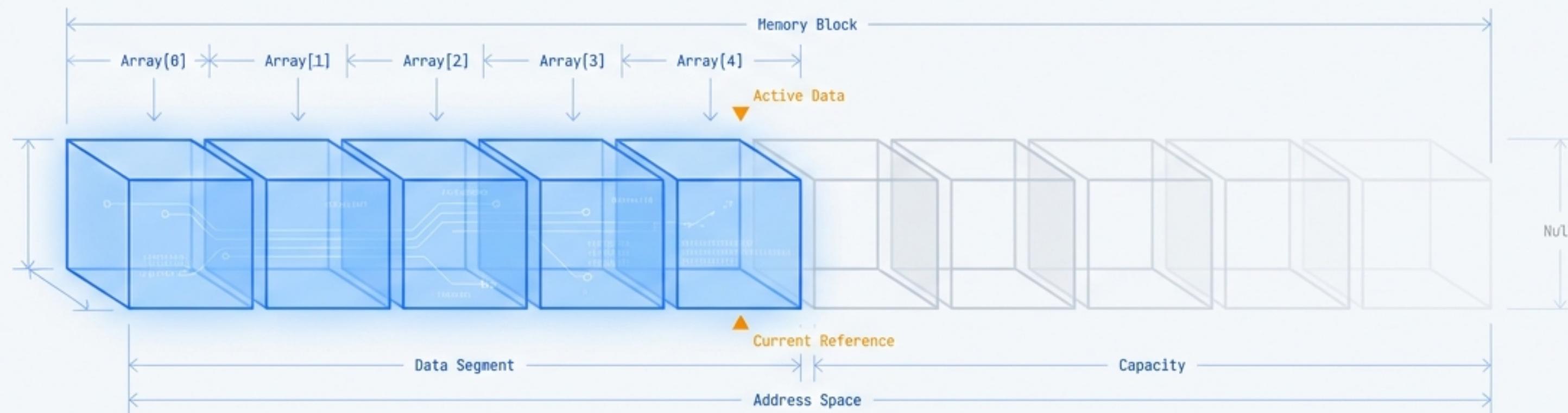


# Programación en Java: Estructuras de Almacenamiento y Cadenas

De la gestión de memoria a la manipulación de texto



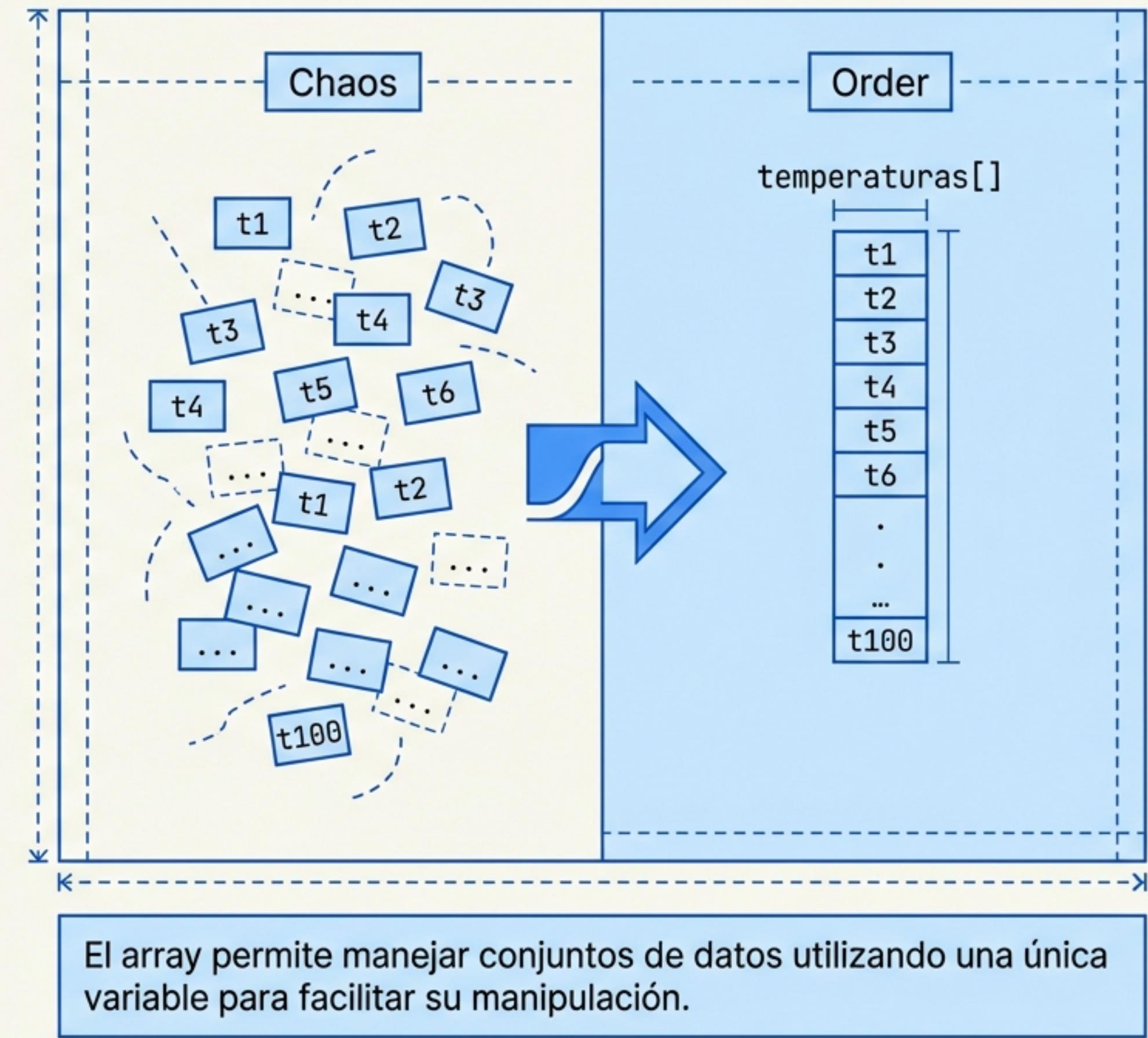
# El Problema de la Escala: Datos Compuestos

## La Limitación

Los tipos primitivos definen un dominio de valores único. ¿Cómo gestionamos 100 temperaturas o 50 nombres? Declarar 100 variables independientes es insostenible.

## La Solución

Los **Datos Compuestos** (como Arrays y Matrices) actúan como contenedores. Nos permiten agrupar valores del mismo tipo bajo un único identificador.



# Anatomía de un Array



## Homogeneidad

Todos los elementos deben ser del mismo tipo de dato.

## Índices

La posición se gestiona numéricamente (0 a n-1).

## Longitud Fija

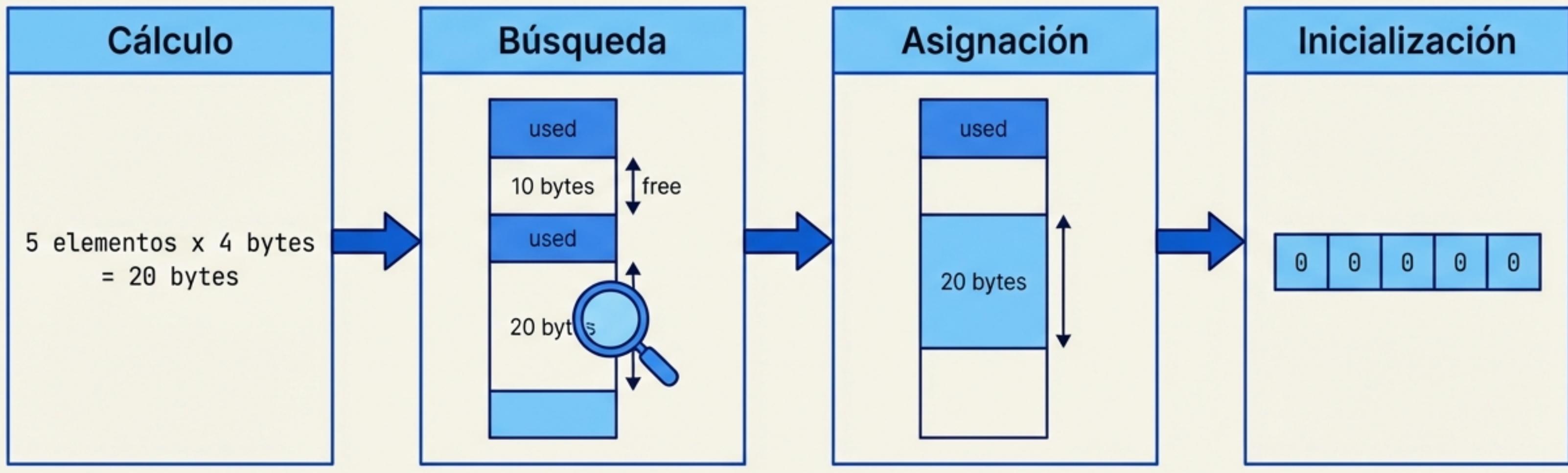
Se define al nacer y no cambia.



**Error Crítico:** Intentar acceder a un índice fuera del rango (ej. índice 10) provocará una excepción en tiempo de ejecución.

# Bajo el Capó: La Reserva de Memoria y el Operador `new`

Visualizando `int datos[] = new int[5];`



El sistema calcula el espacio total requerido.

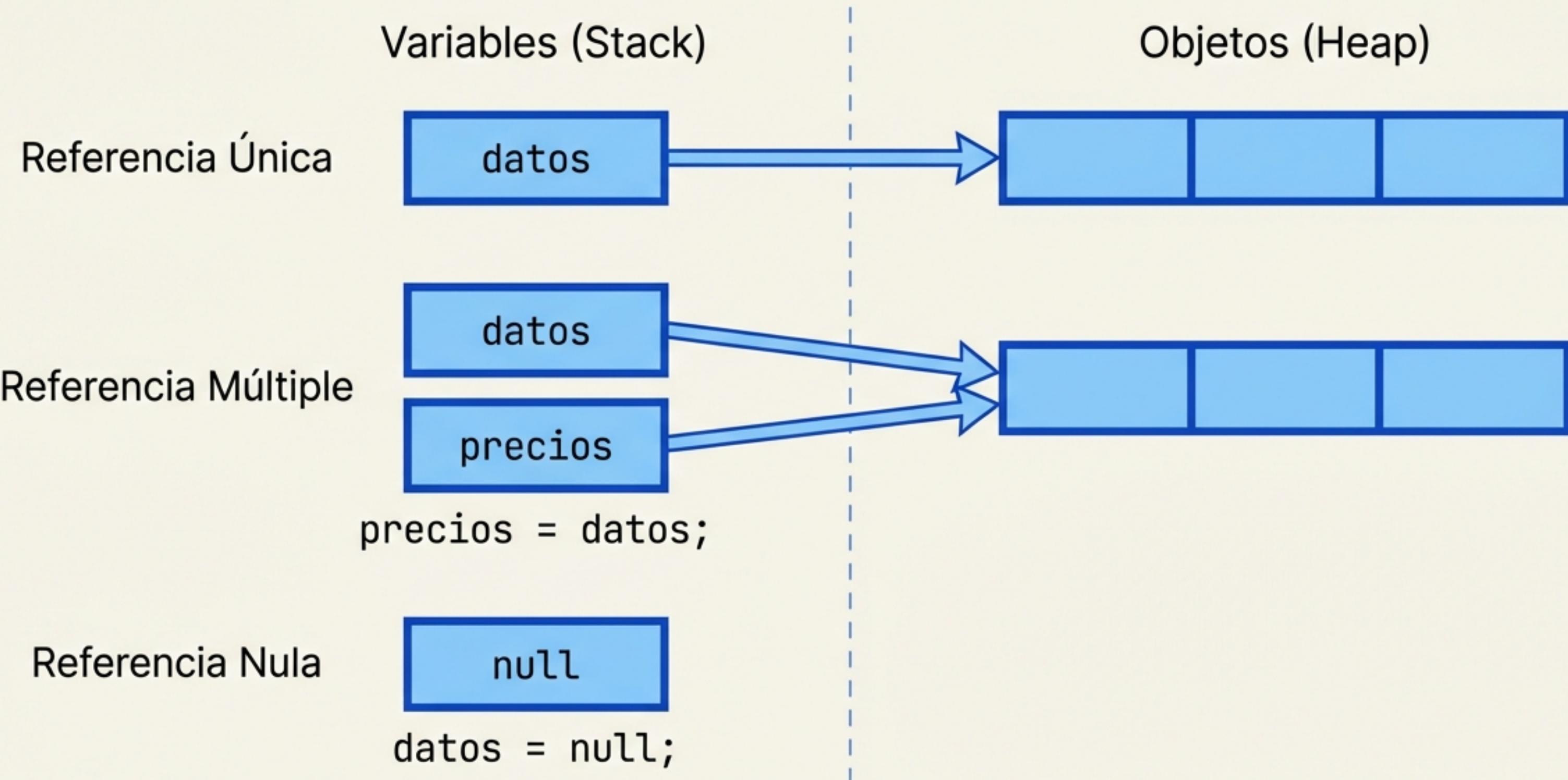
Búsqueda de espacio contiguo libre en RAM.

Reserva del espacio físico.

Valores por defecto asignados (0).

El operador `new` no es mágico; es una solicitud de espacio físico contiguo en la memoria del ordenador.

# El Juego de las Referencias



La variable no *contiene* el array; contiene la dirección de memoria donde empieza el array.

# Gestión Automática: El Recolector de Basura (Garbage Collector)

## 1. El Escenario:

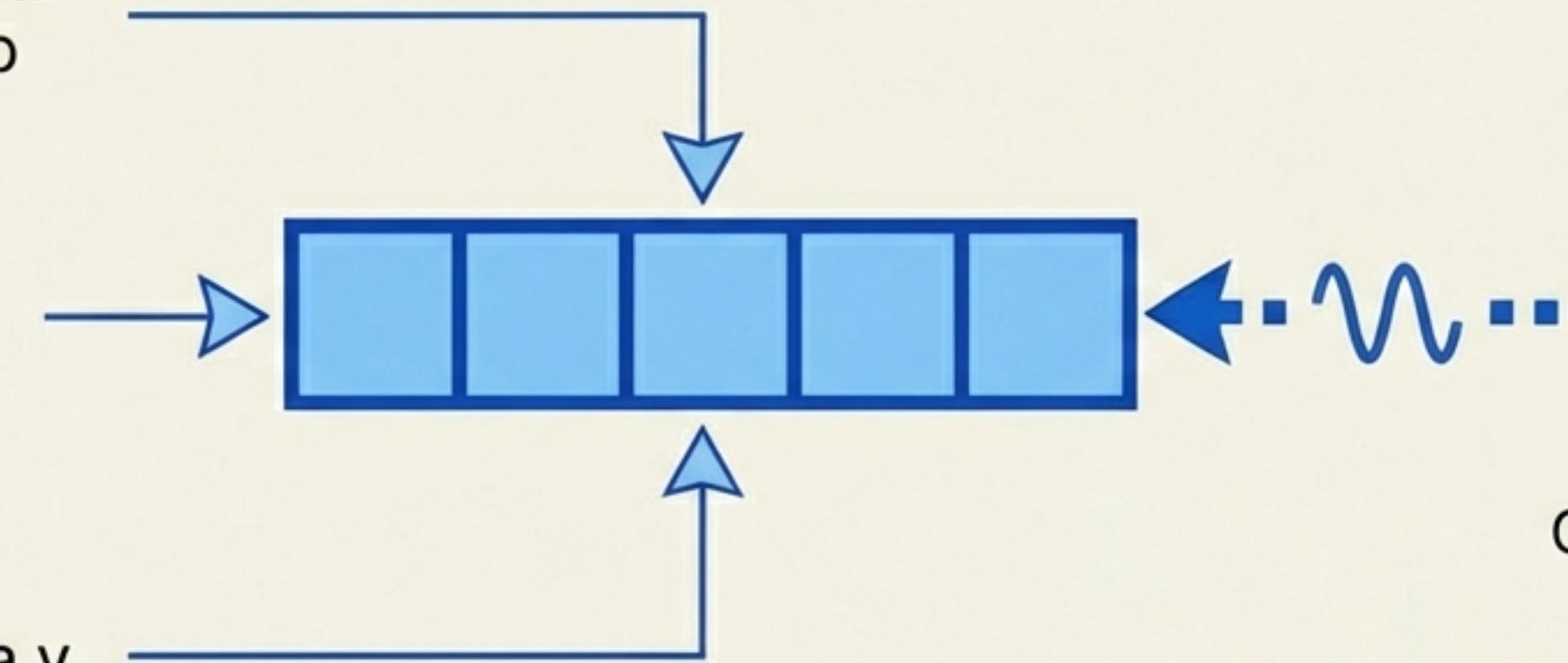
"Se asigna `null` a la referencia. El acceso se rompe."

## 2. La Acción:

"Java detecta el objeto huérfano."

## 3. El Resultado:

"La memoria es liberada y devuelta al sistema."



Garbage Collector  
'JetBrains Mono'

**"Si no hay referencia, no hay acceso. Si no hay acceso, se limpia."**

# Sintaxis: Declaración e Inicialización

## Declaración Estándar

```
● ○ ●  
int[] edades;  
// Solo declara la variable  
edades = new int[9];  
// Reserva memoria e inicializa
```

## Inicialización Directa

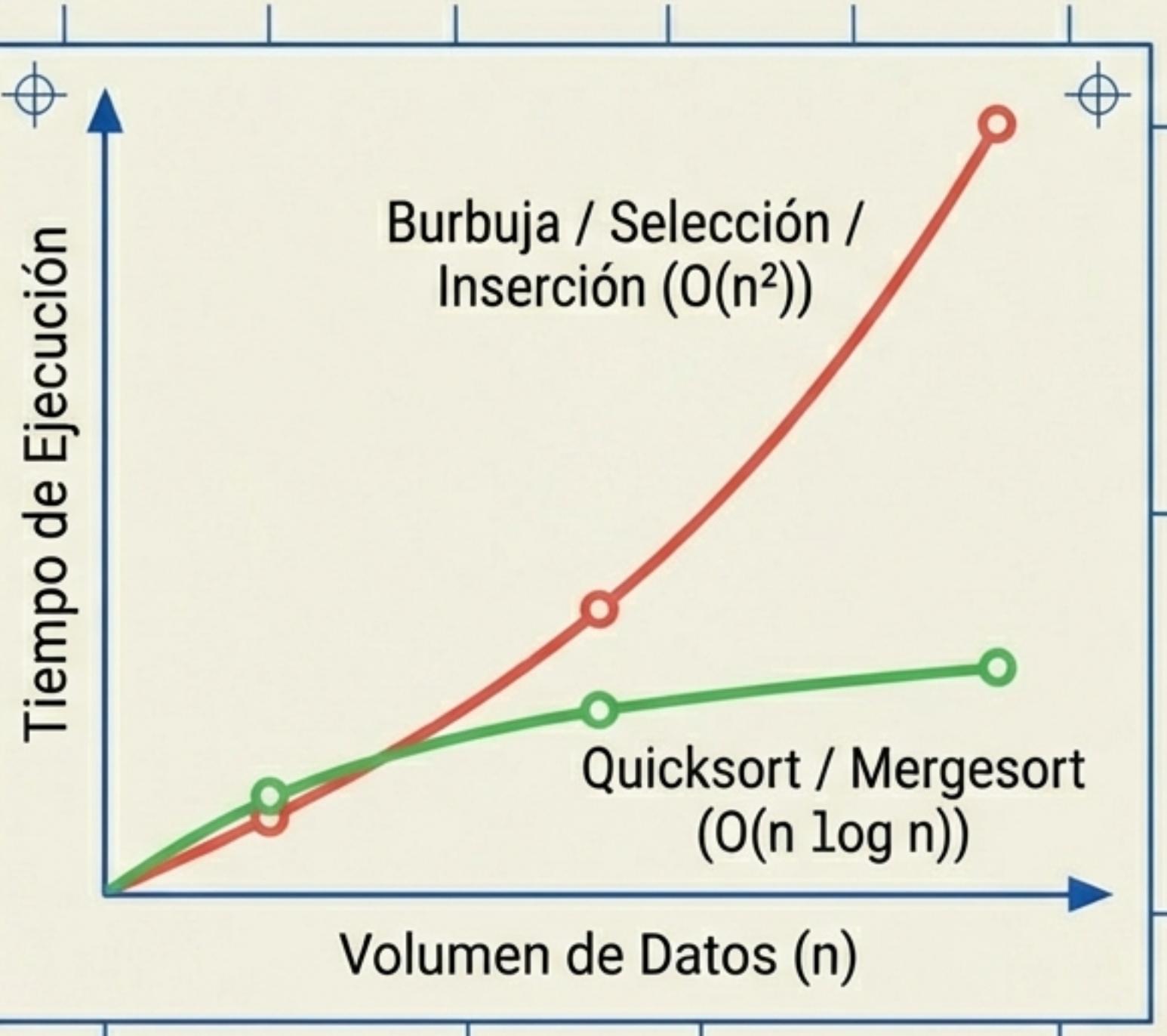
```
● ○ ●  
int[] edades = {11, 12, 13, 14, 15};  
// Crea y rellena en un paso
```

## Por Iteración

```
● ○ ●  
for (int i=0; i < edades.length; i++) {  
    edades[i] = i * 10;  
}  
// Relleno dinámico
```

Un array numérico nace lleno de ceros (0); un array booleano nace lleno de `false`.

# Poniendo Orden al Caos: Algoritmos de Ordenación



## Métodos Básicos (Fuerza Bruta)

Burbuja, Selección, Inserción

**Eficiencia  $O(n^2)$ .** Lentos con grandes volúmenes. 100 elementos = 10,000 operaciones.

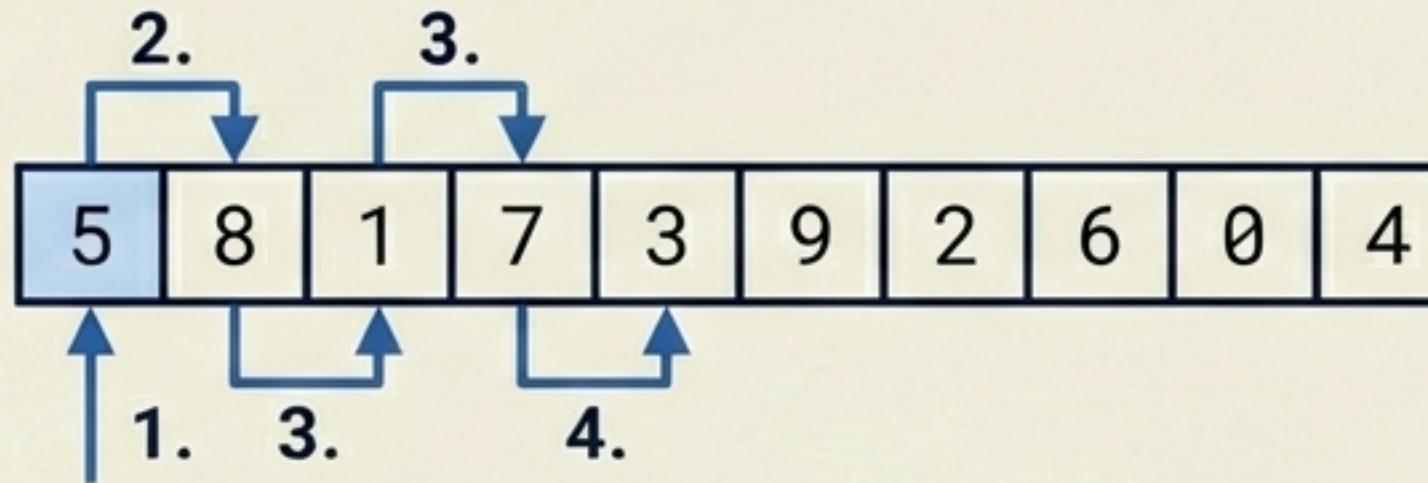
## Métodos Avanzados (Divide y Vencerás)

Quicksort, Mergesort

**Eficiencia  $O(n \log n)$ .** Estándar en la industria. Rápidos y escalables.

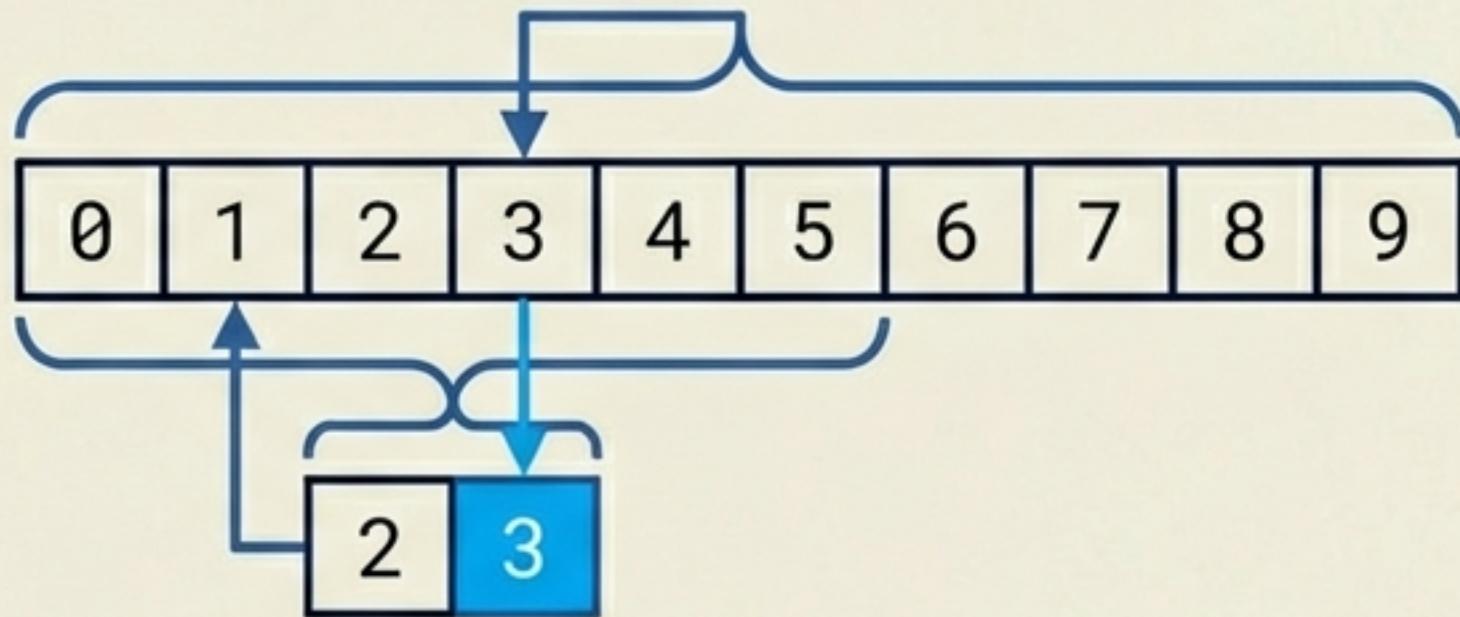
# Estrategias de Búsqueda: ¿Fuerza Bruta o Lógica?

## Búsqueda Lineal



- **Cómo funciona:** Recorre elemento por elemento.
- **Coste:** Alto ( $O(n)$ ).
- **Requisito:** Ninguno (funciona desordenado).

## Búsqueda Binaria



- **Cómo funciona:** Divide el rango a la mitad repetidamente.
- **Coste:** Muy Bajo ( $O(\log n)$ ).
- **Requisito:** El array debe estar ORDENADO.

Ordenamos primero para buscar más rápido después.

# Dimensiones Extra: Tablas Bidimensionales (Matrices)

	Col 0	Col 1	Col 2	Col 3
Fila 0				
Fila 1				
Fila 2				

matriz[1][2]

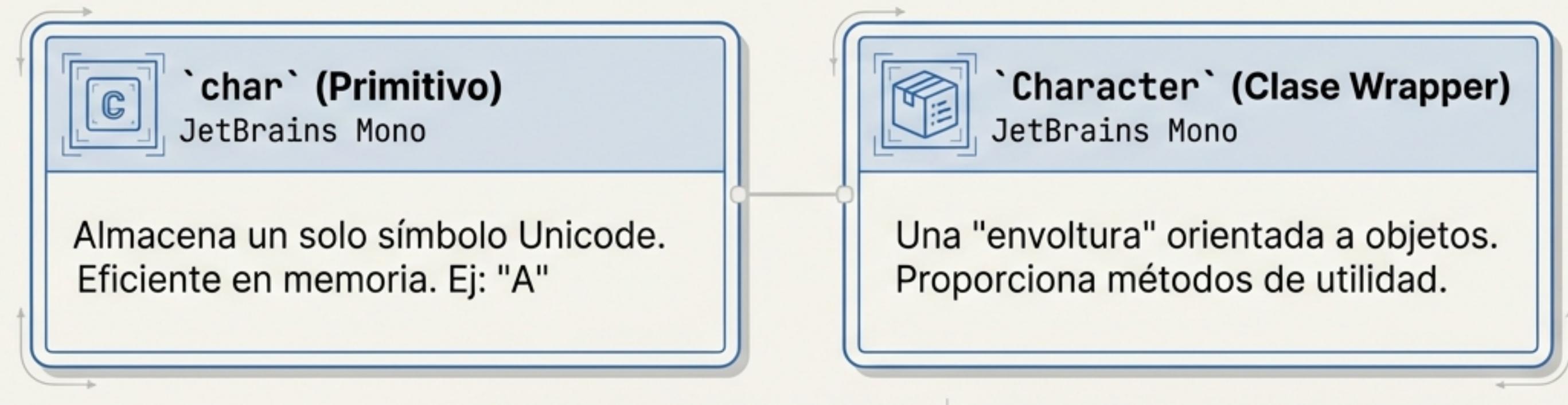


## Iteración anidada

```
int[][] matriz = new int[3][4];  
  
// Iteración anidada  
for (int i = 0; i < 3; i++) { // Filas  
    for (int j = 0; j < 4; j++) { // Columnas  
        procesar(matriz[i][j]);  
    }  
}
```

**\*\*Analogía Real\*\*:** Un avión. Dimensión 1 = Número de Vuelo. Dimensión 2 = Fila. Dimensión 3 = Asiento.

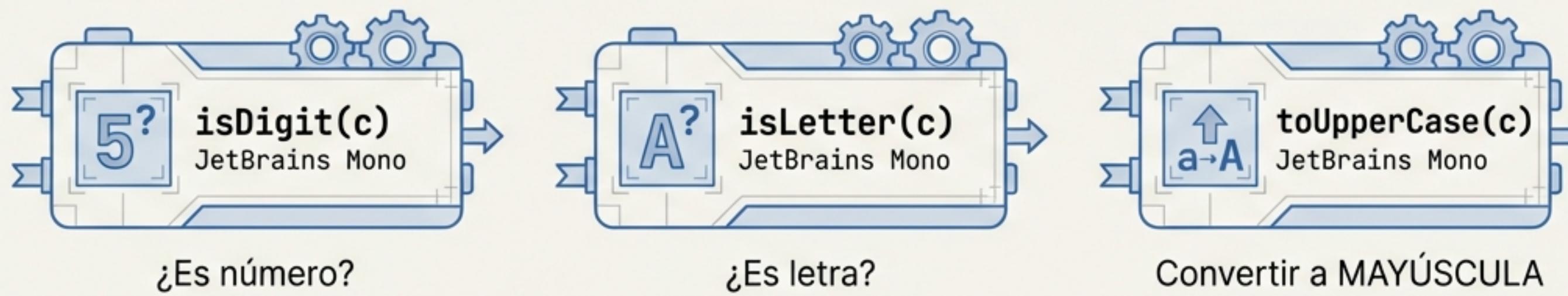
# La Unidad Básica: `char` y `Character`



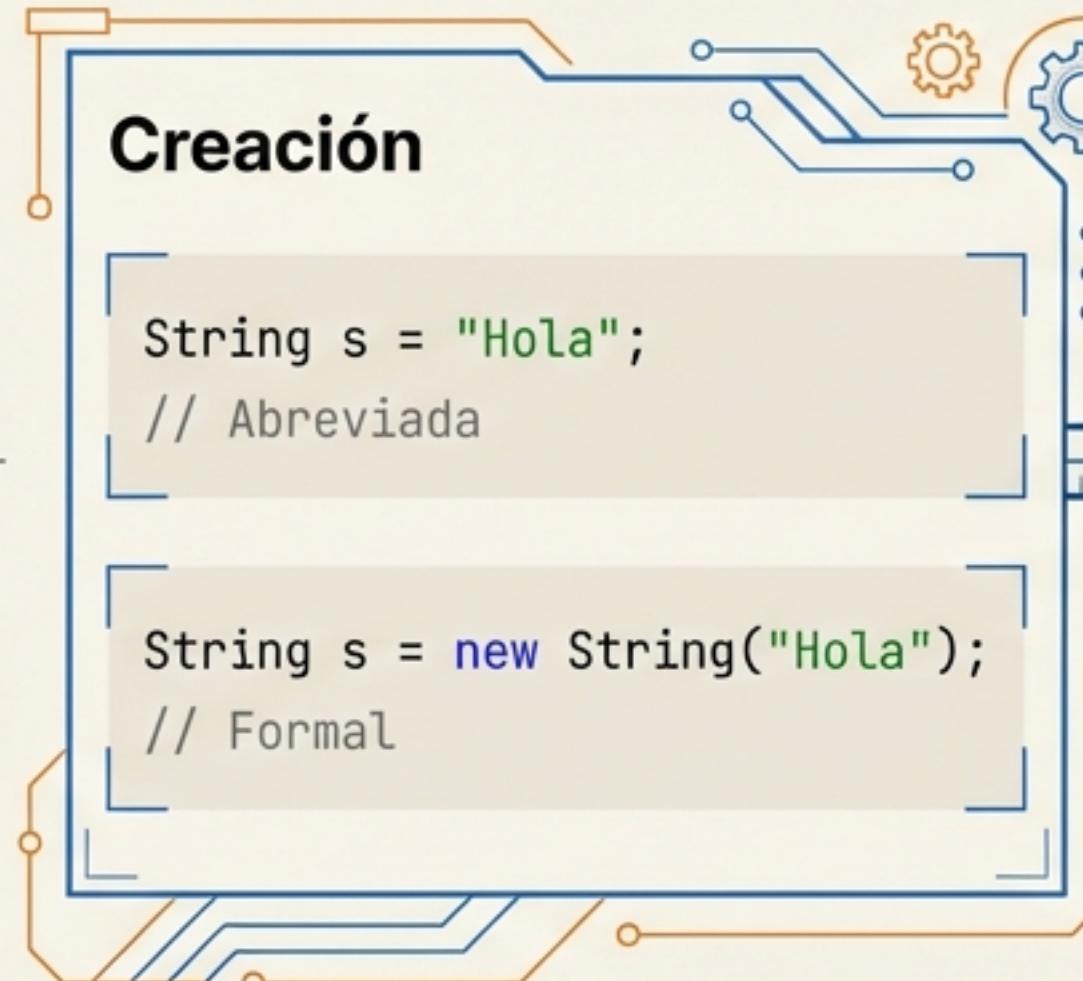
## Secuencias de Escape

- \n : Nueva línea
- \t : Tabulador
- "" : Comillas

## Herramientas de Character



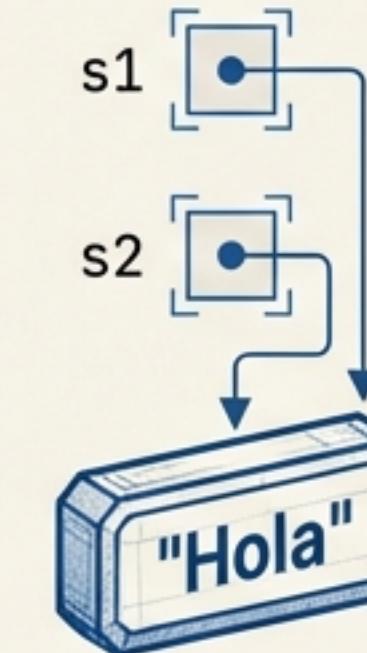
# Cadenas de Texto: La Clase `String`



Una vez creada, no cambia.  
Modificarla crea un objeto nuevo.

## Comparación (¡Importante!)

`==`      `.equals()`



Checks **References**  
(Are they the same  
object in memory?).

Checks **Content**  
(Do they say the  
same thing?).

**Consejo:** Para texto, usa siempre `.equals()`.

# Manipulación de Cadenas: La Navaja Suiza

## Medir

`.length()`



1

## Buscar

`.indexOf("mundo")`



3

## Extraer

`.substring(0, 4)`



2

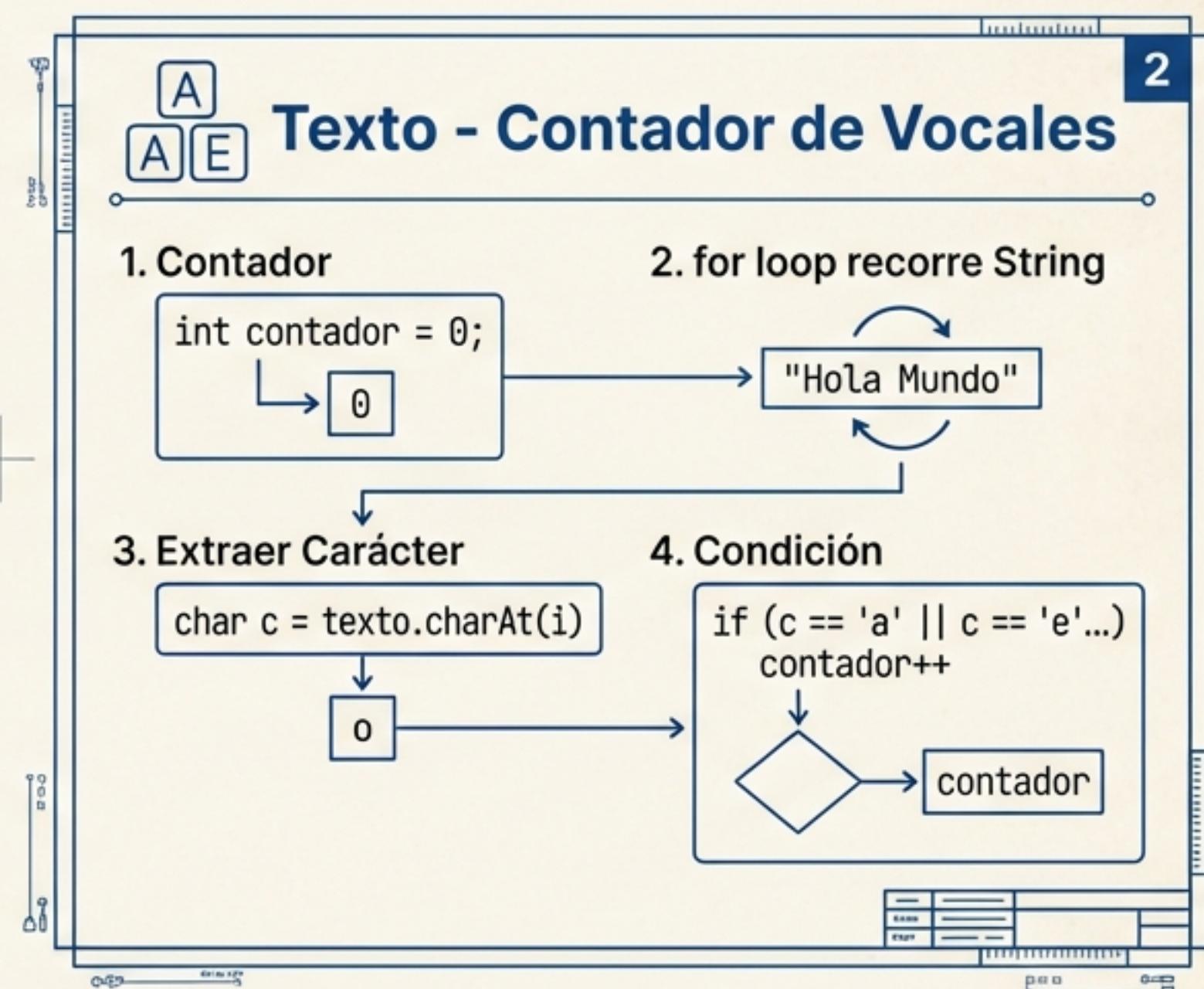
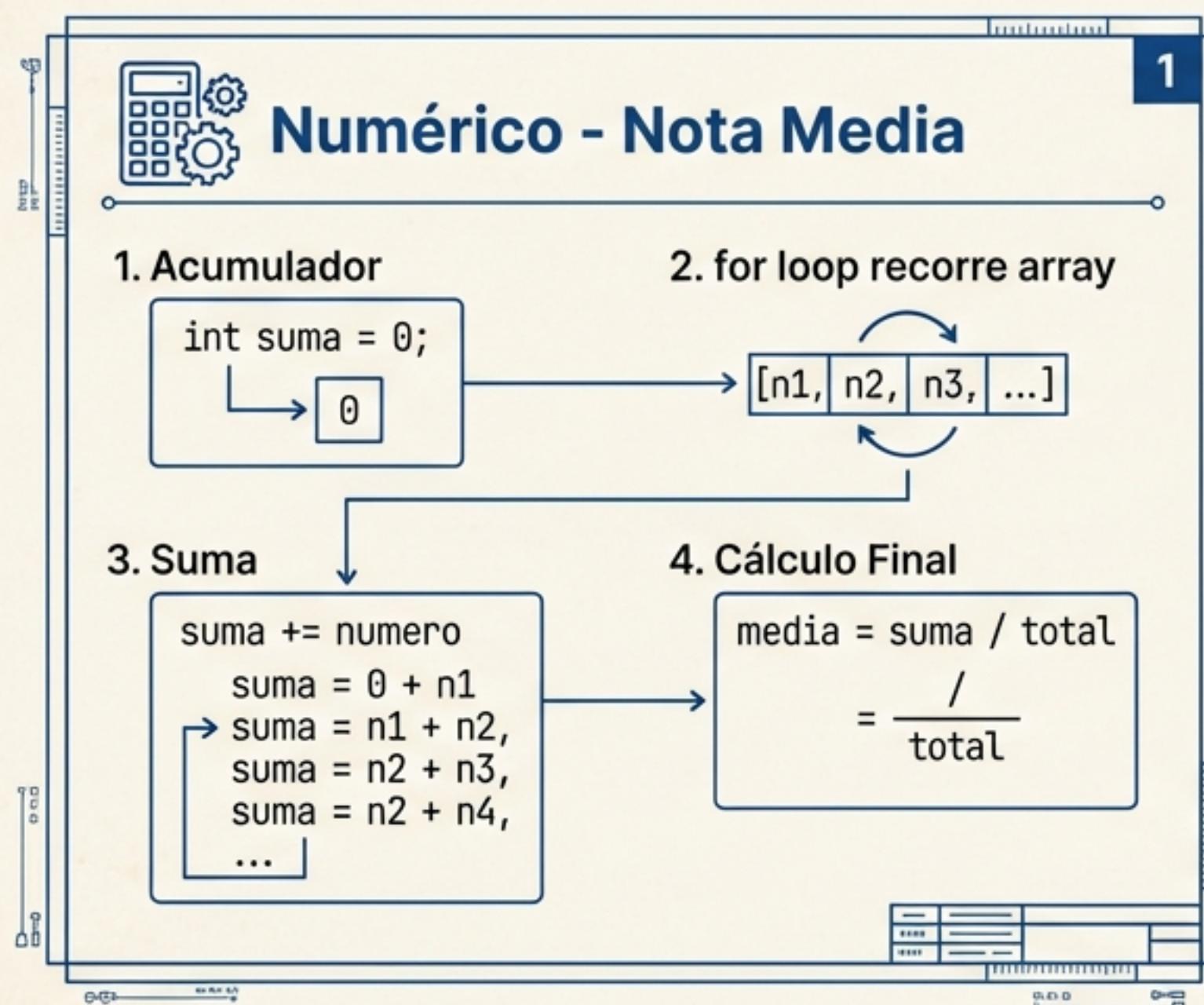
## Acceder

`.charAt(1)`



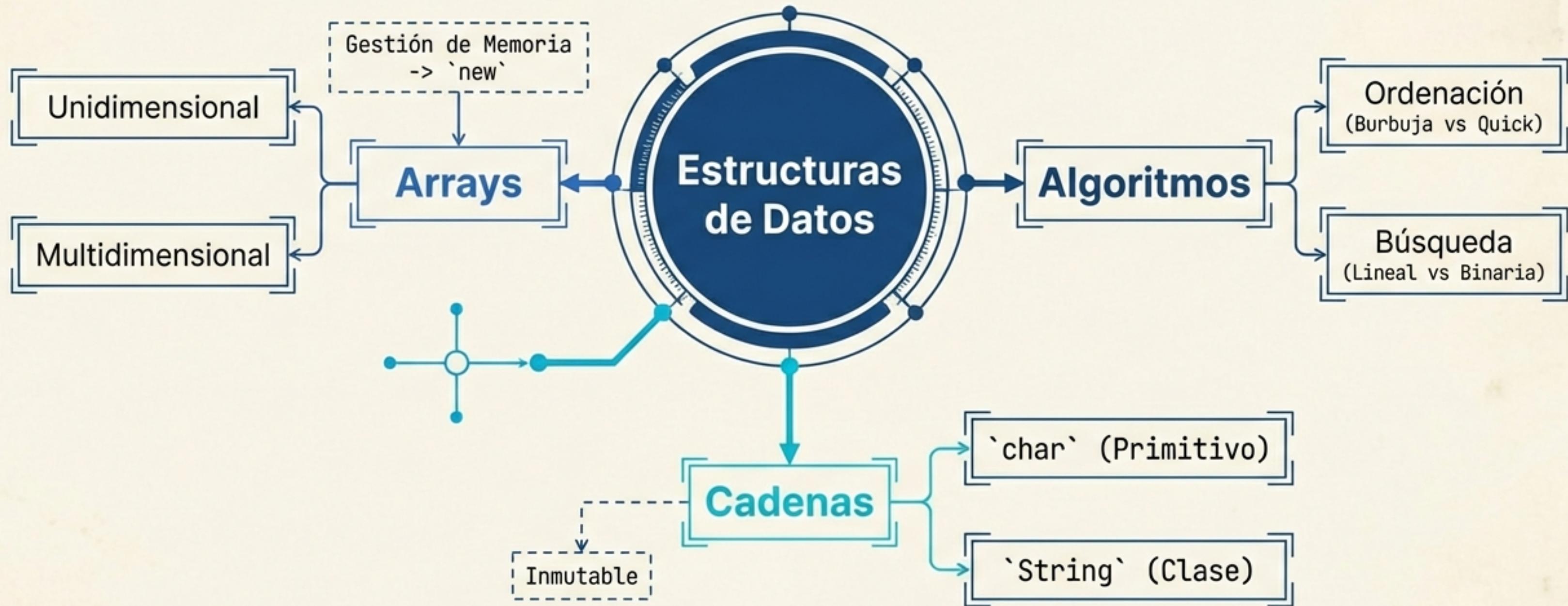
4

# Lógica Aplicada: Casos de Uso



\* Patrón Universal: Iteración + Acumulación/Condición. \*

# Mapa Mental: Estructuras de Datos



Dominar la memoria y los algoritmos es el primer paso para dominar Java.