

# Operadores

Programación I

## Objetivos del tema

- Utilizar operadores aritméticos, incrementales, de relación, lógicos y de asignación; considerando los tipos de dato primitivos sobre los que se aplican
- Evaluar expresiones que empleen datos primitivos, operadores y paréntesis
- Construir expresiones que empleen combinaciones de datos primitivos, operadores y paréntesis

# Operador de asignación

- A una variable se le puede asignar un valor correspondiente con el tipo con el que fue declarada.
- La asignación se realiza usando =
- Una variable puede tomar el valor de otra variable del mismo tipo.

**Por ejemplo:**

```
tipo_de_variable nombre1, nombre2;  
...  
nombre1 = valor_perteneciente_a_tipo_de_variable;  
nombre1 = nombre2;
```

## Ejemplo 1

```
/*El operador de asignación es =
*/
public class Clase_3_Ejemplo_1{
    public static void main(String[] args) {
        int i, j;
        i = 15;
        j = i;
        System.out.println("i = " + i);
        System.out.println("j = " + j);
    }
}
```

# Operadores aritméticos

- Las operaciones aritméticas entre variables/valores que se puede realizar sobre algunos de los tipos primitivos son +, -, \*, /, y % (resto de la división).
- El resultado exacto depende de los tipos primitivos involucrados.

Operador	Descripción	Ejemplo de expresión	Resultado
-	Cambio de signo (unario)	-4	-4
+	Suma	2,5 + 7,1	9,6
-	Resta	235,6 – 103,5	132,1
*	Producto	1,2 * 1,1	1,32
/	División (reales o enteros)	0,05 / 0,2 7 / 2	0,25 3
%	Resto (solo para enteros)	20 % 7	6

## Ejemplo 2

```
/* operadores sobre el tipo entero y double
*/
public class Clase_3_Ejemplo_2{
    public static void main(String[] args) {
        int i, j, res1, res2, res3;
        double a, b, res4, res5;
        i = 7;
        j = 3;
        a = 7.0;
        b = 3.0;
        res1 = i-j;
        res2 = i/j;
        res3 = i%j;
        res4 = a*b;
        res5 = a/4.0;
        System.out.println("Operador resta: i - j = " + res1);
        System.out.println("Operador division: i / j = " + res2);
        System.out.println("Operador resto: i % j = " + res3);
        System.out.println("Operador multiplicacion: a * b = " + res4);
        System.out.println("Operador division: a / 4.0 = " + res5);
    }
}
```

## Separadores

- Los separadores son fundamentales para cumplir con la sintaxis del lenguaje de programación y definir prioridades en la ejecución de operadores.

Separador	Descripción
()	Permiten modificar la prioridad de operadores en una expresión.
{}	Permiten definir bloques de código: del programa, de sentencias como try...catch, etc.
:	Permite separar sentencias.
,	Permite separar identificadores en una declaración de una sola línea.

## Ejemplo 2

```
/* operadores sobre el tipo entero y double
*/
public class Clase_3_Ejemplo_2{
    public static void main(String[] args) {
        int i, j;
        double a, b;
        i = 7;
        j = 3;
        a = 7.0;
        b = 3.0;
        //en todos los casos que siguen primero se resuelve la operación entre ()
        //y luego al resultado convertido en texto se lo concatena
        System.out.println("Operador resto: i - j = " + (i-j));
        System.out.println("Operador division: i / j = " + (i/j));
        System.out.println("Operador resto: i % j = " + (i%j));
        System.out.println("Operador multiplicacion: a * b = " + (a*b));
        System.out.println("Operador division: a / 4.0 = " + (a/4.0));
    }
}
```

## Operadores aritméticos incrementales

- Los operadores incrementales son funcionalidades específicas de algunos de los lenguajes de programación, que permiten realizar incrementos o decrementos de valores de variables.
- Solo aplicaremos dichas operaciones al tipo entero.
- Es posible reemplazar estas operaciones incorporando pasos/sentencias previos.

# Operadores aritméticos incrementales

Operador	Descripción	Ejemplo de expresión	Resultado del Ejemplo
++	<ul style="list-style-type: none"><li>• <code>a++</code>. Incrementa el valor de <code>a</code> en uno.</li><li>• <code>++a</code>. Incrementa el valor de <code>a</code> en uno.</li><li>• <code>a++</code>. Primero se usa el valor de la variable y luego se incrementa <code>a</code>.</li><li>• <code>++a</code>. Primero se incrementa el valor de <code>a</code> y después se utiliza.</li></ul>	<code>a = 5;</code> <code>a++;</code>  <code>a = 5;</code> <code>++a;</code>  <code>a = 5;</code> <code>b = a++;</code>  <code>a = 5;</code> <code>b = ++a;</code>	<code>a vale 6</code>  <code>a vale 6</code>  <code>a vale 6 b vale 5</code>  <code>a vale 6 b vale 6</code>
--	Decremento. Funciona de manera análoga al incremento pero descontando uno al valor.	<code>a--;</code> <code>--a;</code> <code>b = a--;</code> <code>b = --a;</code>	

## Ejemplo 3

```
/* operadores incrementales
*/
public class Clase_3_Ejemplo_3{
    public static void main(String[] args) {
        int a, b;
        a = 5;
        a++;
        System.out.println("a = " + a);
        a = 5;
        ++a;
        System.out.println("a = " + a);
        a = 5;
        b = a++;
        System.out.println("a = " + a + ", b = " + b);
        a = 5;
        b = ++a;
        System.out.println("a = " + a + ", b = " + b);
    }
}
```

## Operadores aritméticos combinados

- Los operadores combinados son funcionalidades específicas de algunos de los lenguajes de programación, que permiten realizar operaciones preestablecidas con una reducción de la sintaxis.
- Las operaciones preestablecidas se corresponden con los tipos donde se pueden aplicar (enteros y reales).

## Operadores aritméticos combinados

Operador	Descripción	Ejemplo de expresión	Resultado
<code>+=</code>	Suma	<code>a += b</code>	<code>a = a + b</code>
<code>-=</code>	Resta	<code>a -= b</code>	<code>a = a - b</code>
<code>*=</code>	Multiplicación	<code>a *= b</code>	<code>a = a * b</code>
<code>/=</code>	División	<code>a /= b</code>	<code>a = a / b</code>
<code>%=</code>	Resto	<code>a %= b</code>	<code>a = a % b</code>

## Ejemplo 4

```
/* operadores combinados sobre el tipo entero y double
*/
public class Clase_3_Ejemplo_4{
    public static void main(String[] args) {
        int i, j;
        double a, b;
        i = 7;
        j = 3;
        a = 7.0;
        b = 3.0;
        i += j;
        a /= b;
        System.out.println("Operador combinado suma: i+=j " + i);
        System.out.println("Operador combinado division: a/=b " + a);
    }
}
```

## Operadores de relación

- Los operadores de relación permiten comparar variables y obtener resultados booleanos de dicha comparación.
- Así como hay un orden entre los números, hay un orden entre los caracteres. En este caso solo se utilizarán sobre caracteres dígitos y caracteres letras:
  - '0'<'1'<'2'...'9'
  - 'a'<'b'<'c'...'z'

# Operadores de relación

Operador	Descripción	Ejemplo de expresión	Resultado
<code>==</code>	Igual a	<code>7 == 38</code>	false
<code>!=</code>	Distinto de	<code>'a' != 'k'</code>	true
<code>&lt;</code>	Menor que	<code>'G' &lt; 'B'</code>	false
<code>&gt;</code>	Mayor que	<code>'b' &gt; 'a'</code>	true
<code>&lt;=</code>	Menor o igual que	<code>'0' &lt;= '2'</code>	true
<code>&gt;=</code>	Mayor o igual que	<code>38 &gt;= 7</code>	true

## Ejemplo 5

```
/* operadores de relacion
*/
public class Clase_3_Ejemplo_5{
    public static void main(String[] args) {
        int i, j;
        char c1, c2;
        boolean resultado;
        i = 7;
        j = 3;
        resultado = (i!=j);
        c1 = '0';
        c2 = '5';
        System.out.println("Operador relacion: i!=j " + resultado);
        System.out.println("Operador realcion: c1>c2 " + (c1>c2));
    }
}
```

# Operadores lógicos

- Los operadores lógicos permiten comparar variables/expresiones booleanas y obtener resultados booleanos de dicha comparación (en función de la tabla de valores de verdades).

Operador	Descripción	Ejemplo de expresión	Resultado
!	Negación – Not	! false	true
	Suma lógica – Or	true    false	true
&&	Multiplicación lógica – And	false && true	false

## Ejemplo 6

```
/* Separadores
*/
public class Clase_3_Ejemplo_6{
    public static void main(String[] args) {
        int i, j, k;
        boolean res;
        i = 7;
        j = 3;
        k = 2;
        //((i==j)|| (i==k)) primero se resuelven los paréntesis (i==j) y
        // (i==k), y luego ||. Finalmente el resultado queda en res
        res = (i==j)|| (i==k);
        System.out.println("Operacion: (i==j)|| (i==k) " + res);
        //((i!=j)|| (i==k)) primero se resuelven los paréntesis (i!=j) y
        // (i==k), y luego ||
        System.out.println("Operacion: ((i!=j)|| (i==k)) " + ((i!=j)|| (i==k)));
        //((!(i!=j))|| (i==k)) primero se resuelven los paréntesis (!(i!=j))
        // (i==k), y luego ||. Para resolver (!(i!=j)) primero se resuelve
        // (i!=j) y luego se le aplica ! (negacion)
        System.out.println("Operacion: ((!(i!=j))|| (i==k)) " + ((!(i!=j))|| (i==k)));
    }
}
```

## Ejemplo 7

```
/* Escribir un programa que permita el ingreso de dos números enteros por teclado e imprima el resultado de comparar:  
* _ el primero es multiplo de 3. _ el primero es multiplo de 5. _ el primero es multiplo del segundo. _ el primero es multiplo de 3 y de 5.  
*/  
  
import java.io.BufferedReader;  
import java.io.InputStreamReader;  
  
public class Clase_3_Ejemplo_7 {  
  
    public static void main(String[] args) {  
  
        int numero1, numero2;  
  
        boolean resultado;  
  
        try{  
  
            BufferedReader entrada = new BufferedReader(new InputStreamReader(System.in));  
  
            System.out.println("Ingrese un numero entero :");  
  
            numero1 = Integer.valueOf(entrada.readLine());  
  
            System.out.println("Ingrese un numero entero :");  
  
            numero2 = Integer.valueOf(entrada.readLine());  
  
            resultado = ((numero1 % 3)==0);  
  
            System.out.println("el primero es multiplo de 3 :" + resultado);  
  
            resultado = ((numero1 % 5)==0);  
  
            System.out.println("el primero es multiplo de 5 :" + resultado);  
  
            resultado = ((numero1 % numero2)==0);  
  
            System.out.println("el primero es multiplo del segundo :" + resultado);  
  
            resultado = (((numero1 % 3)==0)&&((numero1 % 5)==0));  
  
            System.out.println("el primero es multiplo de 3 y 5 :" + resultado);  
  
        }  
  
        catch (Exception exc){  
  
            System.out.println(exc);  
  
        }  
  
    }  
}
```

### Video de explicación

<https://drive.google.com/file/d/17qxrmz95ZlCZg7f6Leev4lw00FjSLASz/view?usp=sharing>

## Práctico

1. Escribir un programa que permita el ingreso de un número entero por teclado e imprima el cociente de la división de dicho número con 2, 3, y 4.
2. Escribir un programa que imprima por pantalla la tabla de valores de verdad para el or y and por separado.
3. Escribir un programa que permita el ingreso de dos números enteros por teclado e imprima el resultado de comparar:
  - \_ el primero mayor al segundo.
  - \_ ambos son múltiplos de 2.
4. Escribir un programa que ingrese un número entero por teclado e imprima el resultado de determinar:
  - \_ es múltiplo de 6 y de 7,
  - \_ es mayor a 30 y múltiplo de 2, o es menor igual a 30,
  - \_ el cociente de la división de dicho número con 5 es mayor a 10.