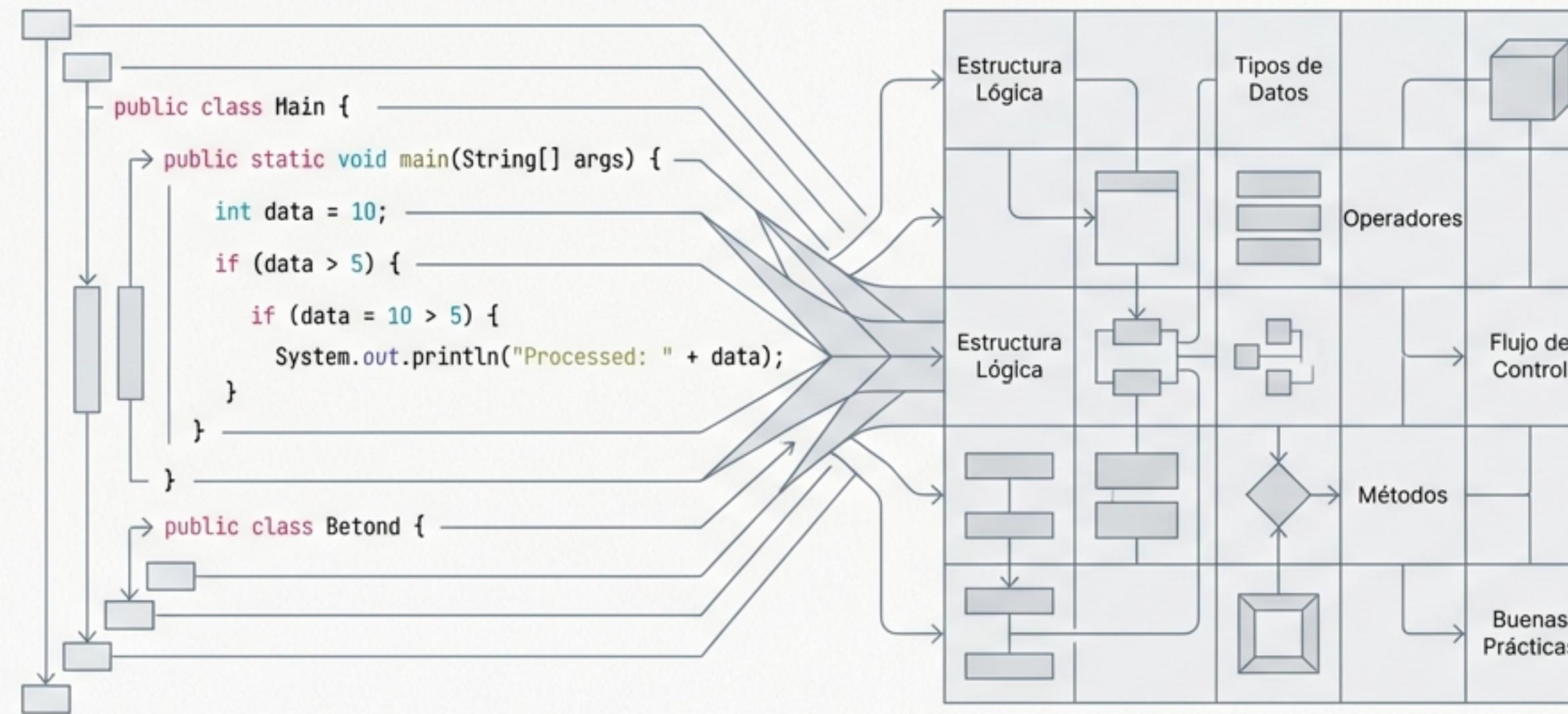


Elementos de un Programa Informático: De la Sintaxis a la Lógica

Estructura, Datos, Operadores y Buenas Prácticas en Java

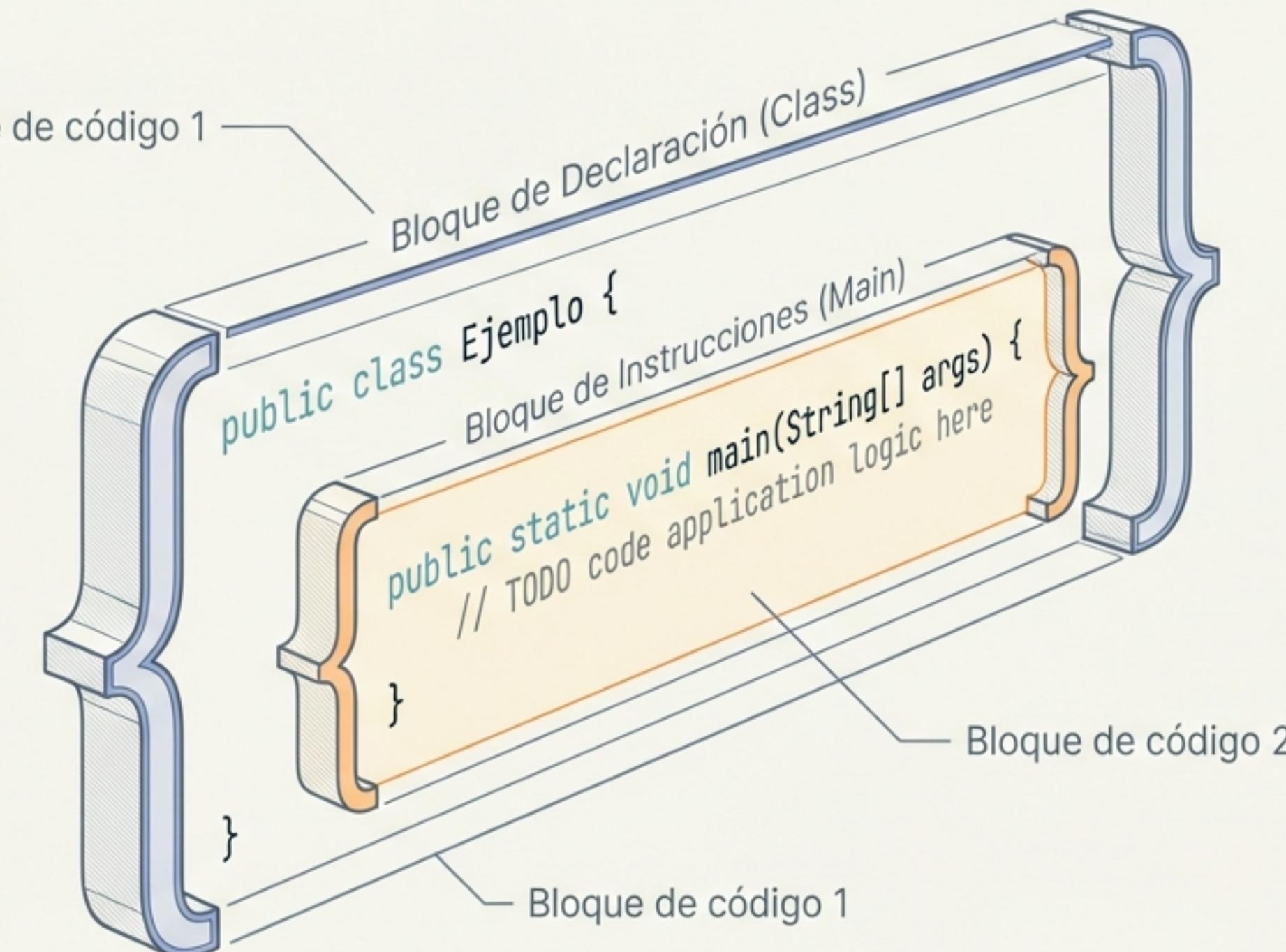


La Arquitectura del Código: Bloques y Estructura

Bloques: Secciones de código delimitadas por llaves '{ }' que definen el alcance y la estructura.

Bloque de Declaración: Donde se definen los objetos y variables (ingredientes) antes de usarlos.

Bloque de Instrucciones: Conjunto de operaciones y lógica (la receta) que procesa los datos de entrada.



Nota: Todo programa sigue reglas sintácticas y semánticas definidas por el lenguaje (Java).

Identificadores: Las Reglas del Nombrado

Reglas de Identificadores

- Nombres para componentes (variables, constantes, métodos).
- No pueden repetirse.
- Deben empezar por una letra.
- Sin espacios en blanco.
- Deben ser significativos.

Tipos de Nombres

A. Palabras Reservadas

abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

JetBrains Mono Inter Regular

Términos predefinidos con significado especial.
No usables como nombres propios.

B. Símbolos del Programador

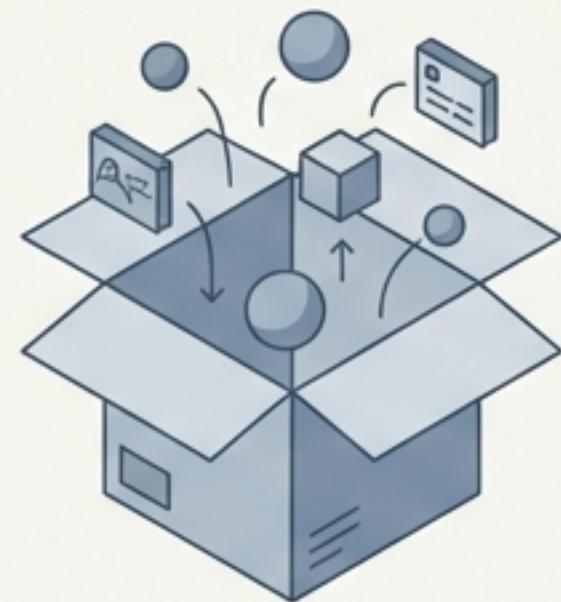


Inter Regular

Los nombres que tú creas
(Variables y Constantes).

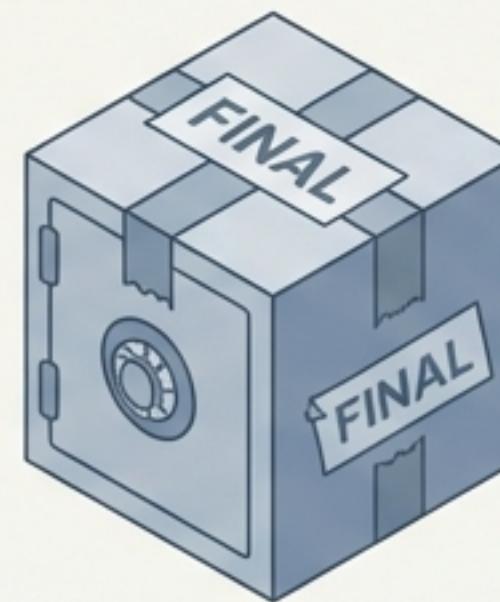
El Contenedor: Variables vs. Constantes

Variable



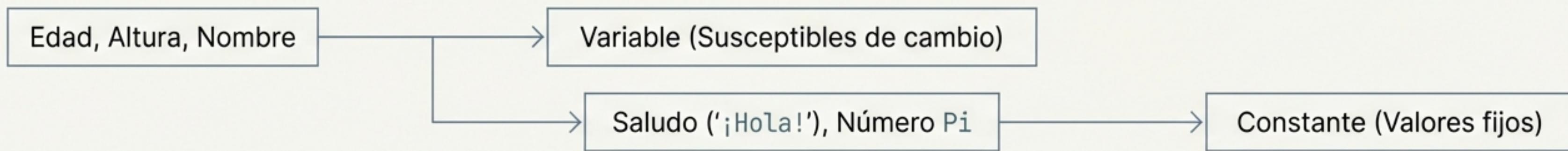
Espacio en memoria cuyo valor cambia durante la ejecución.
Requiere: **Tipo + Nombre + Valor Inicial**.

Constante

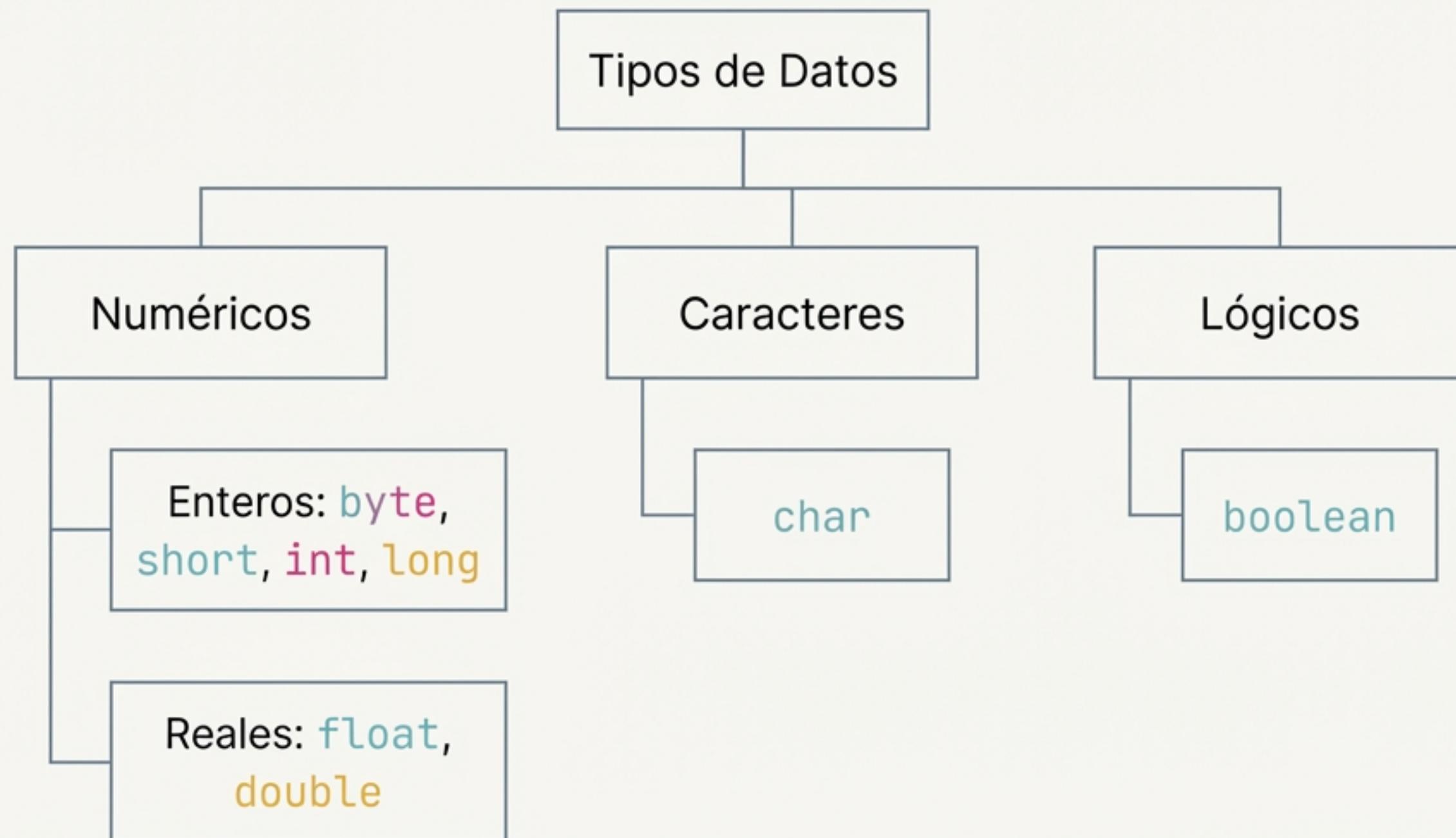


Dato fijo e inmutable. Se declara con la palabra reservada **final**.

Caso Práctico: Clasificación



Taxonomía de Datos: Tipos Primitivos



Insight: El tipo de dato determina el dominio de valores y el espacio ocupado en memoria (**bytes**).

El Universo Numérico: Enteros y Reales

Enteros (Sin decimales)

- `byte` (1 byte)
- `short` (2 bytes)
- `int` (4 bytes) -> *Estándar por comodidad*
- `long` (8 bytes)

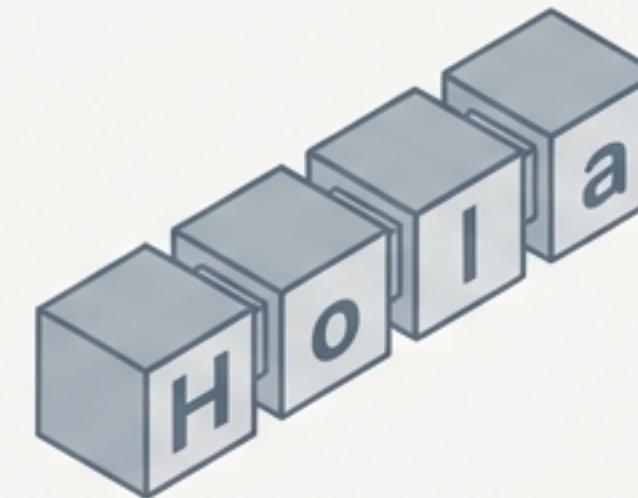
Reales (Con decimales)

- `float` (4 bytes)
- `double` (8 bytes) -> *Estándar por precisión*

Tipo	Bytes	Rango General
<code>int</code>	4	-2^{31} a $2^{31}-1$
<code>double</code>	8	$\pm 4.9E-324$ a $\pm 1.8E308$

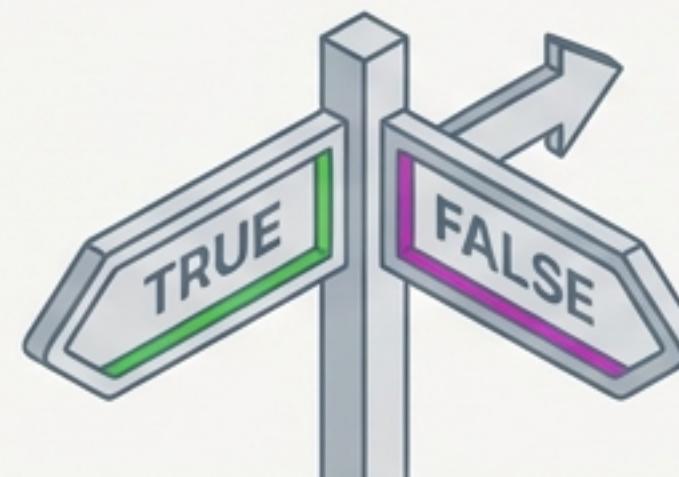
Texto y Verdad: Alfanuméricos y Booleanos

Alfanuméricos



- **Char:** Un único carácter. Comillas simples ('a').
 - **String:** Cadena de caracteres. Comillas dobles (\\"Hola\\").
- Puede ser vacía "".

Lógicos (Boolean)



- **Valores:** Únicamente `true` (verdadero) o `false` (falso).
- **Uso:** No sirven para cálculos matemáticos. Se usan para tomar decisiones.

```
boolean booleano = false;
```

El Motor de Procesamiento: Expresiones y Operadores

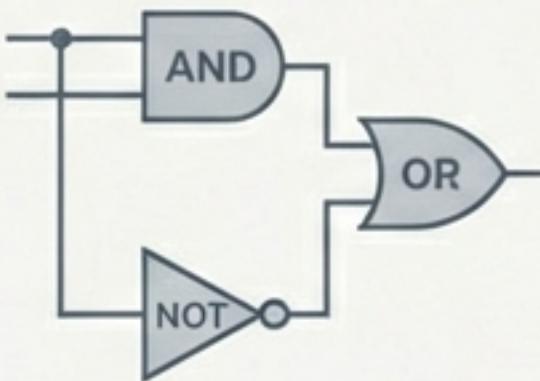
Una **expresión** combina constantes, variables y operadores para obtener un nuevo valor.

Aritméticos



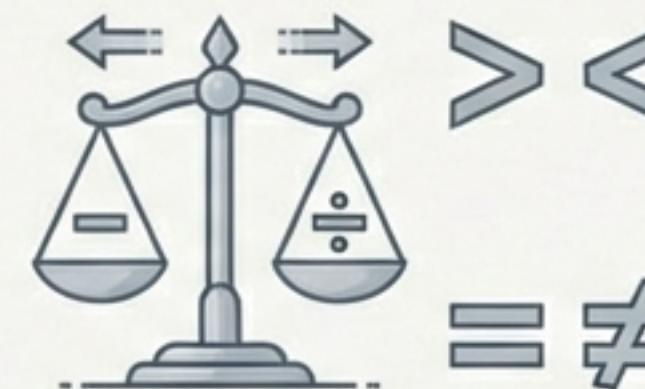
+ , - , ·
*
/
%

Lógicos



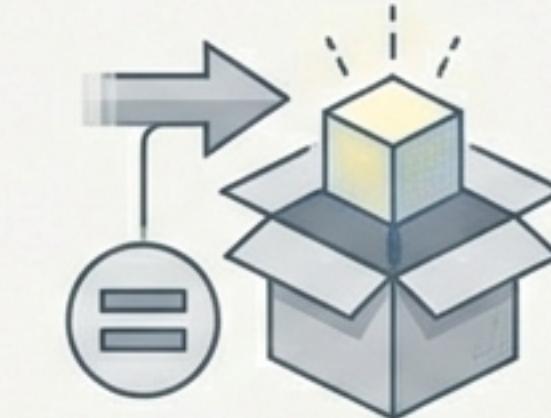
AND (&&)
OR (||)
NOT (!)

Relacionales



>
<
==
!=

Asignación



=
+=
-=

Matemáticas y Comparaciones

Aritmética y el Módulo



Resto (%)

- `%` (Resto): Obtiene el residuo de una división.
- `++` / `--`: Incremento y decremento en 1.

Operadores Relacionales

Devuelven siempre un resultado *booleano* (true / false).

- `==` (Igual que)
- `!=` (Distinto)
- `>` (Mayor que), `<` (Menor que)
- `>=`, `<=`

⚠️ No confundir `==` (comparación) con `=` (asignación).

El Cerebro Lógico: Tablas de Verdad

Evaluación de condiciones complejas.

AND (&&)

T	F	
T && T	= True	
T && F	= False	
F && F	= False	

Verdadero solo si **ambas** son verdaderas.

OR (||)

T	F	
T T	= True	
T F	= True	
F F	= False	

Verdadero si **al menos una** es verdadera.

NOT (!)

!	
! True	= False
! False	= True

Invierte el valor.

```
boolean resultado = booleano1 && booleano2;
```

Caso Práctico: El Misterio de la Precedencia

¿Cuál es la diferencia entre `++x` y `x++`?

The Rule

- **Pre-incremento (`++x`):** Incrementa, luego usa el valor.
- **Post-incremento (`x++`):** Usa el valor actual, luego incrementa.

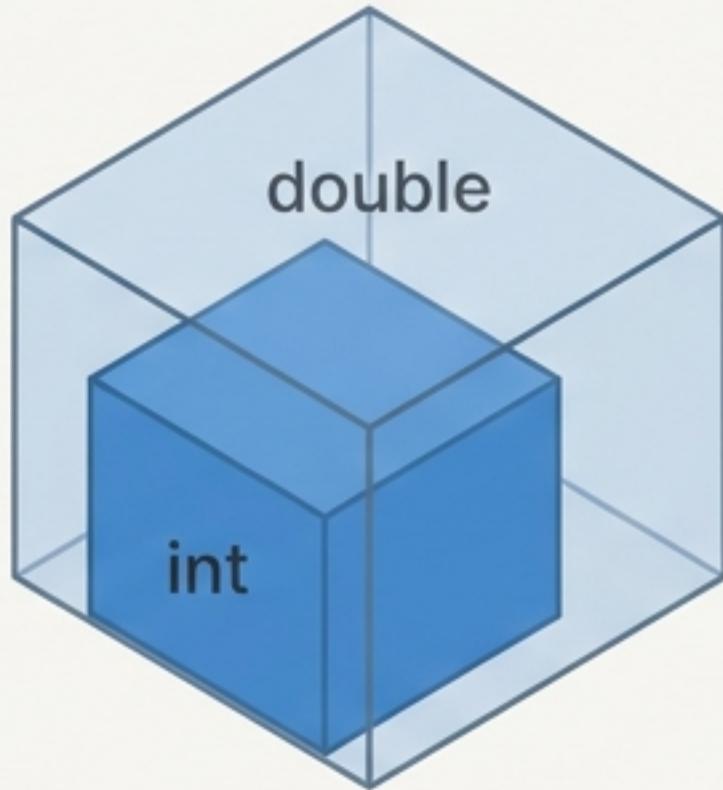
Traza de Ejecución (Code Walkthrough)

Instrucción	Valor de X	Salida en Pantalla
<code>int x = 10;</code>	10	-
<code>System.out.println(x++);</code>	11	10 (Imprime el actual, luego sube)
<code>System.out.println(++x);</code>	12	12 (Sube primero, luego imprime)

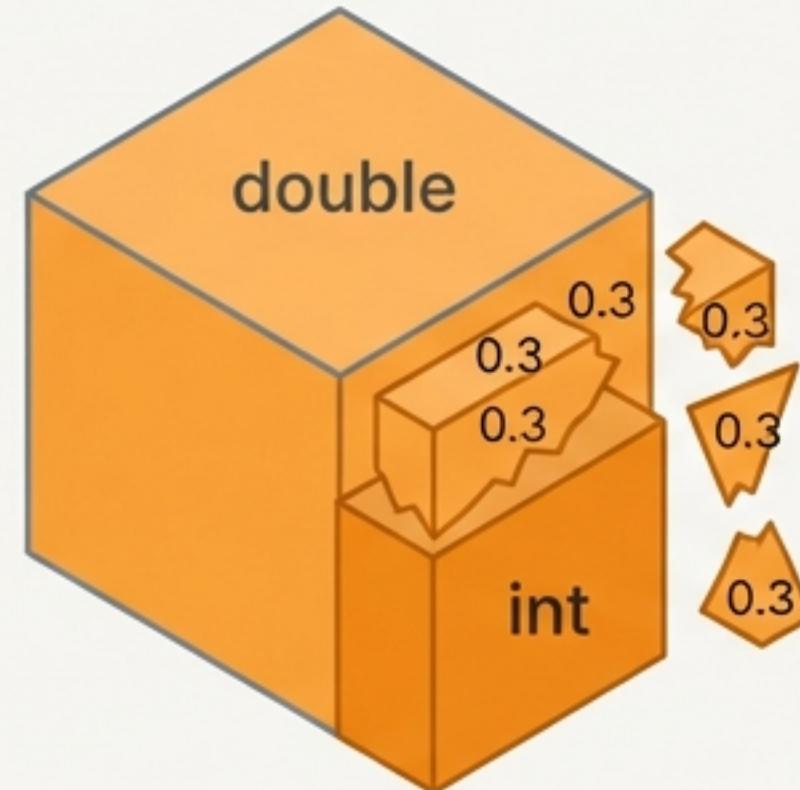
Metamorfosis de Datos: Casting (Conversiones)

Conversiones de tipos en programación.

Upper-Casting (Implícito)



Down-Casting (Explícito)



- De tipo menor a mayor (`int` → `double`).
- Automático. ***Sin pérdida de información***.

- De tipo mayor a menor (`double` → `int`).
- Manual: requiere `(int)` variable`.
- ***Pérdida de precisión*** (se truncan decimales).

```
double real = 8.3;  
int entero = (int) real; // entero vale 8
```

La Interfaz: API de Java y Entrada/Salida

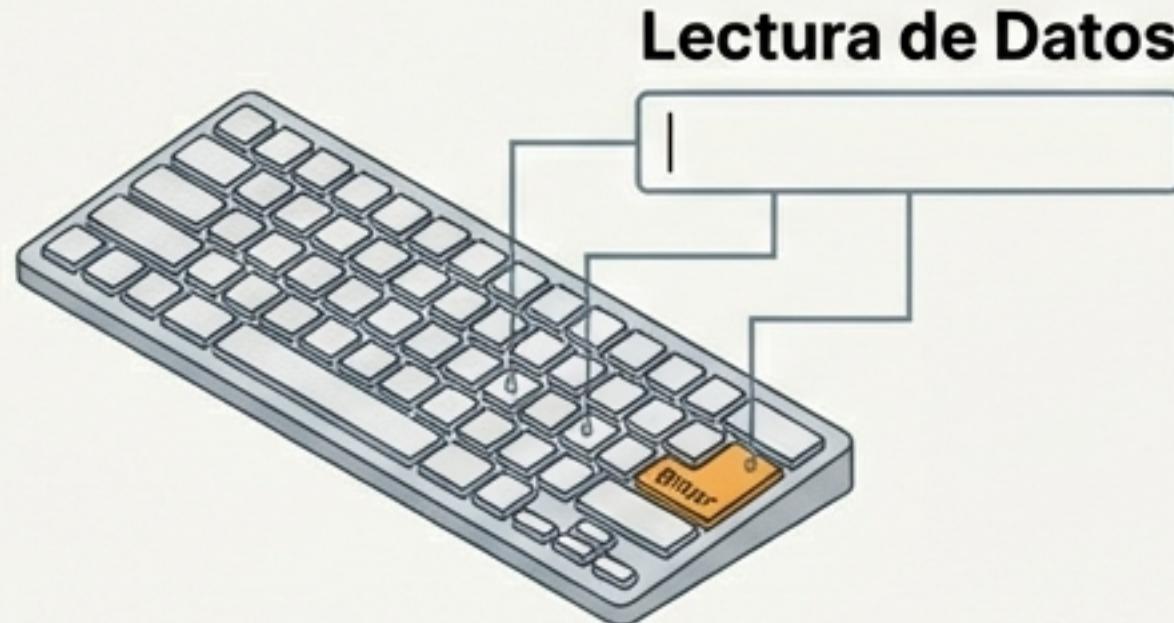
Herramientas predefinidas para comunicar el programa.

Salida (Output) - System.out



- `System.out.print()`: Imprime sin salto de línea.
- `System.out.println()`: Imprime con salto de línea.

Entrada (Input) - Clase Scanner



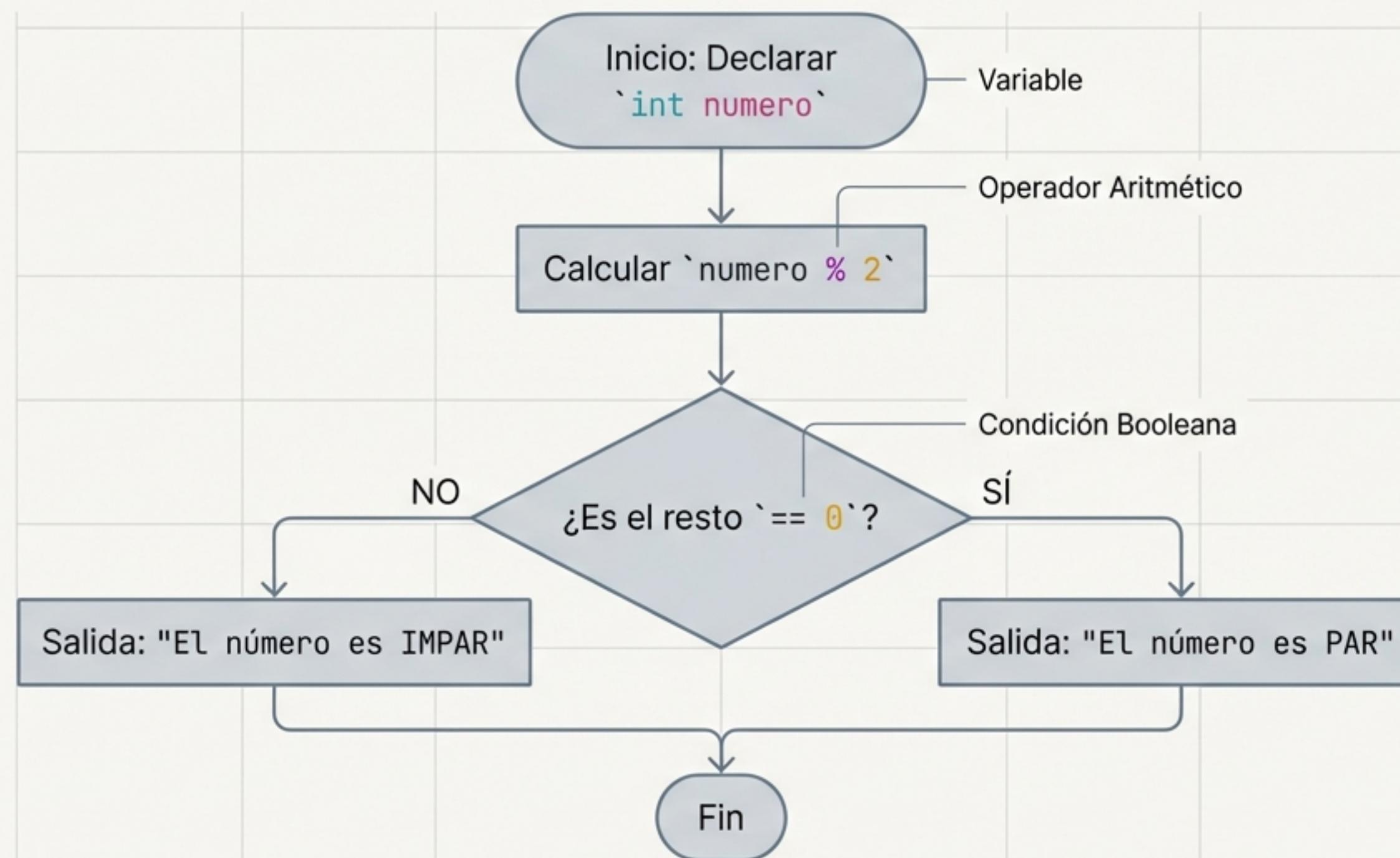
- Inicialización:
`Scanner lectura = new Scanner(System.in);`
- Métodos:
 - `.nextInt()` (Entero)
 - `.nextDouble()` (Real)
 - `.nextLine()` (Texto)

Documentación y Legibilidad

```
// Comprobar si el número es par
if (numero % 2 == 0) {
    /* Si el resto es 0, significa
     que es divisible por 2 */
    System.out.println("Es Par");
}
```

Propósito: Los comentarios explican el *qué* y el *por qué* para tu 'yo' del futuro y otros desarrolladores. Un código limpio es vital.

Resumen Maestro: El Algoritmo ¿Par o Impar?



Este algoritmo integra almacenamiento, cálculo y toma de decisiones.