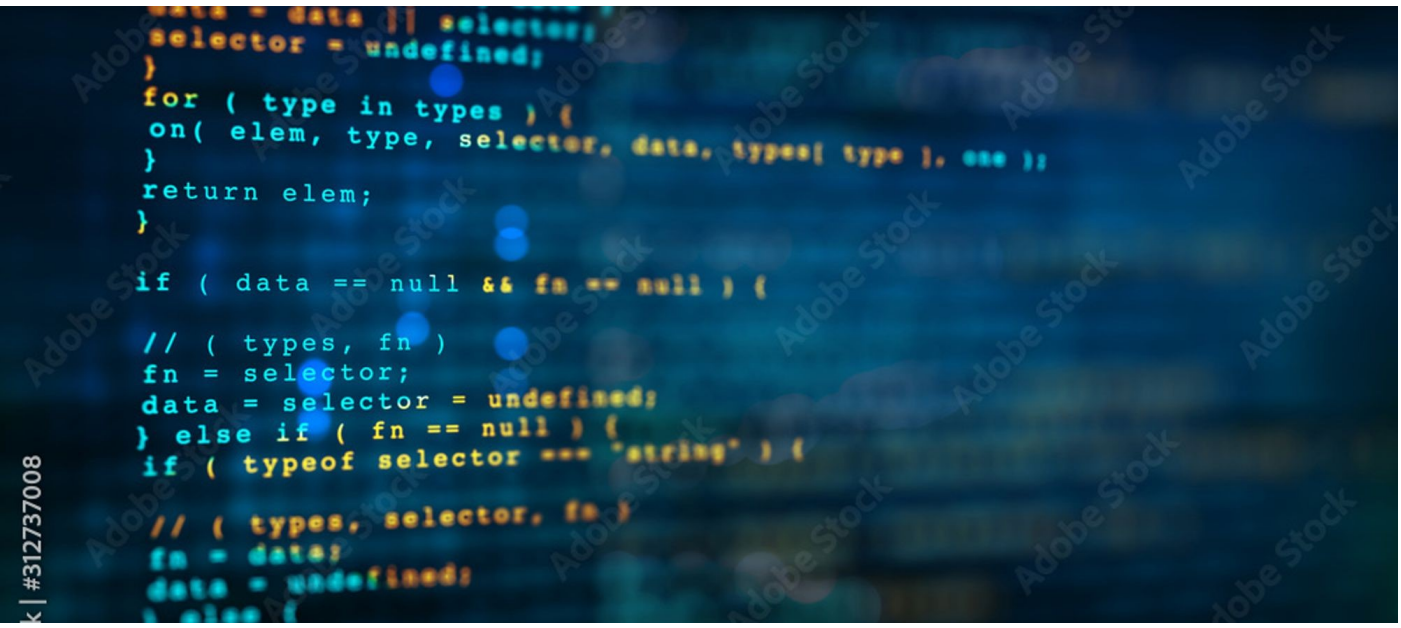

4. Operadores, expresiones y conversiones

Date	Empty
Status	Not started
Type	Empty
Units	2. Estructura de un programa informático

Operadores y expresiones

Existen varios tipos de operadores:

- Aritméticos
- De Asignación
- Relacionales
- Lógicos

Aritméticos

OPERADOR	USO	OPERACIÓN
Monarios		
++	a++	Incremento
--	a--	Decremento
-	-a	Opuesto
Binarios		
+	a+b	Suma
-	a-b	Resta
*	a*b	Producto
/	a/b	División entera si a y son de tipo entero; si no, división real. El signo es el producto de los signos.
%	a%b	Resto de una división entera si a y b son enteros; si no, resto real con cociente entero. El signo es el del primer argumento.

Para los siguientes ejemplos tomase como referencia las siguientes variables:

- byte a = 34;
- int b = 11;
- long c = 9;
- float d = 34.7F;

Los ejemplos de operadores serían:

- a++; // “a” vale 35
- b--; // “b” vale 10
- -c; // “c” vale 8
- ++d; // “d” vale 35.7

- ++d; // "d" vale 35.7
- n1 = 3 * n1; // "n1" vale 36 (3 * n1→3*12→36)
- b=b+5; //"b"vale15(b+5→10+5→15)
- n1 = n1 / b ; // "n1" vale 1 (n1 / b → 12/10 → 1)
- b = b % 3; // "b" vale 1 (b % 3 → 10 % 3 → 1)
- b = -b; // "b" vale -1 (-b→-10)
- d = d / c; // "d" vale 4.4625 (d / c → 35.7 / 8 → 4.4625)
- d = a / c; // "d" vale 4 (a / c → 35 / 8 → 4)
- e = e / 5; // "e" vale -4 (e / 5 → -23 / 5 → -4)
- e = e % 5; // "e" vale -3 (e % 5 → -23 % 5 → -3)
- c = c / -5; // "c" vale -1 (c / -5 → 8 / -5 → -1)
- c = c % -5; // "c" vale 3 (c % -5 → 8 % -5 → 3)
- e = e / -5; // "e" vale 4 (e / -5 → -23 / -5 → 4)
- e = e % -5; // "e" vale -3 (e % -5 → -23 % -5 → -3)

El orden de aplicación es el siguiente:

ORDEN	OPERADORES	ASOCIATIVIDAD
1	( )	
2	* / %	De izquierda a derecha
3	+ -	De izquierda a derecha

Asignación

OPERADOR	USO	OPERACIÓN
+=	a += b	a = a + b
-=	a -= b	a = a - b
*=	a*=b	a = a * b
/=	a/=b	a = a / b
%=	a%=b	a = a % b
&=	a&=b	a = a & b
|=	a|=b	a = a | b
^=	a^=b	a = a ^ b
<<=	a<<=b	a = a << b

>>=	a>>=b	a = a >> b
>>>=	a>>>=b	a = a >>> b

Relacionales o de comparación

OPERADOR	USO	OPERACIÓN
<	a<b	Menor
>	a>b	Mayor
<=	a<=b	Menor o igual
>=	a>=b	Mayor o igual
==	a==b	Igual
!=	a!=b	Distinto

Sean las siguientes variables:

- int a = 20, b = 2,c = 7;
- char d = 'b';
- float e = 20.5F;

Los ejemplos serían los siguientes:

- a+1==3*c* → *20+1==37* → 21==3*7→ 21==21 → true
- d+3>'h' → 98+3>104 → 101>104 → false
- e-b<=a → 20.5-2<=20 → 18.5<=20.0 → true

Este tipo de sentencias, al devolver una variable booleana se utiliza mucho en bloques if como se verá en el siguiente tema

Lógicos

Son aquellos utilizados para evaluar expresiones booleanas. Al igual que los operadores de comparación, son muy utilizados para aplicarlos en estructuras de control

OPERADOR	USO	OPERACIÓN
&&	a && b	Y lógico. Devueve verdadero si lo dos son verdadero, y falso si alguno de los dos es falso. Si el primero es falso, no se evalúa el segundo.
&	a & b	Y lógico. Devuelve verdadero si lo dos son verdadero, y falso si alguno de los dos es falso.

||	a||b	O lógico. Devuelve falso si los dos son falsos, y verdadero si alguno de los dos es verdadero. Si el primero es verdadero, no se evalúa el segundo
|	a|b	O lógico. Devuelve falso si los dos son falsos, y verdadero si alguno de los dos es verdadero
	!(a)	No lógico. Devuelve verdadero si el operando es falso; y falso, si el operando es verdadero
^	a^b	O exclusivo. Devuelve falso si los dos son falsos o verdaderos; verdadero, en caso contrario.

El orden de aplicación de los operadores es:

ORDEN	OPERADORES	ASOCIATIVIDAD
1	( )	
2	&	De izquierda a derecha
3	|	De izquierda a derecha
4	&&	De izquierda a derecha
5	||	De izquierda a derecha

Comparación de cadenas

Cuando se intenta comparar una cadena de texto, no es suficiente con el uso de operadores de comparación, sino que es necesario del uso de métodos que nos ofrece la propia clase String como son equal compareTo(). Para ello tan solo es necesario llamar a los métodos desde la propia palabra.

```
String palabraUno = "Hola"; String palabraDos = "Adios"; Stri
ng palabraTres = "Adios"; boolean comparacion = palabraUno.eq
uals(palabraDos); boolean comparacionDos = palabraDos.equals
(palabraTres); System.out.printf("La comparación es %b %n",co
mparacion); System.out.printf("La comparación es %b %n",compa
racionDos);
```

En el caso del uso del método compareTo, la salida del método será 0 si las dos palabras son iguales

```
String palabraUno = "Hola"; String palabraDos = "Adios"; Stri
```

```
String palabraUno = "Hola"; String palabraDos = "Adios"; Stri
ng palabraTres = "Adios"; boolean comparacion = palabraUno.co
mpareTo(palabraDos)==0; System.out.printf("La comparación es
%b %n",comparacionDos)
```

Conversiones

En algunos casos nos interesará asignar a una variable el valor de otra variable de tipo “superior”. En tal caso, se tendrán que convertir el valor a asignar al tipo de la variable destino con el operador de conversión de tipos.

Sean las siguientes variables:

- int numero = 0;
- float valor = 0;
- char letra = 'a';
- double peso = 64.7;

Para convertir una variable de un tipo en otro tipo se utiliza el casteo directo siempre y cuando el tipo al que se quiera pasar sea mayor que el del origen. Para ello se pone entre paréntesis el tipo al que se quiere pasar.

- numero = (int) peso; // int < double
- letra = (char) peso; // char < double
- valor = (float) peso; // float < double

De no ser así, se utilizará el parseo, método propio de cada uno de los tipos.

Otros operadores

Existen otro tipo de operadores en Java que se irán viendo a lo largo del curso. Entre ellos están:

- []: Declaración de variables de tipo Array. Este tipo de variables permite declarar conjunto de datos en un mismo elemento.
- new: Declaración para crear un tipo de objeto complejo. Se utiliza para instanciar objetos de tipos concretos.
- .: Carácter utilizado para acceder a los mátodos y/o variables que están declarados dentro de una clase.

- this: palabra reservada para llamar a métodos y/o variables declarados en la propia clase.
- <tipo>: carácteres utilizados para marcár un tipo concreto en una variable de tipo colección.