1.5 Operadores

1. Introducción

Un operador lleva a cabo operaciones sobre uno (**operador unario**), dos (**operador binario**) o tres (**operador ternario**) datos u operandos de tipo primitivo devolviendo un valor determinado también de un tipo primitivo. El tipo de valor devuelto tras la evaluación depende del operador y del tipo de los operandos. Por ejemplo, los operadores aritméticos trabajan con operandos numéricos, llevan a cabo operaciones aritméticas básicas y devuelven el valor numérico correspondiente. Los operadores se pueden clasificar en distintos grupos según se muestra en los siguientes apartados.

2. Operador asignación

El operador asignación **=** es un operador binario que asigna el valor del término de la derecha al operando de la izquierda. El operando de la izquierda es una variable. El término de la derecha es una expresión de un tipo de dato compatible.

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
=	Operador asignación	n = 4	n vale 4

No debe confundirse el operador asignación (=) con el operador relacional de igualdad (==) que se verá más adelante. Además Java dispone de otros operadores que combinan la asignación con otras operaciones (operadores aritméticos combinados).

3. Operadores aritméticos

El lenguaje de programación Java tiene varios operadores aritméticos para los datos numéricos enteros y decimales.

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
-	Operador unario de cambio de signo	-4	-4
+	Suma	2.5 + 7.1	9.6
-	Resta	235.6 - 103.5	132.1
*	Multiplicación	1.2 * 1.1	1.32
/	División	0.050 / 0.2 7 / 2	0.25 3
%	Módulo	20 % 7 14.5 % 2	6 0.5

El resultado exacto depende de los tipos de operandos involucrados. Es conveniente tener en cuenta las siguientes peculiaridades:

- El resultado de una expresión se convierte al tipo más general según el siguiente orden de generalidad:

  byte → short → int → long → float → double

  Teniendo esto en cuenta, tenemos que:

  - El resultado es de tipo long si, al menos, uno de los operandos es de tipo long y ninguno es decimal.
  - El resultado es de tipo int si ninguno de los operandos es de tipo long ni decimal.
  - El resultado es de tipo double si, al menos, uno de los operandos es de tipo double.
  - El resultado es de tipo float si, al menos, uno de los operandos es de tipo float y ninguno es double.
- Con los números enteros, si se divide entre cero, se genera la excepción *ArithmeticException*. Pero si se realiza la división entre cero con decimales, el resultado es infinito (`Infinity`).
- El resultado de una expresión inválida, por ejemplo, dividir infinito por infinito, no genera una excepción ni un error de ejecución: es un valor Not a Number (`NaN`).

```java
package tema1_5_Operadores;

public class ArithmeticOperators {

    public static void main(String[] args) {

        int int1 = 100, int2 = 0;
        double dec1 = 20.36, dec2 = 0;

        System.out.println(int1 / int2); //Genera ArithmeticException
        System.out.println(dec1 / dec2); //Infinity
        System.out.println(dec1 % dec2); //NaN

    }

}
```

Hay que tener en cuenta que el resultado de estos operadores varía notablemente si usamos enteros o si usamos números decimales. Por ejemplo:

```
double result1, d1=14, d2=5;
int result2, i1=14, i2=5;
result1= d1 / d2; //result1=2.8
result2= i1 / i2; //result2=2
```

Es más incluso:

```
double result;
int i1=7,i2=2;
result=i1/i2; //result=3.0
result=(double)i1/i2; //result=3.5
```

El operador del módulo (%) sirve para calcular el resto de una división tanto entera como decimal.

```
int remainder, i1=14, i2=5;
remainder = i1 % i2; //remainder=4
```

En los decimales, el resto se calcula asumiendo que la división produce un resultado entero.

```
double remainder, d1=7.5, d2=2;
remainder=d1 % d2;//remainder=1.5
```

4. Operadores aritméticos incrementales

Los operadores aritméticos incrementales son operadores unarios (un único operando). El operando puede ser numérico o de tipo char y el resultado es del mismo tipo que el operando.

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
++	Incremento	4++	5
- -	Decremento	4 - -	3

En el caso de los caracteres, el incremento/decremento se realiza a su código Unicode. Es decir, si una variable char tiene el valor 'C', su código Unicode es 67. Si se incrementa, su código Unicode pasa a valer 68 que corresponde al valor 'D'.

Estos operadores pueden emplearse de dos formas dependiendo de su posición con respecto al operando:

- si el operador está detrás del operando, primero se utiliza la variable y luego se incrementa/decrementa su valor:
  - Post-incremento: a++
  - Post-decremento: a- -
- si el operador está delante del operando, primero se incrementa/decrementa el valor de la variable y luego se utiliza.
  - Pre-incremento: ++a
  - Pre-decremento: - -a

```
package tema1_5_Operadores;
```

```
public class IncrementalArithmeticOperators {

    public static void main(String[] args) {

        int integer1, integer2;
        char character1, character2;

        character1 = 'C';//Unicode 67
        character1++;
        System.out.println(character1);//Al incrementarse vale 'D', Unicode 68

        /*
         * También se pueden utilizar los caracteres con los operadores
         * aritméticos, pero entonces hace falta usar casting:
         */
        character2 = (char) (character1 + 6);
        System.out.println(character2);//character2 vale 'J', Unicode 74
        integer1 = character2 + 2;
        System.out.println(integer1);//integer1 vale 76
        character2++;
        System.out.println(character2);//character2 vale 'K', Unicode 75
        integer1 = character2;
        System.out.println(integer1);//integer1 vale 75

        integer1 = 5;
        integer2 = integer1++;
        System.out.println(integer1); //integer1 vale 6
        System.out.println(integer2); //integer2 vale 5
        integer1 = 5;
        integer2 = ++integer1;
        System.out.println(integer1); //integer1 vale 6
        System.out.println(integer2); //integer2 vale 6

    }

}
```

5. Operadores aritméticos combinados

Combinan un operador aritmético con el operador asignación. Como en el caso de los operadores aritméticos, pueden tener operandos numéricos enteros o decimales y el tipo específico del resultado numérico dependerá del tipo de éstos.

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
+=	Suma combinada	a+=b	a=a+b
- =	Resta combinada	a - =b	a=a - b
*=	Multiplicación combinada	a*=b	a=a*b
/=	División combinada	a/=b	a=a/b
%=	Resto combinado	a%=b	a=a%b

También se pueden utilizar con caracteres:

```
char character='a';
character+=2; //character vale 'c'
```

6. Operadores relacionales

Realizan comparaciones entre datos compatibles de tipos primitivos (numéricos, carácter y booleanos) teniendo siempre un resultado booleano. Los operandos booleanos sólo pueden emplear los operadores de igualdad y desigualdad.

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
==	Igual que	7 == 38	false
!=	Distinto que	'a' != 'k'	true
<	Menor que	'G' < 'B'	false
>	Mayor que	'b' > 'a'	true
<=	Menor o igual que	7.5 <= 7.38	false
>=	Mayor o igual que	38 >= 7	true

7. Operadores lógicos o booleanos

Las puertas lógicas son circuitos electrónicos capaces de realizar operaciones lógicas básicas:

- **Puerta NOT**: la salida es la inversa de la entrada. Se corresponde con la siguiente tabla de verdad:

A (entrada)	S (salida)
0	1
1	0

- **Puerta AND**: la señal de salida se activa solo cuando se activan todas las señales de entrada. Equivale al producto lógico S = A · B y se corresponde con la siguiente tabla de verdad:

A (entrada1)	B (entrada 2)	S (salida)
0	0	0
0	1	0
1	0	0
1	1	1

- **Puerta OR**: la salida se activa cuando cualquiera de las entradas está activada. Equivale a la suma lógica S = A + B y se corresponde con la siguiente tabla de verdad:

A (entrada1)	B (entrada2)	S (salida)
0	0	0
0	1	1
1	0	1
1	1	1

Los operadores lógicos o booleanos realizan operaciones sobre datos booleanos y tienen como resultado un valor booleano:

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
!	Negación - NOT (unario)	! false ! (5 == 5)	true false
||	Suma lógica - OR (binario)	true || false (5 == 5) || (5 < 4)	true true
&&	Producto lógico - AND (binario)	false && true (5 == 5) && (5 < 4)	false false

El **producto lógico** se realiza con cortocircuito, es decir, si el primer operando es *false* entonces el segundo operando no se evalúa ya que el resultado va a ser de todas maneras *false*. En este caso es conveniente situar la condición más propensa a ser falsa en el término de la izquierda.

La **suma lógica** también se realiza con cortocircuito, es decir, si el primer operando es *true* entonces el segundo operando no se evalúa ya que el resultado va a ser de todas maneras *true*. En este caso es conveniente colocar la condición más propensa a ser verdadera en el término de la izquierda.

Estas técnicas reducen el tiempo de ejecución del programa y ayudan al programador a evitar ciertos errores. Por ejemplo, `5/b==2`, si *b* es una variable de tipo entero y su valor es 0, se genera la excepción *ArithmeticException*. Para evitar este problema, el programador puede hacer lo siguiente: `b!=0 && 5/b==2`. Si *b* contiene el valor 0, `b!=0` dará *false*, entonces `5/b==2` no se evalúa, evitando así la generación de la excepción.

Ejemplos de uso de operadores lógicos o booleanos:

```
boolean adult, younger;
int age = 21;
adult = age >= 18; //adult será true
younger = !adult; //younger será false
```

```java
boolean drivingLicense=true;
int age=20;
boolean canDrive= (age>=18) && drivingLicense;
/*Si la edad es de al menos 18 años y tiene carnet de conducir,
  entonces puede conducir*/
```

```java
boolean snow =true, rain=false, hail=false;
boolean badWeather= snow || rain || hail;
//Si nieva o llueve o graniza, hace mal tiempo
```

8. Operador condicional

Este operador ternario permite devolver valores en función de una expresión lógica. Su sintaxis es la siguiente:

`expresionLogica ? expresion_1 : expresion2`

Si el resultado de evaluar la expresión lógica es verdadero, devuelve el valor de la primera expresión, y en caso contrario, devuelve el valor de la segunda expresión.

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo

? :	Operador condicional	a = 4;
		b = a == 4 ? a+5 : 6-a;
		b = a > 4 ? a*7 : a+8;
			
			b vale 9
			b vale 12

Ejemplo:

```java
pay = (age>18) ? 6000 : 3000;
```

En este caso, si la variable edad es mayor de 18, la paga será de 6000, sino será de 3000.

9. Operador concatenación de cadenas

El operador concatenación, `+`, es un operador binario que devuelve una cadena resultado de concatenar las dos cadenas que actúan como operandos. Si solo uno de los operandos es de tipo cadena, el otro operando se convierte implícitamente en tipo cadena.

Operador	Descripción	Ejemplo de expresión	Resultado del ejemplo
+	Operador concatenación	"Hola" + "Juan"	"HolaJuan"

10. Operadores a nivel de bits

Manipulan los bits de los números.

Operador	Descripción

Operador	Descripción
&	AND
|	OR
~	NOT
^	XOR
>>	Desplazamiento a la derecha
<<	Desplazamiento a la izquierda
>>>	Desplazamiento a la derecha con relleno de ceros
<<<	Desplazamiento a la izquierda con relleno de ceros