ARRAYS EN JAVA

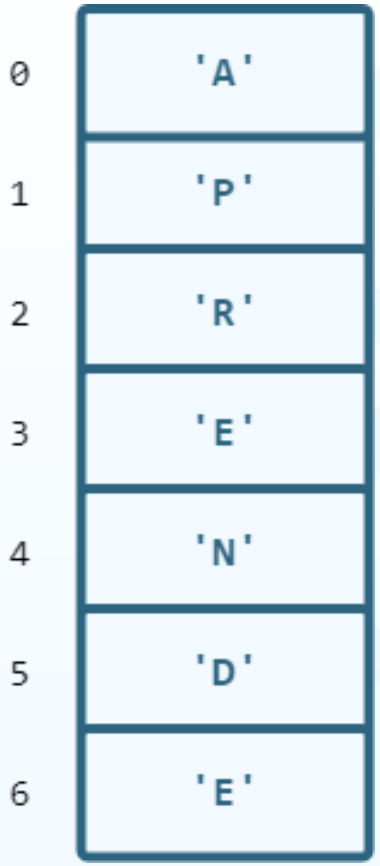
Charly Cimino

Arrays en Java

Charly Cimino

Versión 2023.03.15

Definición



Estructura de datos **estática** que permite guardar elementos **del mismo tipo** en forma contigua.

Permite el acceso a los elementos de forma **aleatoria**, a través de un índice numérico entero que comienza a contarse a partir del valor **0**.

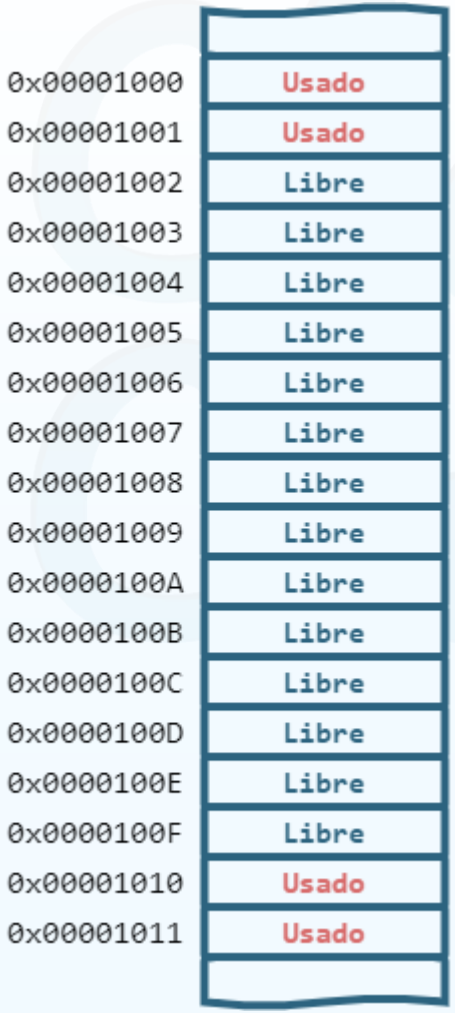
- Array
- Vector
- Matriz unidimensional
- Arreglo (mala traducción del inglés)

Sinónimos

En la RAM...

La posibilidad de construir un array de N elementos depende de la disponibilidad de celdas de memoria adyacentes en la RAM en tal instante y el tamaño en bytes de cada elemento.

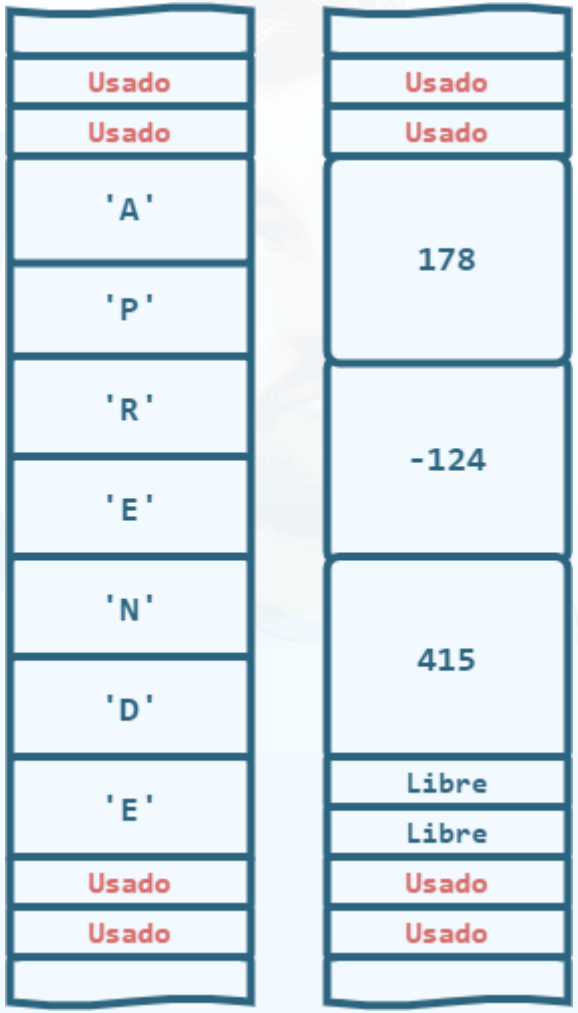
Supongamos que esta es la única parte libre de la memoria...



Array de 7 caracteres.

1 char = 2 bytes.

(En lenguajes derivados de C)



Array de 3 enteros.

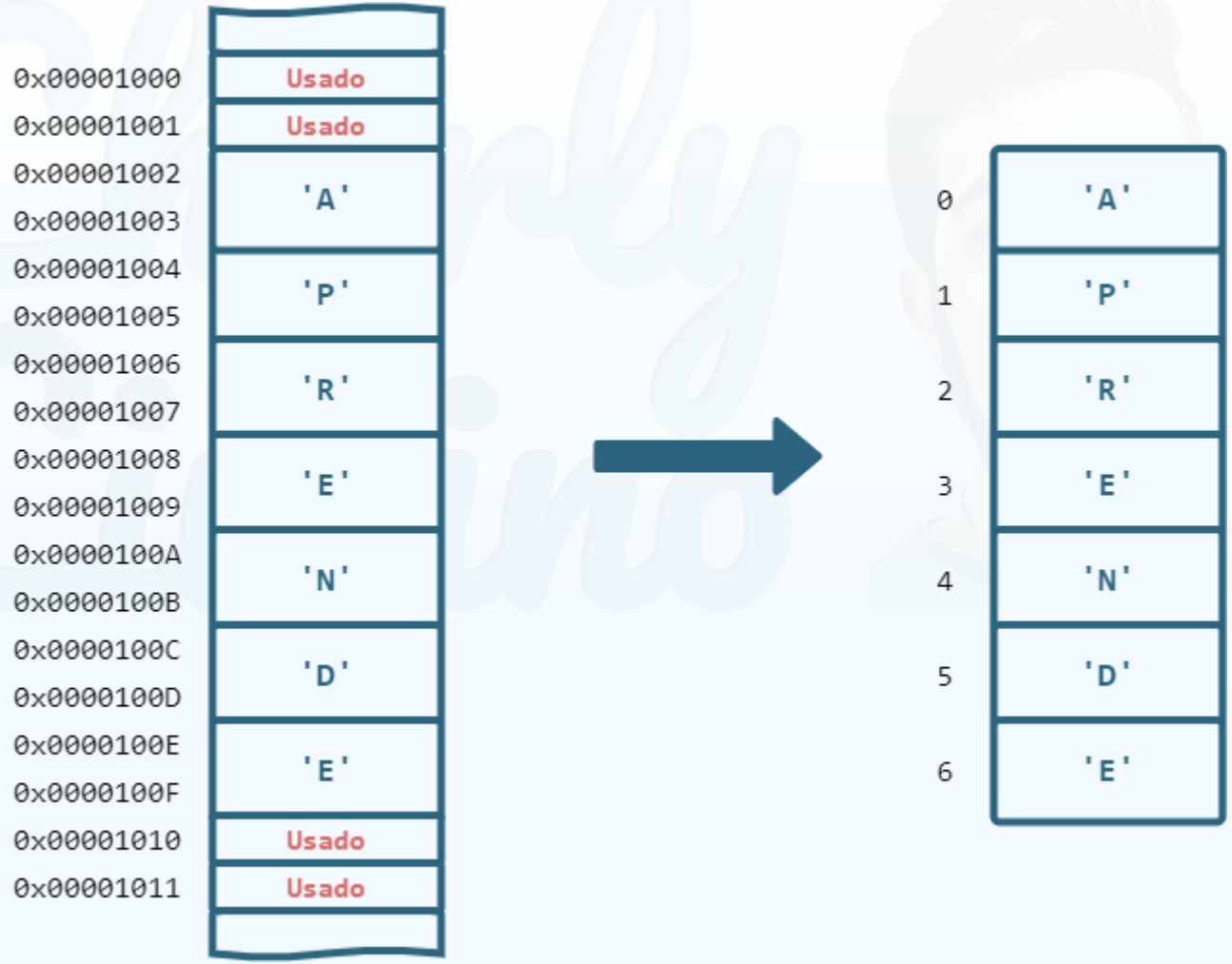
1 int = 4 bytes.

(En lenguajes derivados de C)

En la práctica...

Como programadores de alto nivel, podemos abstraernos de la implementación a bajo nivel y trabajar con arrays de manera abstracta, utilizando índices lógicos en lugar de direcciones de memoria (punteros).

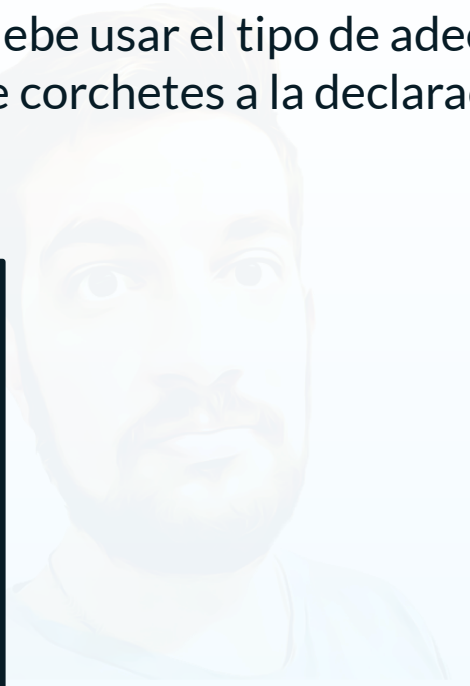
En lenguajes de alto nivel (Java, C#, Python, JavaScript, etc...) el programador se desentiende del manejo de memoria.



Tipo de dato ‘array’

Para poder trabajar con variables que apunten a arrays, se debe usar el tipo de adecuado.
Se pueden tener arrays de ‘lo que sea’ añadiendo un par de corchetes a la declaración.

```
int valor;
int[] valores;
boolean[] banderas;
char[] letras;
String[] palabras;
NaveEspacial[] naves;
```



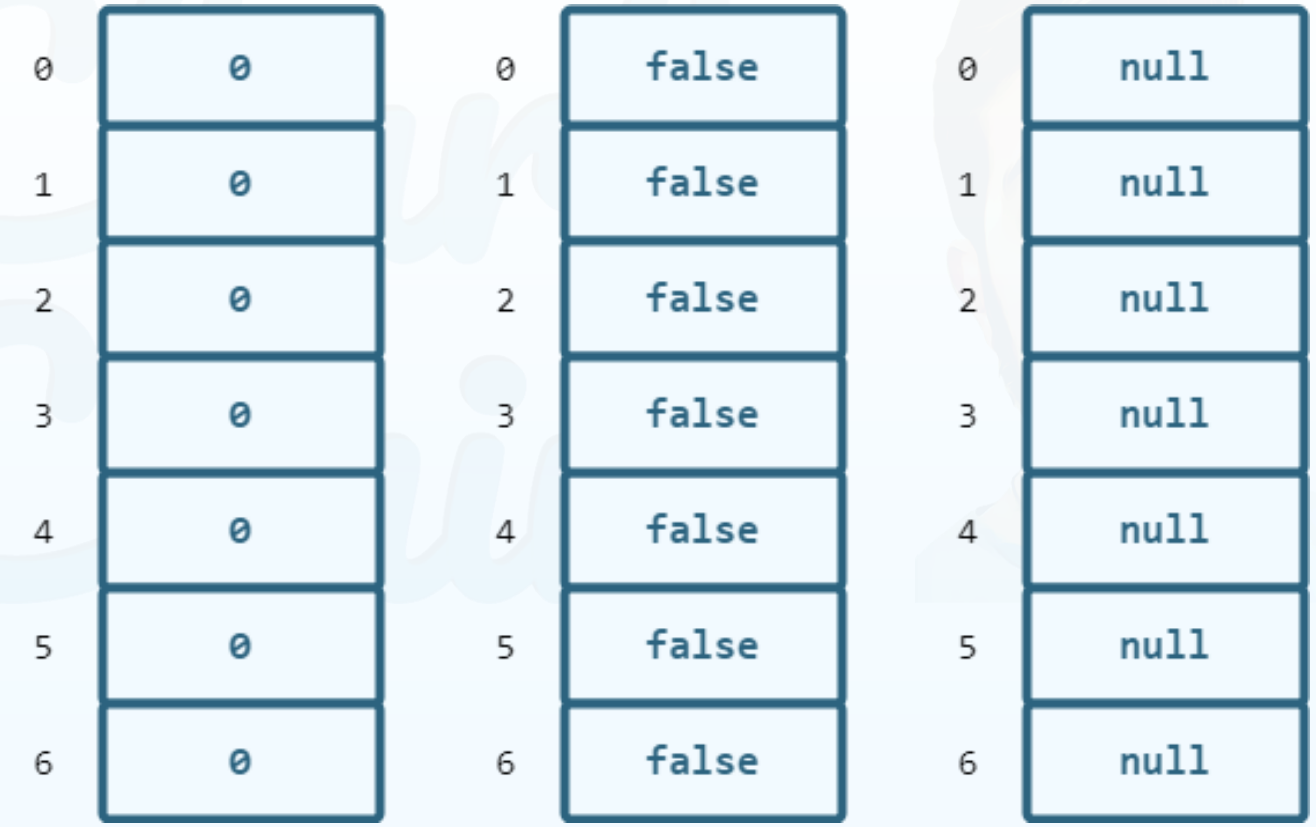
Crear un array

En Java, un array es un **objeto**. Como tal, debe usarse el operador **new** para crear una nueva instancia.
Los arrays son de longitud fija, la cual debe definirse en la creación, siendo inmutable.

```
int[] valores = new int[5]; // Una variable que apunta a un array de 5 enteros

boolean[] banderas = new boolean[3]; // Una variable que apunta a un array de 3 booleanos

int tam = 7;
char[] letras = new char[tam]; // Una variable que apunta a un array de 7 caracteres

int n = (new Scanner(System.in)).nextInt(); // Lee un entero desde la consola
String[] palabras = new String[n]; // Una variable que apunta a un array de n cadenas
```

Valores por defecto en los arrays

Los valores que toma cada celda de un array por defecto depende del tipo de dato.
Utiliza el mismo criterio de los atributos sin inicializar de un objeto.



Numéricos

(byte, short, int, long, float, double, char)

Lógicos

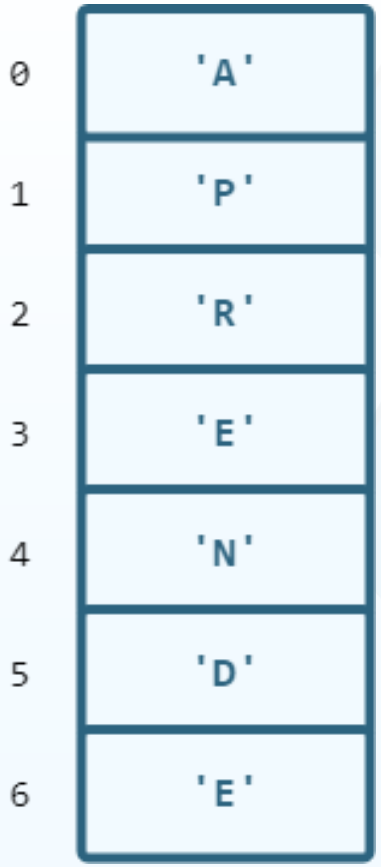
(boolean)

Objetos

(String y demás...)

Crear un array ya inicializado

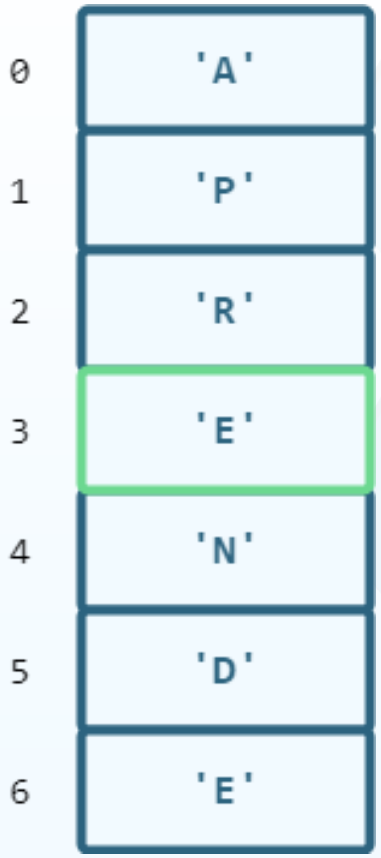
En Java, es posible crear un array inicializado con valores definidos.




```java
// Se listan los valores separados por comas entre un par de corchetes
char[] letras = {'A','P','R','E','N','D','E'};
```

```java
// Si no se declara y asigna en la misma línea no funciona…
char[] letras;
letras = {'A','P','R','E','N','D','E'};
```

Obtener un valor

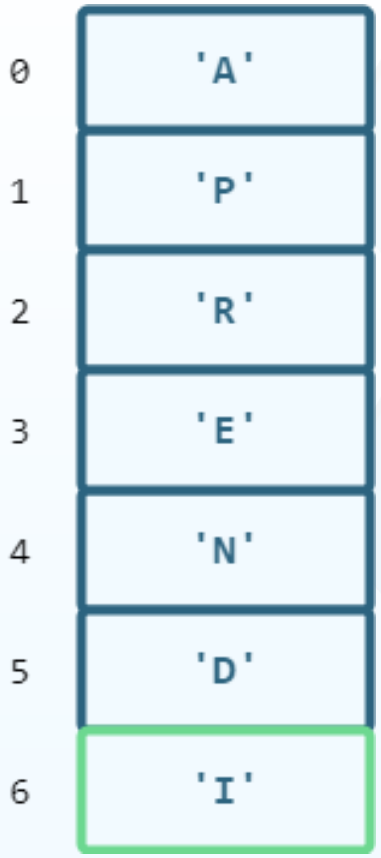



```
char[] letras = {'A','P','R','E','N','D','E'};
System.out.println( letras[3] ); // Muestra una 'E'
```

```
// Un índice fuera de rango provoca una excepción…
System.out.println( letras[-1] );
System.out.println( letras[7] );
```

ArrayIndexOutOf
BoundsException

Establecer un valor




```
char[] letras = {'A','P','R','E','N','D','E'};
letras[6] = 'I';
```

```
// Un índice fuera de rango provoca una excepción…
letras[-4] = '@'; ⊗
letras[37] = '3'; ⊗
```

ArrayIndexOutOf
BoundsException

Obtener longitud

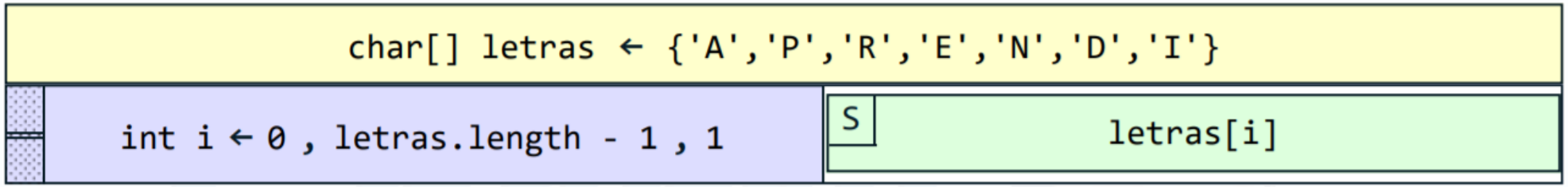

```
char[] letras = {'A','P','R','E','N','D','I'};
System.out.println( letras.length ); // Muestra 7
```

```
// Es un atributo, no un método…
letras.length ✓
letras.length() ⊗
```

```
// Es final, no se puede reasignar…
letras.length = 250; ⊗
```

¿Si es `final`, por qué `length` no va en mayúsculas? 🤷

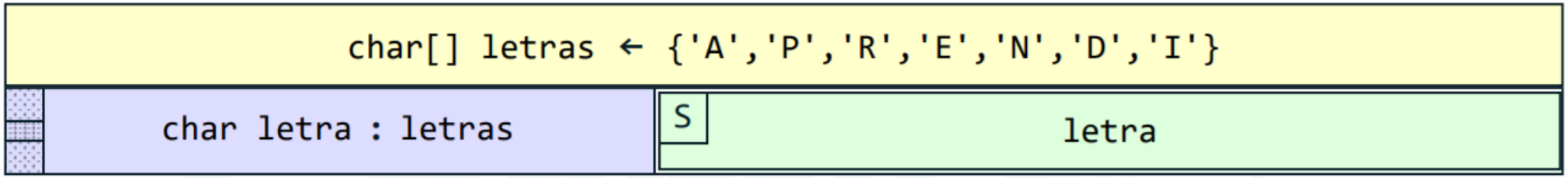
Recorrer todos los elementos



Usando ciclo for

```
char[] letras = {'A','P','R','E','N','D','I'};
for (int i = 0; i < letras.length; i++) {
    System.out.println(letras[i]);
}
```

Recorrer todos los elementos



Usando ciclo foreach

```
char[] letras = {'A','P','R','E','N','D','I'};
for (char letra: letras) {
    System.out.println(letra);
}
```

Se puede recorrer con **foreach** porque en Java los arrays son **objetos**.
En otros lenguajes no lo son, por lo que usar **for** es lo más común.

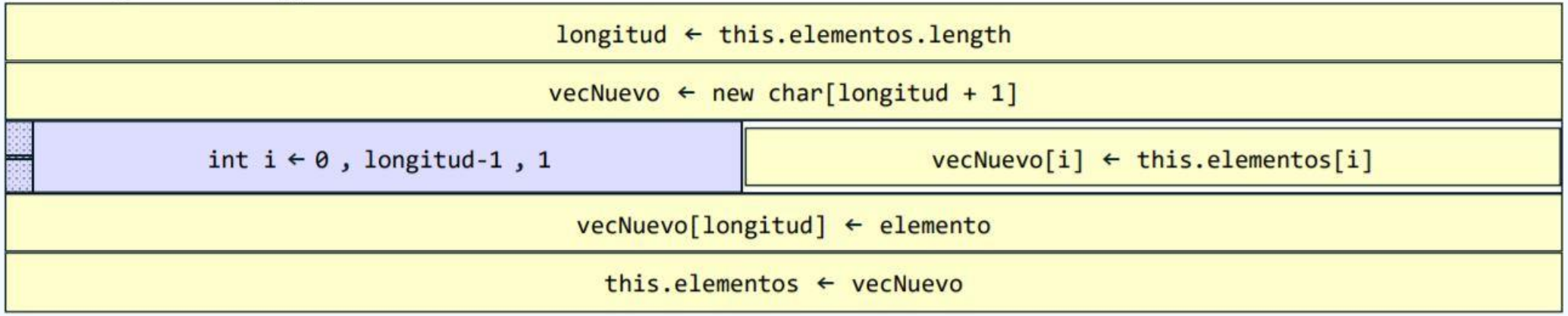
Agregar o quitar elementos

Al ser una estructura estática, un array no puede cambiar de longitud en tiempo de ejecución.

Entonces, ¿cómo hace ArrayList para agregar?

Una implementación posible, si no hay más espacios disponibles para alojar elementos, es generar un nuevo array de mayor longitud a partir del existente.


```
class MiArrayListDeChars :
    public void add ( char elemento )
 int longitud    char[] vecNuevo
```



¿Cuándo usar arrays?

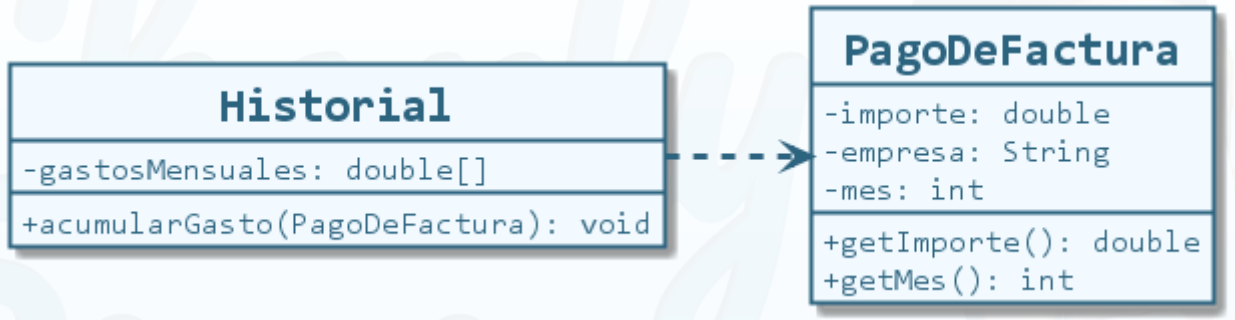
"Un torneo consta de 20 equipos..."

"Un auto tiene 4 ruedas..."

"Una guardería almacena como máximo 10 embarcaciones..."

Posicionamiento directo

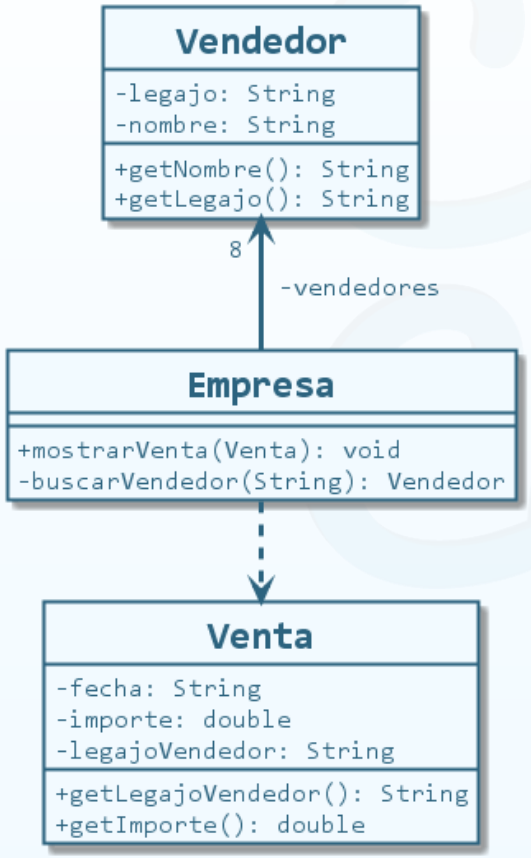
Cuando se puede calcular la posición de determinado valor en un array, se puede acceder a él directamente.




```
class Historial :
    public void acumularGasto ( PagoDeFactura unPago )
                this.gastosMensuales[unPago.getMes() - 1] += unPago.getImporte()
```

Posicionamiento indirecto

Cuando no se puede calcular la posición de determinado valor en un array, debe buscarse en la estructura.




```
class Empresa :
    public void mostrarVenta ( Venta venta )
Vendedor vendedor
    vendedor ← buscarVendedor( venta.getLegajoVendedor() )
    vendedor != null
V                                                    F
S  vendedor.getNombre() + " vendió un monto de $" + venta.getImporte()
                                                     S  "Vendedor desconocido"
```

```
class Empresa :
    private Vendedor buscarVendedor ( String legajo )
int i ← 0    Vendedor encontrado ← null
    i < this.vendedores.length && encontrado == null
        this.vendedores[i].getLegajo() == legajo
    V                                                F
        encontrado ← this.vendedores[i]              i++
    return encontrado
```

ArrayList vs. Array

	ArrayList	Observaciones	Array	Observaciones
Tipo	Dinámico		Estático	
Declaración	`ArrayList<Character> letras`	Para tipos primitivos, hay que usar clases envoltorio (*wrapper classes*)	`char[] letras`	
Instanciación	`letras = new ArrayList<>()`	Crea un AL lógicamente vacío a partir de un array de longitud 10.	`letras = new char[4]`	Crea un array de 4 posiciones con valores por defecto.
	`letras = new ArrayList<>(length)`	Crea un AL lógicamente vacío a partir de un array de la longitud indicada como argumento.	`letras = {'A','B','C','D'}`	Crea un array de 4 posiciones con valores inicializados
Longitud	`letras.size()`		`letras.length`	
Obtener valor	`letras.get(0)`	Obtiene el primer valor	`letras[0]`	Obtiene el primer valor
Establecer valor	`letras.set(0, 'Z')`	La primera letra será `'Z'`	`letras[0] = 'Z'`	La primera letra será `'Z'`
Agregar valor	`letras.add('!')`	Agrega un `'!'`	No soportado	
Quitar valor	`letras.remove(0)`	Quita el primer valor	No soportado	

FIN DE LA PRESENTACIÓN

Encontrá más como estas en mi sitio web.