

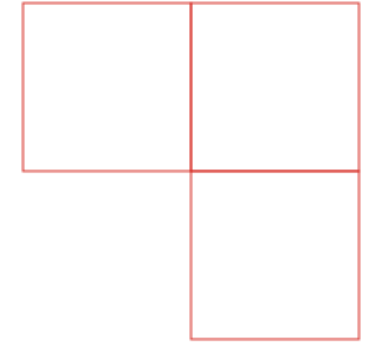
5

Operadores y expresiones

Ve más allá

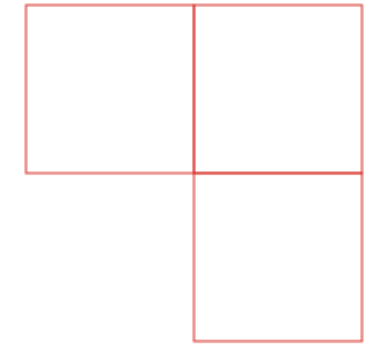
CONTENIDOS

- **Unidad 1: Fundamentos del lenguaje Java**
 1. Nuestro primer programa
 2. Sentencias y errores
 3. Variables y constantes
 4. Tipos de datos
 - 5. Operadores y expresiones**
 6. Entrada y Salida



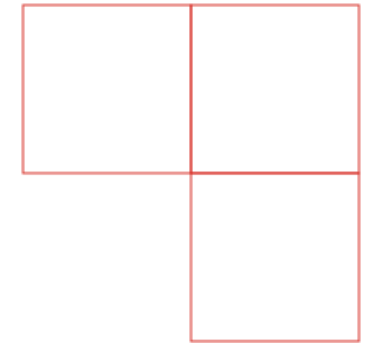
ÍNDICE

- Operadores y expresiones
 - 1.Introducción
 - 2.Aritméticos
 - 3.Relacionales
 - 4.Lógicos
 - 5.Unitarios
 - 6.Bits
 - 7.Asignación
 - 8.Precedencia





Introducción

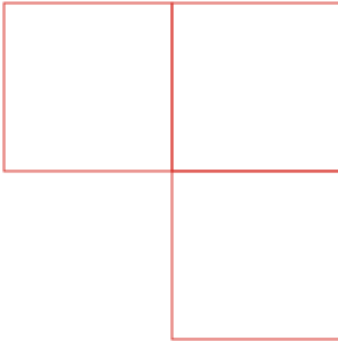


En Java podemos utilizar expresiones que tienen diferentes operadores.

Vamos a conocer los operadores aritméticos, relacionales, lógicos, unitarios, bits y de asignación.

Además, debemos saber que hay ciertas operaciones que se hacen antes que otras debido a su precedencia (importancia).





Aritméticos

- Los operadores **aritméticos** son utilizados para realizar operaciones matemáticas.

Operador	Uso	Operación
+	A + B	Suma
-	A - B	Resta
*	A * B	Multiplicación
/	A / B	División
%	A % B	Módulo o resto de una división entera

- En el siguiente ejemplo se puede observar la utilización de operadores aritméticos:

```
int n1=2, n2;  
n2=n1 * n1;           // n2=4  
n2=n2-n1;             // n2=2  
n2=n2+n1+15;          // n2=19  
n2=n2/n1;             // n2=9  
n2=n2%n1;             // n2=1
```



Aritméticos

Arithmetic Operators Experimentation



+ - * /

After reading the fact sheet, what do you expect a to contain

Tryout the code in bluej. What do you actually get

```
int x = 3;  
int y = 4;  
int xy = 11;  
int cos = 5;
```

```
double a = ((x+1 * 4) * (2+y));
```

```
double b = x + y/2 + xy;
```

```
double c = (4 + cos) / 2;
```

b:

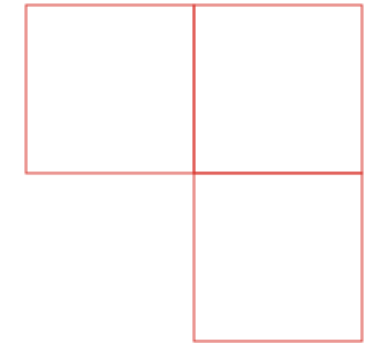
c:

a

b

c





Aritméticos

Integer Arithmetic

```
int pennies = 435;
```

```
int dollars = pennies / 100;
```

```
int cents = pennies % 100;
```

```
pennies / 10
```

```
pennies % 10
```

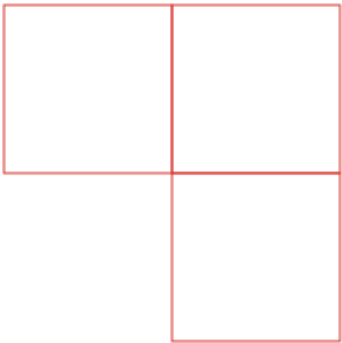


$$435 \div 100 = 4$$

modulus operator or mod
yields remainder

$$435 - 4 \times 100 = 35$$



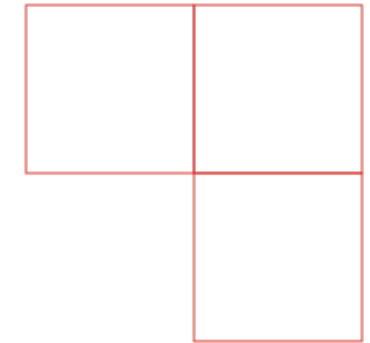


Relacionales

Con los operadores relacionales se puede evaluar la igualdad y la magnitud.
En el siguiente ejemplo se puede observar la utilización de operadores relacionales:

```
int m=2, n=5;
boolean res;
res =m > n;           //res=false
res =m < n;           //res=true
res =m >= n;          //res=false
res =m <= n;          //res=true
res =m == n;          //res=false
res =m != n;          //res=true
```

Operador	Uso	Operación
<	A < B	A menor que B
>	A > B	A mayor que B
<=	A <= B	A menor o igual que B
>=	A >= B	A mayor o igual que B
!=	A != B	A distinto que B
=	A = B	A igual que B



Relacionales

Practice With Relational Operators

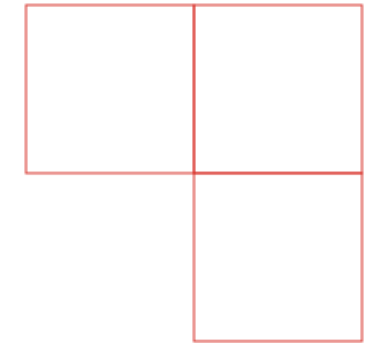
cay = 3 sara = 4

cay + 1 <= sara ☒ true ☐ false

cay + 1 >= sara ☐ true ☒ false

cay + 1 != sara ☐ true ☒ false





Relacionales

Practice With Relational Operators



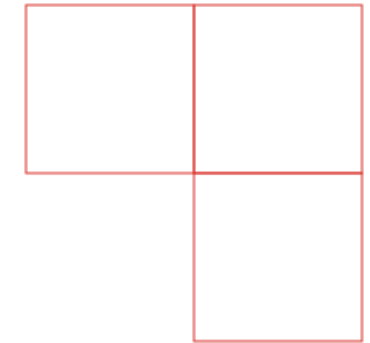
What is the opposite
of $cay + 1 \leq sara$?

$cay + 1 \leq sara$ ☒ true ☐ false

$cay + 1 \geq sara$ ☒ true ☐ false

$cay + 1 \neq sara$ ☐ true ☒ false



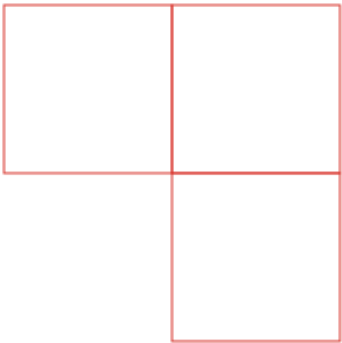


Lógicos

Con los operadores **lógicos** se pueden realizar operaciones lógicas.
En el siguiente ejemplo se puede observar la utilización de operadores lógicos:

```
int m=2, n=5;  
boolean res;  
res =m > n && m >= n;    //res=false  
res =!(m < n || m != n); //res=false
```

Operador	Uso	Operación
&& o &	A&& B o A&B	A AND B. El resultado será true si ambos operandos son true y false en caso contrario.
o	A B o A B	A OR B. El resultado será false si ambos operandos son false y true en caso contrario.
!	!A	Not A. Si el operando es true el resultado es false y si el operando es false el resultado es true.
^	A ^ B	A XOR B. El resultado será true si un operando es true y el otro false, y false en caso contrario.



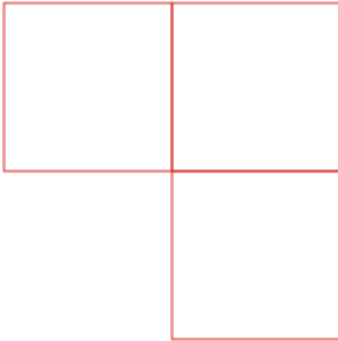
Unitarios

Operadores **unitarios**:

En el siguiente ejemplo se puede observar la utilización de operadores unitarios:

```
int m=2, n=5;  
m++;    // m=3  
n--;    // n=4
```

Operador	Uso	Operación
~	~A	Complemento a 1 de A
-	-A	Cambio de signo del operando
--	A--	Decremento de A
++	A++	Incremento de A
!	! A	Not A (ya visto)



Bit

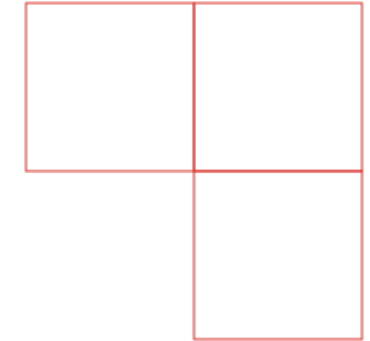
Operadores de **bits**:

En el siguiente ejemplo se puede observar la utilización de operadores de bits:

```
int num=5;
num = num << 1;    // num = 10, equivale a num = num * 2
num = num >> 1;    // num = 5, equivale a num = num / 2
```

Operador	Uso	Operación
&	A & B	AND lógico. A AND B.
	A B	OR lógico. A OR B.
^	A ^ B	XOR lógico. A XOR B.
<<	A << B	Desplazamiento a la izquierda de A B bits rellenando con ceros por la derecha.
>>	A >> B	Desplazamiento a la derecha de A B bits rellenando con el BIT de signo por la izquierda.
>>>	A >>> B	Desplazamiento a la derecha de A B bits rellenando con ceros por la izquierda.

Nota: recuerda que son operaciones en binario



Asignación

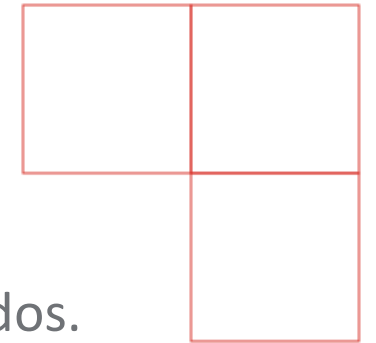
Operadores de **asignación**:

En el siguiente ejemplo se puede observar la utilización de operadores de asignación:

```
int num=5;
```

```
num += 5;    // num = 10, equivale a num = num + 5
```

Operador	Uso	Operación
=	A = B	Asignación. Operador ya visto.
*=	A *= B	Multiplicación y asignación. La operación A*=B equivale a A=A*B.
/=	A /= B	División y asignación. La operación A/=B equivale a A=A/B.
%=	A %= B	Módulo y asignación. La operación A%=B equivale a A=A%B.
+=	A += B	Suma y asignación. La operación A+=B equivale a A=A+B.
-=	A -= B	Resta y asignación. La operación A-=B equivale a A=A-B.



Precedencia

La precedencia de operadores determina el orden en el cual los operadores son evaluados. Los operadores con mayor precedencia son evaluados primero.

Por ejemplo, sea la operación:

```
int valor = 3 + 4 * 5;
```

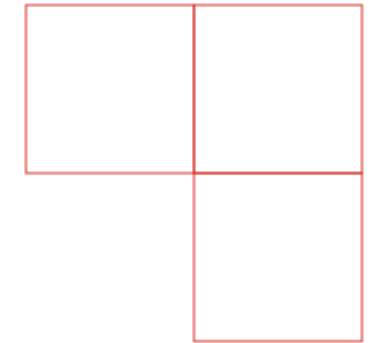
El operador de multiplicación ("*") tiene una precedencia más alta que el operador de suma ("+") y por eso será evaluado primero. Por lo tanto, el resultado es 23 (3 + 20) y no 35 (7 * 5).

Mi consejo es utilizar paréntesis y de esa forma puedes dejar los programas más legibles y controlar las operaciones sin tener que depender de la precedencia.

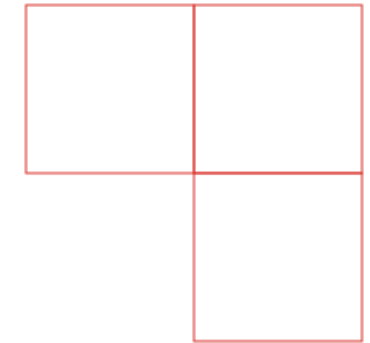


Precedencia

Programación/Operadores y expresiones



	OPERADORES
MAS PRIORIDAD	() [] .
	-- ~ ! ++ --
	new (tipo)expresión
	* / %
	+ -
	<< >> >>>
	< <= > >= instanceof
	== !=
	&
	^
	&&
	?:
MENOS PRIORIDAD	= *= /= %= += -= <<= >>= >>>= &= = ^=



APUNTE...



32 alucinantes documentales sobre videojuegos



**Universidad
Europea**

GRACIAS

Pedro J. Camacho

Universidadeuropea.com

Ve más allá