


Operadores a nivel de bits



Fundamentos

- Realizan operaciones a nivel de bits (operaciones lógicas y desplazamiento de bits).
 - Los que realizan operaciones lógicas pueden aplicarse tanto con valores enteros como boolean.
 - A diferencia de los lógicos, los operadores a nivel de bits no operan en modo cortocircuito (evalúan todos los operandos de la expresión).
- 

Operador &

- Realiza la operación AND a nivel de bits, es decir, aplica la operación AND a cada pareja de bits:

```
int a=7;  
int b=10;  
System.out.println(a&b); //2
```



```
0111 AND  
1010  
----  
0010
```

- Se puede utilizar también con boolean

```
boolean x=true;  
boolean y=false;  
System.out.println(x&y); //false
```

Operador |

- Realiza la operación OR a nivel de bits, aplicando la operación a cada pareja de bits:

```
int a=8;  
int b=10;  
System.out.println(a|b); //10
```



```
1000 OR  
1010  
-----  
1010
```

- Se puede utilizar también con boolean

```
boolean x=true;  
boolean y=false;  
System.out.println(x|y); //true
```

Operador \wedge

➤ Realiza la operación XOR a nivel de bits. Con este tipo de operación, el resultado es false si ambos bits son iguales y true si son diferentes:

```
int a=5;  
int b=9;  
System.out.println(a^b); //12
```



```
0101 XOR  
1001  


---

1100
```

➤ Se puede utilizar también con boolean

```
boolean x=true;  
boolean y=false;  
System.out.println(x^y); //true  
System.out.println(!x^y); //false
```

Precedencia operadores

- La siguiente tabla muestra la precedencia de operadores a nivel de bits en relación al resto de operadores Java

<i>Operador</i>
()
++ --
* / %
+ - !
> >= < <=
== !=
&
^
&&
?:
= += -= *= /= %=

Revisión conceptos



¿Cuál será el resultado de la siguiente operación?

```
int a=12;  
int b=9;  
int c=4;  
System.out.println((a|b^c)<10&c<5);
```

Respuesta

Primeramente se procesará la operación entre paréntesis $(a|b^c)$. Como la operación \wedge tiene prioridad sobre $|$, lo primero será $1001 \text{ XOR } 0100$, que resultará 13 (1101). Después, $1100 \text{ OR } 1101$, que dará como resultado 13 . 13 no es menor que 10 , por lo que el resultado de la comparación es **false**. Después se procesa la operación $\&c<5$, ya que este operador ($\&$) no funciona en modo cortocircuito. Como $<$ tiene más prioridad que $\&$, primero se compara $c<5$ que da como resultado **true**. Así pues, nos queda **false** & **true**, que resulta **false**.