



# OPERADORES Y EXPRESIONES

Julio / 2019

INTRODUCCIÓN A LA PROGRAMACIÓN  
INGENIERÍA EN INFORMÁTICA

# Objetivos

Caracterizar los operadores como parte de la construcción de un programa computacional.

Caracterizar las expresiones como parte de la construcción de un programa computacional.

# Sumario

Operadores.  
Expresiones.

# Bibliografía

La Esencia de la Lógica de Programación.  
Manual algoritmos.  
Aprenda Java como si estuviera en  
primero.

# Operadores

Los operadores son signos que nos permiten expresar relaciones entre variables y/o constantes, relaciones de las cuales normalmente se desprende un resultado.

# Operadores

Los mismos se pueden clasificar en :

- ◇ Operadores aritméticos
- ◇ Operadores de asignación
- ◇ Operadores unarios
- ◇ Operador instanceof

# Operadores

Los mismos se pueden clasificar en :

- ◇ Operadores incrementales
- ◇ Operadores relacionales
- ◇ Operadores lógicos

# Operadores

Los mismos se pueden clasificar en :

- ◇ Operadores que actúan a nivel de bits
- ◇ Operador condicional ?:
- ◇ Operador de concatenación de cadenas de caracteres (+)



# Operadores

## Operadores aritméticos

Son operadores binarios (requieren siempre dos operandos) que realizan las operaciones aritméticas habituales: **suma** (+), **resta** (-), **multiplicación** (\*), **división** (/) y **resto de la división** (%).

# Operadores

## Operadores de asignación

Los operadores de asignación permiten asignar un valor a una variable. El operador de asignación por excelencia es el operador igual (=).

La forma general de las sentencias de asignación con este operador es:

```
variable = expression;
```

# Operadores

## Operadores de asignación

Java dispone de otros operadores de asignación. Se trata de versiones abreviadas del operador (=) que realizan operaciones “acumulativas” sobre una variable.

# Operadores

## Operadores de asignación

La siguiente tabla muestra estos operadores y su equivalencia con el uso del operador igual (=).

# Operadores

## Operadores de asignación

Operador	Uso	Equivalente
$+=$	$op1 += op2$	$op1 = op1 + op2$
$-=$	$op1 -= op2$	$op1 = op1 - op2$
$*=$	$op1 *= op2$	$op1 = op1 * op2$

# Operadores

## Operadores de asignación

Operador	Uso	Equivalente
/=	op1/=op2	op1=op1/op2
%=	op1%=op2	op1=op1%op2

# Operadores

## Operadores unarios

Los operadores **más** (+) y **menos** (-) unarios sirven para mantener o cambiar el signo de una variable, constante o expresión numérica. Su uso en Java es el estándar de estos operadores.

# Operadores

## Operador instanceof

El operador **instanceof** permite saber si un objeto pertenece o no a una determinada clase. Es un operador binario cuya forma general es,

```
objectName instanceof ClassName
```

y que devuelve **true** o **false** según el objeto pertenezca o no a la clase.



# Operadores

## Operadores incrementales

Java dispone del operador incremento ( $++$ ) y decremento ( $--$ ). El operador ( $++$ ) incrementa en una unidad la variable a la que se aplica, mientras que ( $--$ ) la reduce en una unidad.

# Operadores

## Operadores incrementales

Estos operadores se pueden utilizar de dos formas:

- ◇ *Precediendo a la variable* (por ejemplo: `++i`). En este caso primero se incrementa la variable y luego se utiliza (ya incrementada) en la expresión en la que aparece.

# Operadores

## Operadores incrementales

Estos operadores se pueden utilizar de dos formas:

- ◇ *Siguiendo a la variable* (por ejemplo:  $i++$ ). En este caso primero se utiliza la variable en la expresión (con el valor anterior) y luego se incrementa.

# Operadores

## Operadores relacionales

Sirven para realizar comparaciones de igualdad, desigualdad y relación de menor o mayor.

El resultado de estos operadores es siempre un valor boolean (true o false) según se cumpla o no la relación considerada.

# Operadores

## Operadores relacionales

Operador	Uso	Cierto si ...
$>$	$op1 > op2$	si $op1$ es mayor que $op2$
$<$	$op1 < op2$	si $op1$ es menor que $op2$
$>=$	$op1 >= op2$	si $op1$ es mayor o igual que $op2$

# Operadores

## Operadores relacionales

Operador	Uso	Resultado
&& (AND)	op1&&op2	true si op1 y op2 son true. Si op1 es false ya no se evalúa op2
(OR)	op1  op2	true si op1 u op2 son true. Si op1 es true ya no se evalúa op2

# Operadores

## Operadores relacionales

Operador	Uso	Resultado
!(negación)	! op1	true si op es false y false si op es true
& (AND)	op1 & op2	true si op1 y op2 son true. Siempre se evalúa op2

# Operadores

## Operadores relacionales

Operador	Uso	Resultado
(OR)	op1	true si op1 u op2 son true. Siempre se evalúa op2



# Operadores

## Operadores lógicos

Los operadores lógicos se utilizan para construir expresiones lógicas, combinando valores lógicos (true y/o false) o los resultados de los operadores relacionales.

# Operadores

## Operadores que actúan a nivel de bits

Es un conjunto de operadores que actúan a nivel de bits. Las operaciones de bits se utilizan con frecuencia para definir señales o flags, esto es, variables de tipo entero en las que cada uno de sus bits indican si una opción está activada o no.

# Operadores

## Operadores que actúan a nivel de bits

Operador	Uso	Resultado
>>	$op1 \gg op2$	Desplaza los bits de op1 a la derecha una distancia op2

# Operadores

## Operadores que actúan a nivel de bits

Operador	Uso	Resultado
<<	$op1 \ll op2$	Desplaza los bits de op1 a la izquierda una distancia op2

# Operadores

## Operadores que actúan a nivel de bits

Operador	Uso	Resultado
$\gg$	$op1 \gg op2$	Desplaza los bits de $op1$ a la derecha una distancia $op2$ (positiva)

# Operadores

## Operadores que actúan a nivel de bits

Operador	Uso	Resultado
&	op1 & op2	Operador AND a nivel de bits
	op1   op2	Operador OR a nivel de bits

# Operadores

## Operadores que actúan a nivel de bits

Operador	Uso	Resultado
$\wedge$	$op1 \wedge op2$	Operador XOR a nivel de bits (1 si sólo uno de los operandos es 1)
$\sim$	$\sim op2$	Operador complemento (invierte el valor de cada bit)

# Operadores

## Operadores que actúan a nivel de bits

Operador	Uso	Equivalente
$\&=$	$op1 \&= op2$	$op1 = op1 \& op2$
$ =$	$op1  = op2$	$op1 = op1   op2$
$\wedge=$	$op1 \wedge= op2$	$op1 = op1 \wedge op2$
$\ll=$	$op1 \ll= op2$	$op1 = op1 \ll op2$



# Operadores

Operadores que actúan a nivel de bits

Operador	Uso	Equivalente
$\gg \gg \gg =$	$op1 \gg \gg \gg = op2$	$op1 =$ $op1 \gg \gg \gg op2$
$\gg =$	$op1 \gg \gg = op2$	$op1 =$ $op1 \gg \gg op2$

# Operadores

## Operador condicional ?:

Este operador, tomado de C/C++, permite realizar bifurcaciones condicionales sencillas. Su forma general es la siguiente:

```
booleanExpression ? res1 : res2
```

donde se evalúa `booleanExpression` y se devuelve `res1` si el resultado es **true** y `res2` si el resultado es **false**.

# Operadores

## Operador de concatenación de cadenas de caracteres (+)

El operador **más** (+) se utiliza también para concatenar cadenas de caracteres. Por ejemplo, para escribir una cantidad con un rótulo y unas unidades puede utilizarse la sentencia:

```
System.out.println("El total asciende a " + result  
+ " unidades");
```

# Operadores

## Precedencia de operadores

El orden en que se realizan las operaciones es fundamental para determinar el resultado de una expresión. (**Ver página 24. Aprenda Java como si estuviera en primero**)

# Sentencias o expresiones

Una **expresión** es un conjunto variables unidos por **operadores**. Son órdenes que se le dan al computador para que realice una tarea determinada.

# Sentencias o expresiones

Una **sentencia** es una **expresión** que acaba en **punto y coma (;)**. Se permite incluir varias sentencias en una línea, aunque lo habitual es utilizar una línea para cada sentencia.

```
i = 0; j = 5; x = i + j; /* Línea compuesta de tres  
sentencias */
```

# Sentencias o expresiones

Según sea el tipo de datos que manipulan, se clasifican las expresiones en:

- ◇ Aritméticas
- ◇ Relacionales
- ◇ Lógicas

# Conclusiones

- ◇ Los operadores relacionales se utilizan con mucha frecuencia en las bifurcaciones y en los bucles.
- ◇ Un programa puede constar desde una expresión o sentencia hasta miles ellas.



# Estudio independiente

Dado el siguiente fragmento de código:

```
int a, b, c;  
a = 1;  
b = 2;  
c = 3;  
a = a + 2;  
b = a + 2 + b;  
c = a + 2 + c;  
a = a / 2;  
b = b / 2;  
c = c / 2;
```

Qué valores quedan en las variables a, b y c ?

# UNIVERSIDAD DE MATANZAS

cosechando el saber

FIN