierda. Esta técnica puede reducir el tiempo de ejecución del programa. De forma equivalente es preferible emplear el operador || al operador |. En este caso es conveniente colocar la condición más propensa a ser verdadera en el término de la izquierda.

7 EL OPERADOR CONDICIONAL

Este operador ternario tomado de C/C++ permite devolver valores en función de una expresión lógica. Sintaxis:

expresionLogica ? expresion\_1 : expresion\_2

Si el resultado de evaluar la expresión lógica es verdadero, devuelve el valor de la primera expresión, y en caso contrario, devuelve el valor de la segunda expresión.



**Figura 7** Operador condicional

La sentencia de asignación:

valor = (expresionLogica ? expresion\_1 : expresion\_2);`

como se verá más adelante es equivalente a:

if (expresionLogica)

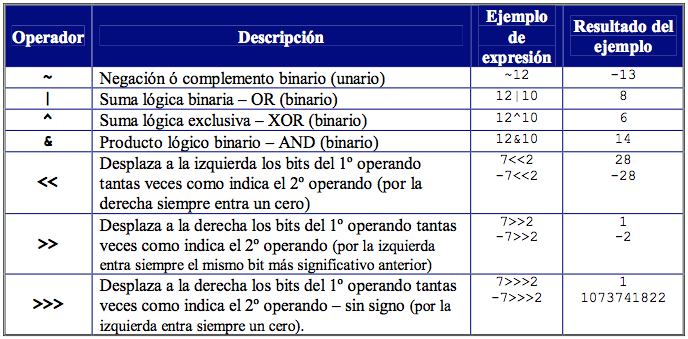
valor = expresion\_1;

else

valor = expresion\_2

8 OPERADORES DE BIT

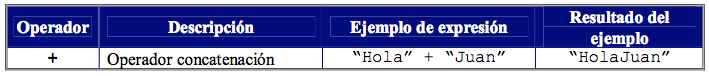
Tienen operandos de tipo entero (o char) y un resultado de tipo entero. Realizan operaciones con dígitos (ceros y unos) de la representación binaria de los operandos. Exceptuando al operador negación, los demás operadores son binarios. En la siguiente tabla se resumen los diferentes operadores de esta categoría.



**Figura 8** Operadores de bit

9 OPERADOR CONCATENACIÓN DE CADENAS

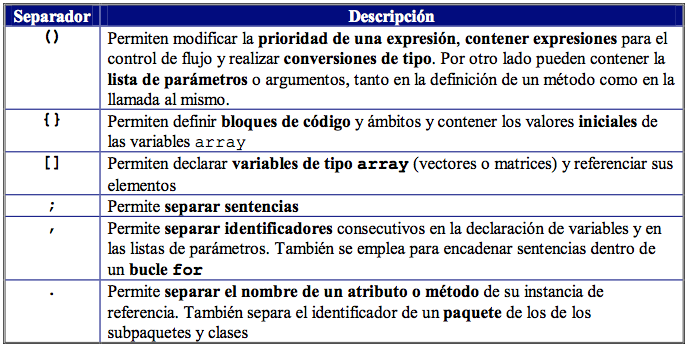
El operador concatenación +, es un operador binario que devuelve una cadena resultado de concatenar las dos cadenas que actúan como operandos. Si sólo uno de los operandos es de tipo cadena, el otro operando se convierte implícitamente en tipo cadena.



**Figura 9** Operador concatenación

10 SEPARADORES

Existen algunos caracteres que tienen un significado especial en el lenguaje Java. En la siguiente tabla se resumen los diferentes separadores que pueden encontrarse en el código fuente de un programa.



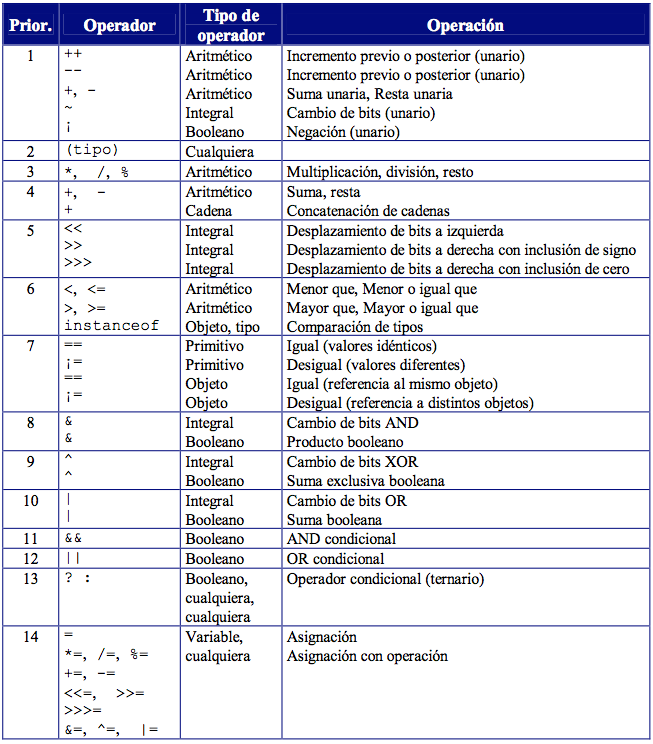
**Figura 10** Separadores en Java

11 EXPRESIONES

Una expresión es una combinación de operadores y operandos que se evalúa generándose un único resultado de un tipo determinado.

12 PRIORIDAD ENTRE OPERADORES

Si dos operadores se encuentran en la misma expresión, el orden en el que se evalúan puede determinar el valor de la expresión. En la siguiente tabla se muestra el orden o prioridad en el que se ejecutan los operadores que se encuentren en la misma sentencia. Los operadores de la misma prioridad se evalúan de izquierda a derecha dentro de la expresión.



**Figura 11** Prioridad de los operadores

[Operadores | Java (arkaitzgarro.com)](https://www.arkaitzgarro.com/java/capitulo-4.html)