

# Programación Java

Tutorial Java. Aprende a programar con Java desde cero.




Última entrada

Teoría

Ejercicios

Ejercicios POO

C++

## Operadores Java

La siguiente tabla muestra todos los operadores del lenguaje Java

### OPERADORES JAVA

ARITMÉTICOS	RELACIONALES	LÓGICOS	UNITARIOS	A NIVEL DE BITS	ASIGNACIÓN	CONDICIONAL
+	<	&&	+	&	=	? :
-	<=		-		+=	
*	>	!	++	^	-=	
/	>=		--	<<	*=	
%	!=		~	>>	/=	
	==		!	>>>	%=	
					<<=	
					>>=	
					>>>=	
					&=	
					=	
					^=	

### OPERADORES JAVA ARITMÉTICOS

Los operadores aritméticos en java son:

- + Suma. Los operandos pueden ser enteros o reales
- Resta. Los operandos pueden ser enteros o reales
- \* Multiplicación. Los operandos pueden ser enteros o reales
- / División. Los operandos pueden ser enteros o reales. Si ambos son enteros el resultado es entero. En cualquier otro caso el resultado es real.
- % Resto de la división. Los operandos pueden ser de tipo entero o real.

Ejemplo de operaciones aritméticas

Supongamos que tenemos las siguientes variables:

```
int a = 10, b = 3;
double v1 = 12.5, v2 = 2.0;
char c1='P', c2='T';
```

En la tabla se muestran distintas operaciones aritméticas que podemos hacer con ellas:

Operación	Valor	Operación	Valor	Operación	Valor
a+b	13	v1+v2	14.5	c1	80
a-b	7	v1-v2	10.5	c1 + c2	164
a*b	30	v1*v2	25.0	c1 + c2 + 5	169
a/b	3	v1/v2	6.25	c1 + c2 + '5'	217
a%b	1	v1%v2	0.5		

En las operaciones aritméticas donde intervienen variables de tipo char, el valor utilizado para realizar el cálculo es el valor numérico del carácter correspondiente según la tabla ASCII o UNICODE.

Por eso, según vemos en el ejemplo: c1 + c2 = 164.

c1 = 'P' y el valor numérico de 'P' según la tabla ASCII es 80.

Este sitio utiliza cookies de Google para prestar sus servicios y para analizar su tráfico. Tu dirección IP y user-agent se comparten con Google, junto con las métricas de rendimiento y de seguridad, para garantizar la calidad del servicio, generar estadísticas de uso y detectar y solucionar abusos.

### PUBLICACIONES DEL BLOG



JAVA - Ejercicios básicos resueltos



### ENTRADAS POPULARES

Especificadores de form [Java printf para dar](#)

MÁS INFORMACIÓN ENTENDIDO

Como ya se ha comentado antes, Java es un lenguaje fuertemente tipado. Por eso es importante saber de qué tipo es el resultado de una operación aritmética si intervienen datos de tipos distintos.

En aquellas operaciones en las que aparecen operandos de distinto tipo, java convierte los valores al tipo de dato de mayor precisión de todos los datos que intervienen y este será el tipo del resultado. Esta conversión se realiza de forma temporal, solamente para realizar la operación. Los tipos de datos originales permanecen igual después de la operación. Es muy importante tenerlo en cuenta para poder asignar el resultado de la operación a una variable del mismo tipo.

Respecto a la conversión temporal de tipos: Los tipos short, byte y char se convierten automáticamente a int.

Por ejemplo, dadas las siguientes variables:

```
int i = 7;
double f = 5.5;
char c = 'w';

byte b = 1;
```

Operación	Valor	Tipo
$i + f$	12.5	double
$i + c$	126	int
$i + c - '0'$	78	int
$(i + c) - (2 * f / 5)$	123.8	double
$b + c$	120	int

## OPERADORES JAVA RELACIONALES

Los operadores relacionales comparan dos operandos y dan como resultado de la comparación verdadero ó falso.

Los operadores relacionales en java son:

```
<    Menor que
>    Mayor que
<=   Menor o igual
>=   Mayor o igual
!=    Distinto
==    Igual
```

Los **operandos** tienen que ser de **tipo primitivo**.

Por ejemplo:

```
int a = 7, b = 9, c = 7;
```

Operación	Resultado
$a == b$	false
$a >= c$	true
$b < c$	false
$a != c$	false

## OPERADORES JAVA LÓGICOS

Los operadores lógicos se utilizan con operandos de tipo boolean. Se utilizan para construir expresiones lógicas, cuyo resultado es de tipo true o false.

Los operadores lógicos en Java son:

**&&** AND. El resultado es verdadero si los dos operandos son verdaderos. El resultado es falso en caso contrario. Si el primer operando es falso no se evalúa el segundo, ya que el resultado será falso.

**||** OR. El resultado es falso si los dos operandos son falsos. Si uno es verdadero el resultado es verdadero. Si el primer operando es verdadero no se evalúa el segundo.

**!** NOT. Se aplica sobre un solo operando. Cambia el valor del operando de verdadero a falso y viceversa.

Las definiciones de las operaciones OR, AND y NOT se recogen en unas tablas conocidas como **tablas de verdad**.

A	B	A OR B
F	F	F
F	V	V
V	F	V
V	V	V

A	B	A AND B
F	F	F
F	V	F
V	F	F
V	V	V

A	NOT A
F	V
V	F

F: Falso  
V: Verdadero

datos que se imprimen por pantalla en Java. Este problema se nos plantea por ejempl...

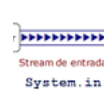
### Calcular el factorial de un número en Java

Factorial en java Programa que calcule el factorial de un número entero que se introduce por teclado. El factorial de un número se expresa m...



Java Ejercicios Básicos Resueltos 1  
Relación Nº 1:  
Ejercicios 1, 2 y 3  
Empezaremos por unos ejercicios

básicos de programas Java con estructura secuencial, es decir, en es...



### Java Scanner para lectura de datos

La clase Scanner se utiliza para la lectura de datos en los programas

Java. Primero veremos varios ejemplos de lectura de datos en Java c...

### Mostrar la tabla de multiplicar de un número en Java

Programa Java que lea un número entero N y muestre la tabla de multiplicar de ese número. Por ejemplo, si se lee el valor 7 se mostrará por...



### Estructuras de control en Java

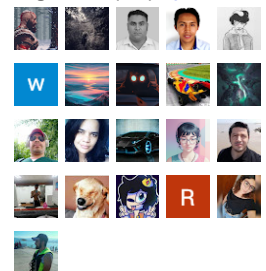
Las estructuras de control determinan la secuencia de ejecución de las sentencias de un programa. Los programas contienen instrucciones...



Programación Java  
Enrique García Hernández

## SEGUIDORES

Seguidores (267) [Siguiendo](#)



[Seguir](#)

## TRANSLATE

Seleccionar idioma ▼

## LENGUAJE C++

[Programacion C++](#)  
Números amigos en C++

Este sitio utiliza cookies de Google para prestar sus servicios y para analizar su tráfico. Tu dirección IP y user-agent se comparten con Google, junto con las métricas de rendimiento y de seguridad, para garantizar la calidad del servicio, generar estadísticas de uso y detectar y solucionar abusos.

[MÁS INFORMACIÓN](#) [ENTENDIDO](#)

Expresión	Resultado
i >= 6 && c != 'w'	false
i >= 6    c != 'w'	true
f < 10 && i > 100	false
!(c != 'p')    i % 2 == 0	false
i + f <= 10	false
i >= 6 && c == 'w' && f == 5	false
c != 'p'    i + f <= 10	true

Las expresiones lógicas en java se evalúan sólo hasta que se ha establecido el valor cierto o falso del conjunto. Cuando, por ejemplo, una expresión va a ser seguro falsa por el valor que ha tomado uno de sus operandos, no se evalúa el resto de expresión.



## OPERADORES JAVA UNITARIOS.

Los operadores unitarios en java son:

-	+	signos negativo y positivo
++	--	incremento y decremento
~		complemento a 1
!		NOT. Negación

Estos operadores **afectan a un solo operando**.

El **operador ++** (operador incremento) incrementa en 1 el valor de la variable.

Ejemplo de operador incremento:

```
int i = 1;
i++; // Esta instrucción incrementa en 1 la variable i.
// Es lo mismo que hacer i = i + 1; i toma el valor 2
```

El **operador --** (operador decremento) decrementa en 1 el valor de la variable.

Ejemplo de operador decremento:

```
int i = 1;
i--; // decrementa en 1 la variable i.
// Es lo mismo que hacer i = i - 1; i toma el valor 0
```

Los operadores incremento y decremento pueden utilizarse como prefijo o sufijo, es decir, pueden aparecer antes o después de la variable.

Por ejemplo:

```
i = 5;
i++; // i vale ahora 6
++i; // i vale ahora 7
```

Cuando estos operadores intervienen en una expresión, si preceden al operando (++i), el valor se modificará antes de que se evalúe la expresión a la que pertenece.

En cambio, si el operador sigue al operando (i++), entonces el valor del operando se modificará después de evaluar la expresión a la que pertenece.

Por ejemplo:

Este sitio utiliza cookies de Google para prestar sus servicios y para analizar su tráfico. Tu dirección IP y user-agent se comparten con Google, junto con las métricas de rendimiento y de seguridad, para garantizar la calidad del servicio, generar estadísticas de uso y detectar y solucionar abusos.

[MÁS INFORMACIÓN](#) [ENTENDIDO](#)

x contiene 3, i contiene 4.

Si las instrucciones son:

```
int x, i = 3;
x = ++i;
```

En esta instrucción primero se incrementa i y el resultado se asigna a x. Por lo tanto, después de ejecutarla:

x contiene 4, i contiene 4.

Otro ejemplo:

```
int i = 1;
System.out.println(i);
System.out.println (++i);
System.out.println (i);
```

Estas instrucciones mostrarían por pantalla:

```
1
2
2
```

En cambio, si se cambia la posición del operador:

```
int i = 1;
System.out.println(i);
System.out.println (i++);
System.out.println (i);
```

Estas instrucciones mostrarían por pantalla:

```
1
1
2
```

El operador complemento a 1 ~ cambia de valor todos los bits del operando (cambia unos por ceros y ceros por unos). Solo puede usarse con datos de tipo entero. El carácter ~ es el ASCII 126.

Por ejemplo:

```
int a = 1, b;
b = ~a;
```

## OPERADORES JAVA A NIVEL DE BITS

Realizan la manipulación de los bits de los datos con los que operan.

Los datos deben ser de tipo entero.

Los operadores a nivel de bits en java son:

```
&    and a nivel de bits
|    or a nivel de bits
^    xor a nivel de bits
<<  desplazamiento a la izquierda, rellenando con ceros a la derecha
>>  desplazamiento a la derecha, rellenando con el bit de signo por la izquierda
>>> desplazamiento a la derecha rellenando con ceros por la izquierda
```

## OPERADORES JAVA DE ASIGNACIÓN.

Se utilizan para asignar el valor de una expresión a una variable.

Los operandos deben ser de tipo primitivo.

Los operadores de asignación en java son:

```
=      Asignación
+=     Suma y asignación
-=     Resta y asignación
*=     Producto y asignación
/=     División y asignación
%=     Resto de la división entera y asignación
<<=   Desplazamiento a la izquierda y asignación
>>=   Desplazamiento a la derecha y asignación
>>>=  Desplazamiento a la derecha y asignación rellenando con ceros
&=     AND sobre bits y asignación
|=     OR sobre bits y asignación
^=     XOR sobre bits y asignación
```

Este sitio utiliza cookies de Google para prestar sus servicios y para analizar su tráfico. Tu dirección IP y user-agent se comparten con Google, junto con las métricas de rendimiento y de seguridad, para garantizar la calidad del servicio, generar estadísticas de uso y detectar y solucionar abusos.

[MÁS INFORMACIÓN](#) [ENTENDIDO](#)

```
x += 3;      ->  equivale a escribir x = x + 3;
y += 3;      ->  equivale a escribir y = x * 3;
```

En general

variable **op**= expresión equivale a: variable = variable **op** expresión

Ejemplo: dadas las siguientes variables

```
int i = 5, j = 7, x = 2, y = 2, z = 2;
float f = 5.5F, g = -3.25F;
```

En la siguiente tabla vemos más ejemplos de asignaciones.

Expresión	Expresión equivalente	Valor final
i += 5	i = i + 5	10
f -= g	f = f - g	8.75
j *= (i - 3)	j = j * (i - 3)	14
f /= 3	f = f / 3	1.833333
i %= (j - 2)	i = i % (j - 2)	0
x *= -2 * (y + z) / 3	x = x * (-2 * (y + z) / 3)	-4

**Importante:** Si los dos operandos de una expresión de asignación (el de la izquierda y el de la derecha) son de distinto tipo de datos es posible que la asignación no se pueda realizar. **El valor de la expresión de la derecha se asignará a la variable que aparece a la izquierda del operador de asignación siempre que sean del mismo tipo o de tipos compatibles.**

Por ejemplo:

```
int n;
double m = 2.5;
n = m + 1; //esta instrucción produce un error.
```

Al intentar realizar la instrucción `n = m + 1`; se producirá un error. El resultado de la operación aritmética `m + 1` es de tipo `double` y no se puede asignar a una variable de tipo `int`.

En este otro caso la asignación sí es posible:

```
double m;
int n = 2;
m = n + 50;
```

El resultado de la operación aritmética `n + 50` es de tipo `int` y `m` es de tipo `double`. Aunque ambos tipos no coinciden, la asignación se puede realizar porque el rango de valores de un `int` es menor que el de un `double` y por lo tanto no puede haber pérdida de información.

En general, en una asignación del tipo `A = B`, si el rango de valores que puede almacenar `A` es mayor que el rango de valores que puede almacenar `B` entonces la asignación se podrá realizar. Por ejemplo, si `A` es de tipo `double` y `B` de tipo `float` la asignación `A = B` se puede hacer pero no se podrá realizar en caso contrario (siendo `A` `float` y `B` `double`).

Por último indicar que en Java están permitidas las **asignaciones múltiples**.

Ejemplo: `a = b = c = 3`; equivale a: `a = 3`; `b = 3`; `c = 3`;

La asignación de valor a las variables en una asignación múltiple se realiza de derecha a izquierda. En el ejemplo, primero se asigna el valor 3 a `c`, a continuación el valor de `c` se asigna a `b` y finalmente el valor de `b` se le asigna a `a`.

## OPERADOR JAVA CONDICIONAL

Se puede utilizar en sustitución de la sentencia de control `if-else`, pero hace las instrucciones menos claras.

El operador condicional java se forman con los caracteres **?** y **:**

Se utiliza de la forma siguiente:

expresión1 **?** expresión2 **:** expresión3

Si expresión1 es cierta entonces se evalúa expresión2 y éste será el valor de la expresión condicional. Si expresión1 es falsa, se evalúa expresión3 y éste será el valor de la expresión condicional.

Ejemplo de operador condicional:

```
int i = 10, j;
j = (i < 0) ? 0 : 100;
```

Esta expresión asigna a `j` el valor 100. Su significado es: si el valor de `i` es menor que 0 asigna a `j` el valor 0, sino asigna a `j` el valor 100. Como `i` vale 10, a `j` se le asigna 100.

Este sitio utiliza cookies de Google para prestar sus servicios y para analizar su tráfico. Tu dirección IP y user-agent se comparten con Google, junto con las métricas de rendimiento y de seguridad, para garantizar la calidad del servicio, generar estadísticas de uso y detectar y solucionar abusos.

MÁS INFORMACIÓN ENTENDIDO

```
j = 0;
else
j = 100;
```

Más ejemplos de operador condicional:

```
int a = 1, b = 2, c = 3;
c += (a > 0 && a <= 10) ? ++a : a/b; // c toma el valor 5
int a = 50, b = 10, c = 20;
c += (a > 0 && a <= 10) ? ++a : a/b; // c toma el valor 25
```

## PRIORIDAD Y ORDEN DE EVALUACIÓN DE LOS OPERADORES EN JAVA


La siguiente tabla muestra todos los operadores Java ordenados de mayor a menor prioridad. La primera línea de la tabla contiene los operadores de mayor prioridad y la última los de menor prioridad. Los operadores que aparecen en la misma línea tienen la misma prioridad.

Una expresión entre paréntesis siempre se evalúa primero y si están anidados se evalúan de más internos a más externos.

Operador	Asociatividad
() [] .	Izquierda a derecha
++ -- ~ !	Derecha a izquierda
new	Derecha a izquierda
* / %	Izquierda a derecha
+ -	Izquierda a derecha
>> >>> <<	Izquierda a derecha
> >= < <=	Izquierda a derecha
== !=	Izquierda a derecha
&	Izquierda a derecha
^	Izquierda a derecha
	Izquierda a derecha
&&	Izquierda a derecha
	Izquierda a derecha
?:	Derecha a izquierda
= += -= *= ...	Derecha a izquierda

Si te ha sido útil compártelo  
Post


## 36 comentarios:

 **marco** 15 de febrero de 2021, 12:54

hola ,me puede explicar esto del %?  
a % b 1  
v1 % v2 0.5


como puede salir este resultado ?  
saludos

[Responder](#)

 **Enrique** 14 de octubre de 2020, 21:31


Me alegro de que os sea útil. Gracias por los comentarios y por seguir el blog.

[Responder](#)

 **AlexT-Dev** 3 de junio de 2020, 16:46

Excelente información.

[Responder](#)

 **Unknown** 22 de agosto de 2018, 7:20

super completo, gracias.

Este sitio utiliza cookies de Google para prestar sus servicios y para analizar su tráfico. Tu dirección IP y user-agent se comparten con Google, junto con las métricas de rendimiento y de seguridad, para garantizar la calidad del servicio, generar estadísticas de uso y detectar y solucionar abusos.

[MÁS INFORMACIÓN](#) [ENTENDIDO](#)

Que buen blog

[Responder](#)



**JL440** 5 de noviembre de 2016, 20:57

Quiero agradecerle el esfuerzo de realizar estos tutoriales, me están siendo de gran utilidad para aprendr JAVA.  
Un saludo

[Responder](#)



**Billi230** 26 de febrero de 2016, 12:28

o mejor esto:  
`int i = 1;`  
`i++;` // Esta instrucción incrementa en 1 la variable i.  
// Es lo mismo que hacer `i = i + 1;` i toma el valor 2

[Responder](#)



**Billi230** 26 de febrero de 2016, 12:26

Como exactamente sale esto?

```
i = 5;
i++; // i vale ahora 6
++i; // i vale ahora 7
Hay una forma sencia para expresarlo?
```

[Responder](#)



**Yesid** 5 de enero de 2016, 19:25

Buenas tardes me gustaría sabe cual es la logia de estas dos expresiones:  
`System.out.println(5&6);` resultado 4  
`System.out.println((12&13) + "\n");` resultado 12

Gracias

[Responder](#)



**Yesid** 5 de enero de 2016, 19:21

Este comentario ha sido eliminado por el autor.

[Responder](#)



**Unknown** 20 de octubre de 2015, 7:46

Hola yo tengo una duda con respecto a las operaciones aritmeticas, de donde sacan los valores de tipo char por ejemplo  
`c=80`  
`c1+c2=164`  
`c1+c2+5=169`  
`c1+c2+'5'=217`  
`i+c=126`  
`i+c-b=78`  
etc  
por favor si alguien me pudiera resolver mi duda se los agradeceria

[Responder](#)

[Respuestas](#)



**Enrique** 21 de octubre de 2015, 20:23

De la tabla ASCII Es el valor numérico que corresponde a ese carácter.



**Unknown** 23 de octubre de 2015, 7:11

Okey muchísimas gracias ahora todo esta mas claro



**Unknown** 20 de octubre de 2015, 7:46

Este comentario ha sido eliminado por el autor.

[Responder](#)

**Anónimo** 19 de septiembre de 2014, 2:31

Bueno info muchas Gracias

[Responder](#)

[Respuestas](#)[Responder](#)**Enrique** 30 de mayo de 2014, 20:51

De nada, espero que te haya servido de ayuda y que sigas visitando el blog. Saludos

**Anónimo** 4 de abril de 2014, 0:59

Por que c toma el valor de 25. no he entendido esa parte??

[Responder](#)[Respuestas](#)[Responder](#)**wvq86** 13 de noviembre de 2014, 5:38

Porque  $c = 20$ ,  $a = 50$  y  $b = 10$   
 $(a > 0 \ \&\& \ a \leq 10)? \ ++a : \ a/b$ ; se lee ¿a es un número entre 1 y 10? Si es cierto haga  $a+1$ , si es falso haga  $a/b$   
 Al evaluar la expresión:  
 $c += (a > 0 \ \&\& \ a \leq 10)? \ ++a : \ a/b$   
 $20 = 20 + 5 = 25$

**Anónimo** 26 de febrero de 2015, 4:33

al tener la "c" el operador "+" al lado "c+", se le sumara la evaluación que se ejecute a la izquierda. " $a/b$ "=5,  
 por lo tanto  $c=20$ ,  $20+5=25$

**raymond** 19 de abril de 2021, 19:04

que grande eres. Gracias

**Unknown** 10 de marzo de 2014, 16:41

Me has despejado las dudas, gracias.

[Responder](#)**Enrique** 19 de febrero de 2014, 20:10

Gracias a todos por los comentarios. Rosendo no temas, se complica un poco pero nada que no sea imposible de superar.  
 Animo y a seguir aprendiendo Java!!!

[Responder](#)**Unknown** 16 de febrero de 2014, 19:09

Muchisimas gracias. Me has animado a aprender java. Tal y como lo explicas parece sencillo.  
 aunque por lo que estoy viendo se complica exponencialmente.

[Responder](#)**Anónimo** 7 de febrero de 2014, 1:15

Excelente información (:

[Responder](#)**Anónimo** 26 de enero de 2014, 20:42

muy util

[Responder](#)**Anónimo** 20 de noviembre de 2013, 16:55

^ quero saber como se usa este oerador para  
 usarlo como operador de potencias..... $3^4$  81  
 que se lee 3 a la cuarta potencia  
 en javascript.....

[Responder](#)[Respuestas](#)[Responder](#)**Unknown** 16 de agosto de 2015, 18:50

en java:  
 $\text{Math.pow}(3, 4);$



en java:  
Math.pow(3, 4);



**Ángel P.C.** 5 de febrero de 2016, 17:02

Muy buena, gracias



**Unknown** 13 de noviembre de 2013, 2:53

muy bueno excelente trabajo muchas gracias hermano

[Responder](#)

**Anónimo** 21 de septiembre de 2013, 17:42

gracias por tu trabajo amigo me ayudo mucho

[Responder](#)

**Anónimo** 28 de agosto de 2013, 3:21

chever gracias por la informacion

[Responder](#)



**Javier Lopez Enamorado** 1 de agosto de 2013, 22:27

Excelente

[Responder](#)

**Anónimo** 21 de marzo de 2013, 16:31

si solo que deberían agregarle mas para que este mas completo

[Responder](#)

**Anónimo** 21 de marzo de 2013, 16:28

buena informacion

[Responder](#)



Escribe tu comentario

[Entrada más reciente](#)

[Inicio](#)

[Entrada antigua](#)

Suscribirse a: [Enviar comentarios \(Atom\)](#)

#### LICENCIA



Programación Java by [Enrique García Hernández](#)

Esta obra está bajo una licencia [Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 España License](#).

Para reconocer la autoría debes poner el enlace <http://puntocomnoesunlenguaje.blogspot.com.es>

Con la tecnología de [Blogger](#).

[Configuración de la privacidad y las cookies](#)

Gestionado por Google Cumple el TCF de IAB. ID de CMP: 300