Contenido

1 ASIGNACIÓN.

=

- **Asigna un valor a una variable**
  Variable = expresión
- **Término de la derecha compatible con el de la izquierda**

Ejemplos:

```
int a, b;
a = 100;
b = a + 200:
```

Ejemplos en: opAsignacion.java

2 Aritméticos

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
*	Producto	1.2 * 1.1	1.32
/	División	0.050 / 0.2	0.25
%	resto de la división entera	20 % 7	6

```
int resto = 100 % 3;
System.out.println("Resto: " + resto);
System.out.println("Resto: " + (100 % 3) );

int a=20, b=30;
System.out.println("Suma: " + (a+b) );
System.out.println("Suma: " + a + b );

b=0;
System.out.println(" a / b : "+ (a/b));
//error java.lang.ArithmeticException: / by zero

double x=123.4;
x= x/0;
```

```
System.out.println(" Valor de x: " + x);
//Valor de x: Infinity --Division por 0
```

Ejemplos en `OpAritmeticos.java`

Operadores unarios

++a, a++	Incrementa en 1 la variable **a**
--a, a--	Decrementa en 1 la variable **a**
-a	Cambio de signo para la variable **a**

Prefija:	Postfija:
`x = 3;`	`x = 3;`
`y = ++x;`	`y = x++;`
`// x vale 4, y vale 4`	`// x vale 4, y vale 3`

Prueba en: `opIncrementales.java`

Leer por teclado 2 números enteros y mostrar la suma, el producto, la división y la resta.

Leer por teclado 2 números decimales y mostrar la suma, el producto, la división y la resta.

Controlar las excepciones, si se produce algún cálculo erróneo (la división) mostrar mensaje de error y continuar la ejecución

(`LeerDosNumeros.java`)

Para mostrar decimales con formato. Ejemplo:

```
double d = 10000;
System.out.printf("Valor1: %,9.2f %n", d);
System.out.printf("Valor2: %f %n", d);
System.out.printf("Valor3: %,12.2f %n", d);
```

```
Valor1: 10.000,00
Valor2: 10000,000000
Valor3:    10.000,00
```

3 Operadores opera y asigna:

Operación	Operador	Utilización	Operación equivalente
Suma	+=	A += B	A = A + B
Resta	-=	A -= B	A = A - B
Multiplicación	*=	A *= B	A = A * B
División	/=	A /= B	A = A / B
Resto de división	%=	A %= B	A = A % B

Ejemplos en: `OpCombinados.java`

```
int a = 10;
int b = 20;
System.out.println("Antes de sumar: " + a);
a += b;
System.out.println("Despues de sumar: " + a);
```

```
Antes de sumar: 10
Despues de sumar: 30
```

Ejercicios (`EjercicioOperacionesAritmeticas.java`):

```
int a = 3, b = 5, c = -2;
a = a * 2 + ++b;
b++;
a--;
b += c++;

VALORES DE a, b y c ¿?
```

```
a = -3; b = 7; c = -2;
a += b++ - c--;
b *= --c;
c += a * b++;

VALORES DE a, b y c ¿?
```

```
a = -3; b = 7;
a += b++ - b--;

VALORES DE a y b  ¿?
```

```
a = -3; b = 7;
a += ++b - --b;

VALORES DE a y b  ¿?
```

```
a = 8; b = 6;
c = 2 * ++a + 3 * b - 1;
VALOR DE c  ¿?
```

```
a = -3; b = 7;
a += b++ - b-- + ++b;

VALORES DE a y b  ¿?
```

Repasar Actividades Resueltas 1.3 a 1.7

4 Operadores relacionales

Las comparaciones siempre tienen un resultado lógico

(Expresión operador_relacional Expresión) => true o false

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	'beso' > 'alamo'	true
<=	menor o igual que	7.5 <= 7.38	false
>=	mayor o igual que	38 >= 7	true

```
Scanner sc = new Scanner(System.in);
System.out.print("Escriba su edad: ");
int edad = sc.nextInt();
boolean mayorEdad = edad >= 18;
System.out.println("Mayor de edad: " + mayorEdad);
```

Pruebas en: OpRelacionales.java

5 Operadores lógicos

(Expresión operador_logico Expresión) => true o false

&&	Operador and (y)
||	Operador or (o)
!	Operador not (no)

Comprueba si dep tiene un valor entre 1 y 10 (incluidos el 1 y el 10):

```
int dep = 11;
boolean entre1y10 = (dep >= 1) && (dep <= 10);
System.out.println("Departamento "+ dep+
        ", entre 1 y 10?: " + entre1y10);
```

Comprueba si dep tiene un valor que no está entre 1 y 10:

```
int dep = 11;
boolean fuerarango = dep <  1 || dep > 10;
System.out.println("Departamento "+ dep
     +", fuera del rango [1 a 10]?: " + fuerarango);
```

Podemos usar paréntesis en las expresiones booleanas: (dep < 1) || (dep > 10)
EJEMPLOS:

```
boolean valor1 = true && true;   // true
boolean valor2 = true && false;  // false
boolean valor3 = false && false; // false

valor1 = true || true;   // true
valor2 = true || false;  // TRUE
valor3 = false || false; // false

valor1 = !true;   // false
valor2 = !false;  // true
valor3 = !(false || false); //true
```

El operador lógico AND

X && Y

x	y	resultado
true	true	true
true	false	false
false	true	false
false	false	false

El operador lógico OR

X || Y

x	y	resultado
true	true	true
true	false	true
false	true	true
false	false	false

El operador lógico NOT

!X

x	resultado
true	false
false	true

http://www.sc.ehu.es/sbweb/fisica/cursoJava/fundamentos/introduccion/operadores1.htm#Los%20operadores%20l%C3%B3gicos

```
LEER POR TECLADO UN DIA Y UN MES.
    COMPROBAR SI EL DIA ESTA COMPRENDIDO ENTRE 1 Y 31
    COMPROBAR SI EL MES ENTRE 1 Y 12.
    COMPROBAR SI LAS DOS EXPRESIONES SE CUMPLEN O NO
    MOSTRAR EL VALOR true o false SI SE CUMPLEN O NO LAS EXPRESIONES
```

6 Operador ternario

Operador	Expresión en Java
?:	condición ? exp1 : exp2

```
(x > y) ? x : y;
```

Se evalúa la condición de si x es mayor que y, en caso afirmativo se devuelve el valor de la variable x, y en caso contrario se devuelve el valor de y.

```
int a = 10, b = 30;
int mayor = (a > b) ? a : b;
System.out.println(" El mayor de "+ a + ", y " +b
```

```
                                +", es: "+ mayor);
```

Repasar Actividades Resueltas 1.12

Ejemplo en: `opTernario.java` y `opCondicional.java`

7 Operador de concatenación

Los operadores de concatenación en Java se utilizan para unir o concatenar dos o más valores de cadena. El operador de concatenación común en Java es el símbolo "+".

Por ejemplo:

```
        String nombre = "Maria";
        String apellido = "Jesús";
        int edad = 20;

        String nombreCompleto = nombre + " " + apellido + " La edad es: " + edad;
        nombreCompleto = nombreCompleto + ", Codigo Postal: " + 19004;

        System.out.println(nombreCompleto);

    // Usamos el operador + al mostrar datos en pantalla
        System.out.println(nombre + " " + apellido);

        System.out.println(nombre + " " + apellido + " La edad es: " + edad
                    + ", Codigo Postal: " + 19004);
```

El operador "+" no solo se utiliza para concatenar cadenas, sino también para sumar números y para concatenar números y cadenas:

```
int numero = 5;
String cadena = "El valor es: ";
System.out.println(cadena + numero); //imprime "El valor es: 5"
```

8 Precedencia

- Operadores postfijos `a++  a--`
- Operadores unarios `++a  --a  +a  -a  !a`
- Multiplicativos `*  /  %`
- Aditivos `+  -`
- Relacionales `<  >  <=  >=`
- Igualdad / desigualdad `==  !=`

- AND lógico `&&`
- OR lógico `||`
- Ternario `?:`
- Asignación `=  +=  -=  *=  /=  %=  ...`

( ) rompe la prioridad, primero lo que está entre paréntesis

9 Conversión de tipos

- No son posibles todas las conversiones
- Existen dos categorías de conversiones:
  - De ensanchamiento o promoción
    - Por ejemplo: pasar de un valor entero a un real.
  - De estrechamiento o contracción
    - Por ejemplo: pasar de un valor real a un entero.

Tipo de origen	Tipo de destino
byte	short, int, long, float, double
short	int, long, float, double
char	int, long, float, double
int	long, float, double
long	double
float	float, double
double	-

Tipo de origen	Tipo de destino
byte	char
short	byte, char
char	byte, short
int	byte, short, char
long	byte, short, char, int
float	byte, short, char, int, long
double	byte, short, char, int, long, float

```
//conversion de byte (8bits)  a char (16bits) produce estrechamiento
byte b = 65; //1 byte - valor ascii letra A
System.out.println(b);//65

char c='A'; //2 bytes
System.out.println(c); //A

int valorasci = c;
System.out.println("Caracter: " + c + ", valor ascii: "+valorasci);

c=(char) b;  //convertimos byte a char
System.out.println("conversion ok: " + c);

b=-12;
c=(char) b;  //convertimos byte a char
System.out.println("conversion con pérdida: " + c);
```

```
65
A
Caracter: A, valor ascii: 65
conversion ok: A
conversion con perdida: ?
```

De forma general trataremos de atenernos a la norma de que **"en las conversiones debe evitarse la pérdida de información".** En la siguiente tabla vemos conversiones que son seguras por no suponer pérdida de información.

TIPO ORIGEN	TIPO DESTINO
byte	double, float, long, int, char, short
short	double, float, long, int
char	double, float, long, int
int	double, float, long
long	double, float
float	Double

https://conceptodefinicion.de/wp-content/uploads/2014/04/codigo-ascii1-yw.jpg

EJEMPLOS casting:

```
byte a = 20;
int x = (int) a;
```

Al escribir int entre paréntesis se fuerza a cambiar el dato de tipo byte a int.

Hay que tener cuidado al realizar esta conversión ya que se puede aplicar a tipos no compatibles, lo que puede derivar en perdidas de información e incluso errores en ejecución.

```
float x = 5.7F;
int y = (int) x;
System.out.println(y); //pinta 5
```

Se realizará la conversión, pero se perderá la parte decimal del número con punto flotante al guardarlo en y. Solo se guardará 5 en y. Aunque parezca ilógico, habrá ocasiones en las que esta pérdida de precisión nos pueden resultar útil.

```
int z = 100;
float m = (float) z;

System.out.println(m); //pinta 100.0

int a = 3;
double b = a;
b = 5.6789;
a = (int) b;
System.out.println(a);//pinta 5

a = (int) 4.5;
System.out.println(a);//pinta 4
```

HACER LA HOJA DE EJERCICIOS

(1_EJERCICIOS_operadores_expresiones.pdf)

Repasar Actividades Resueltas 1.13 1.14