# 3. Operadores lógicos

# Índice

# 3.1 – Lógica booleana

- La lógica booleana es una forma de álgebra que gira en torno a tres palabras sencillas conocidas como operadores booleanos: “o”, “y” y “no”

- Gracias a la lógica booleana vamos a poder establecer condiciones en nuestro código. Seguramente en nuestra infancia hemos jugado al juego del “Quién es quién”. Se trataba de encontrar a un determinado personaje de entre un grupo de gente haciendo para ello una serie de preguntas del tipo: “¿Es hombre?”

Si la contestación es sí, entonces todas las mujeres están descartadas. Otras preguntas similares: ¿es rubio?, ¿usa gafas?, ¿tiene bigote? etc.…

# 3.2 – Incremental, decremental y operadores de asignación

- En Java existen un par de operadores aritméticos que nos van a permitir incrementar (++) o decrementar (--) en una unidad el valor de una variable. Estos dos operadores pueden colocarse antes (prefijos) o después (sufijos) de la variable. De esta forma, podemos incrementar el valor de una variable de tipo in

- Se suelen poner como sufijos

```java
public static void main(String[] args) {

    int numero = 2;

    numero++; //ahora la variable numero vale 3

    numero--; //ahora la variable numero vuelve a valer 2

    System.out.println(numero);

}
```

# 3.2 – Incremental, decremental y operadores de asignación

- Los operadores de asignación no son otra cosa que unir un operador matemático al operador de asignación de Java, el símbolo =

```
public static void main(String[] args) {

    int numero = 2;

    numero = numero + 2; //ahora numero vale 4

    numero += 2; //es igual que lo que tenemos arriba pero simplificado. Ahora numero vale 6

}
```

# 3.3 – Operadores lógicos

- En Java disponemos de los operadores lógicos habituales en lenguajes de programación como son “**es igual**”, “**es distinto**”, **menor**, **menor o igual**, **mayor**, **mayor o igual**, **and** (y), **or** (o) y **not** (no). La sintaxis se basa en símbolos, como veremos a continuación, y cabe destacar que hay que prestar atención a no confundir **==** (equiparación) con **=** (asignación) porque implican distintas cosas

| OPERADOR | DESCRIPCIÓN |
|---|---|
| == | Es igual |
| != | Es distinto |
| <, <=, >, >= | Menor, menor o igual, mayor, mayor o igual |
| && | Operador and (y) |
| \|\| | Operador or (o) |
| ! | Operador not (no) |

Operadores lógicos principales en Java

# 3.3 - Operadores lógicos

- ¿Qué devuelven los distintos operadores? Devuelven lo siguiente:

**>** A > B devuelve cierto si A es mayor estricto que B, de lo contrario devuelve falso.

**<** A < B devuelve cierto si A es menor estricto que B, de lo contrario devuelve falso.

**==** A == B devuelve cierto si A es igual que B, de lo contrario devuelve falso.

**>=** A >= B devuelve cierto si A es mayor o igual que B, de lo contrario devuelve falso.

**<=** A <= B devuelve cierto si A es menor o igual que B, de lo contrario devuelve falso.

**!=** A != B devuelve cierto si A es distinto que B, de lo contrario devuelve falso.

# 3.3 - Operadores lógicos

- El operador **||** (or) se obtiene en la mayoría de los teclados pulsando ALT GR + 1, es decir, la tecla ALT GR y el número 1 simultáneamente

- Los operadores **&&** y **||** se llaman **operadores en cortocircuito** porque si no se cumple la condición de un término no se evalúa el resto de la operación. Por ejemplo: (a == b && c != d && h >= k) tiene tres evaluaciones: la primera comprueba si la variable a es igual a b. Si no se cumple esta condición, el resultado de la expresión es falso y no se evalúan las otras dos condiciones posteriores.

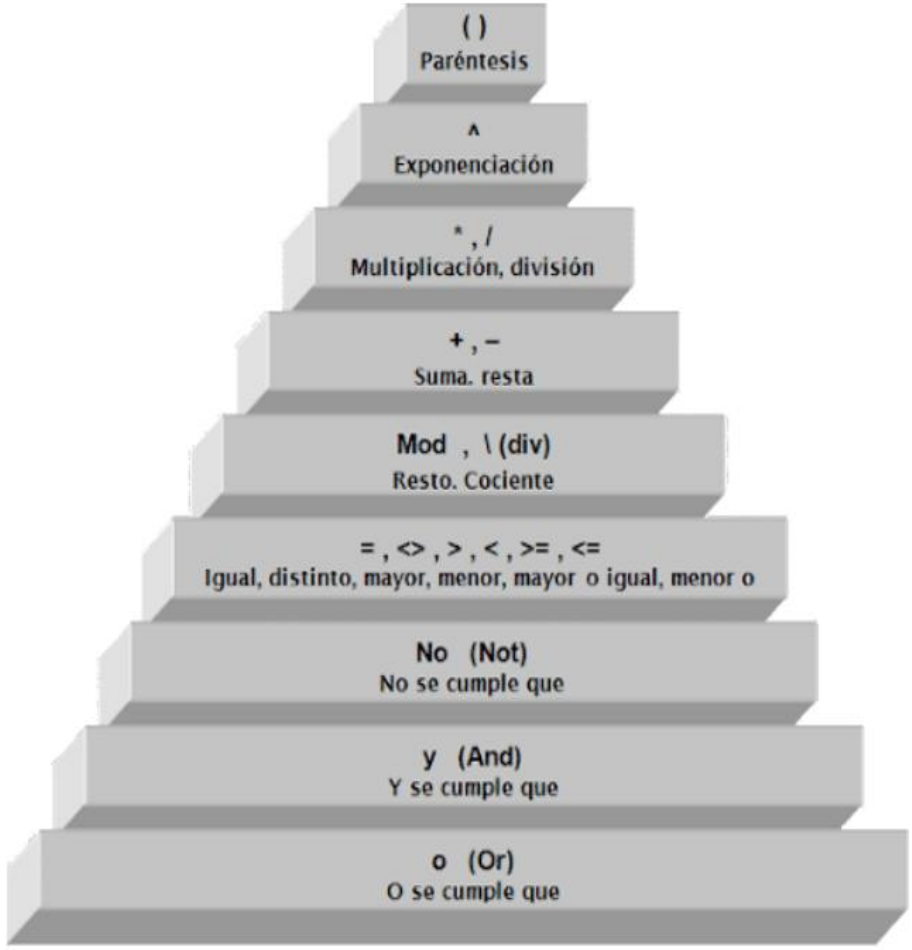
- En un caso como ( a < b || c != d || h <= k) se evalúa si a es menor que b. Si se cumple esta condición el resultado de la expresión es verdadero y no se evalúan las otras dos condiciones posteriores. Si no se cumple la primera condición, pasará a evaluar la segunda, y si tampoco se cumple, pasará a la tercera. Si la tercera sigue sin cumplirse, devolverá falso.

# 3.3 - Operadores lógicos

- El operador ! (not, negación) se recomienda no usarlo hasta que se tenga una cierta destreza en programación. Una expresión como (!esVisible) devuelve false si (esVisible == true), o true si (esVisible == false). En general existen expresiones equivalentes que permiten evitar el uso de este operador cuando se desea

- Los operadores lógicos y matemáticos tienen un orden de prioridad o precedencia. Este es un esquema general que indica el orden en que deben evaluarse en la mayoría de los lenguajes de programación

- Una expresión como A+B == 8 && A-B == 1 siendo A = 3 y B = 5 supondrá que se evalúa primero A+B que vale 8, luego se evalúa A-B que vale -2. Luego se evalúa si se cumple que la primera operación es cierta y luego si la segunda también es cierta, resultando que no, por lo que la expresión es falsa.



# 3.4 – Expresiones booleanas

- Gracias a los operadores lógicos vamos a poder formar expresiones booleanas tomando como operandos otras expresiones booleanas. Son los siguientes:

- AND.- A AND B devuelve cierto si A y B valen cierto y falso en cualquier otro caso.

| A | B | AND |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

# 3.4 – Expresiones booleanas

- OR.- A OR B devuelve cierto si A o B son cierto y falso en cualquier otro caso.

| A | B | OR |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

- NOT.- “not A” devuelve falso si A es cierto y devuelve cierto si A es falso.

| A | NOT |
|---|---|
| 0 | 1 |
| 1 | 0 |

# 3.5 - Ejercicios

1 – Dadas las variables de tipo int con valores A = 5, B = 3, C = -12 indicar si la evaluación de estas expresiones daría como resultado verdadero o falso:

a) A > 3  b) A > C  c) A < C  d) B < C  e) B != C  f) A == 3  g) A * B == 15

h) A * B == -30  i) C / B < A  j) C / B == -10  k) C / B == -4  l) A + B + C == 5

m) (A+B == 8) && (A-B == 2)  n) (A+B == 8) || (A-B == 6)  o) A > 3 && B > 3 && C < 3

p) A > 3 && B >= 3 && C < -3