

# Tema 7 – S1

## Contenidos:

### 1. Introducción:

- ¿Para qué necesito un array?
- Arrays en Java: declaración, creación, inicialización, atributo `length` y operador `[]`
- Manipulación de arrays: excepciones, asignación, copia y paso por valor

### 2. Representación y tratamiento de datos mediante un array

- Representación y operaciones básicas... cuando importa la posición (ejemplo del Hospital) y cuando **NO** (ejemplos de la Lista de Enteros y la Agenda de Teléfonos)



BlueJ: ejemplos – Tema 7

**Descarga** (desde mi carpeta Tema 7 de PoliformaT), **descomprime** y prepárate para usar el proyecto BlueJ *ejemplos – Tema 7*

# Introducción: ¿para qué necesito un array? (I)

**PROBLEMA:** dada la secuencia  $T6 = (4, 7, 10, 13, 16, 19)$

- a) calcular la suma de los valores de  $T6$
- b) comprobar si el **tercer** elemento de  $T6$  es par
- c) enumerar  $T6$  inversamente
- d) intercambiar el **6º** elemento de  $T6$  con el **7º**

☞ **Tipo del dato**  $T6$ : **secuencia** o **lista** o **grupo** o **conjunto** de datos

☞ **Tipo de los datos** de  $T6$ : todos el mismo (**homogéneo**)

- Acceso **secuencial** a los datos, desde el **1º** al **6º**
- Acceso **directo** a un dato indicando su posición

En Java

¿?

int

# Introducción: ¿para qué necesito un array? (II)

Una posible solución Java al problema sería ...

**Pero NO vale**

```
public class SumarValores16 {  
  
    public static void main(String[] args) {  
  
        int 16_1, 16_2, 16_3, 16_4, 16_5, 16_6, suma_16;  
        Scanner teclado = new Scanner(System.in);  
        System.out.println("Escribe seis enteros:");  
        16_1 = teclado.nextInt(); 16_2 = teclado.nextInt();  
        16_3 = teclado.nextInt(); 16_4 = teclado.nextInt();  
        16_5 = teclado.nextInt(); 16_6 = teclado.nextInt();  
        suma_16 = 0;  
        suma_16 += 16_1; suma_16 += 16_2;  
        suma_16 += 16_3; suma_16 += 16_4;  
        suma_16 += 16_5; suma_16 += 16_6;  
        System.out.println("Suma de 16 es" + suma_16);  
    }  
}
```

¿Sirve para 1100?  
¿Puede usarse  
un bucle?

# Introducción: ¿para qué necesito un array? (III)

**GENERALIZACIÓN del problema:** dada la secuencia  $\mathcal{T} = (4, 7, \dots, 16, 19)$  de longitud  $N\_MAX$ : calcular la suma de los valores de  $\mathcal{T}$ ; comprobar si el **tercer** elemento de  $\mathcal{T}$  es par; enumerar  $\mathcal{T}$  inversamente; intercambiar el **6º** elemento de  $\mathcal{T}$  con el **7º**

- ☞ Se necesita **una sola variable**  $\mathcal{T}$  con la que identificar y manejar un grupo de  $N\_MAX$  elementos del mismo tipo
- ☞ Se necesita **acceder a cada uno** de los  $N\_MAX$  elementos del grupo

## Solución matemática: **vector**

- $\mathcal{T}$  es un **vector** de  $N\_MAX$  componentes **de tipo Entero**
- $\forall i: 1 \leq i \leq N\_MAX: \mathcal{T}_i$  es la  $i$ -ésima **componente** del vector  $\mathcal{T}$

## Solución informática: **array**

- $\mathcal{T}$  es un **array** de  $N\_MAX$  componentes **de tipo `int`**
- $\forall i: 0 \leq i < N\_MAX: \mathcal{T}[i]$  es la  $i$ -ésima **componente** del array  $\mathcal{T}$

# Introducción: un array en Java es ...

- Un **objeto** que representa en posiciones consecutivas de memoria un grupo de **datos de tipo homogéneo**
- Un **objeto** que se maneja mediante la correspondiente **variable referencia** (por ejemplo T6)
- Un **objeto** al que solo se le puede aplicar un operador, **[]**, el **operador de indexación**
  - La indexación comienza siempre en **0**
  - Asociando un **índice i** a cada elemento ó componente de un array,  $0 \leq i$ , este operador permite acceder al **i**-ésimo elemento del array de forma directa (por ejemplo T6[0] , T6[1] , ...)
- Un **objeto** que solo tiene un atributo, **length**, que indica su nº de componentes (por ejemplo T6.length = 6)
  - Se establece al crear el array vía **new**
  - Permanece invariable durante la ejecución

# Introducción: declaración, creación e inicialización de arrays

- **Versión 1, vía new**

```
tipoBase[] identificador;
```

```
identificador = new tipoBase[TAMAÑO];
```

Reserva **estática** de memoria, i.e. conocida ya en tiempo de compilación

O también, en la misma línea:

```
tipoBase[] identificador = new tipoBase[TAMAÑO];
```

**MUY IMPORTANTE:** **todas** las componentes del array **se inicializan** a un mismo valor, **el valor por defecto de dicho tipo en Java** (null para tipos referencia, 0 para los tipos numéricos, espacio en blanco ' ' para el tipo char y false para el tipo boolean)

- **Versión 2, vía {}**

```
tipoBase[] identificador;
```

```
identificador = {var_0, var_2, var_3, ... var_(Tamaño - 1)};
```

**MUY IMPORTANTE:** **cada** componente del array **se inicializa a un valor distinto**, el que determina su posición en la lista inicialización

# Introducción: declaración, creación e inicialización de arrays

## Ejemplos con arrays de tipo base PRIMITIVO (I)

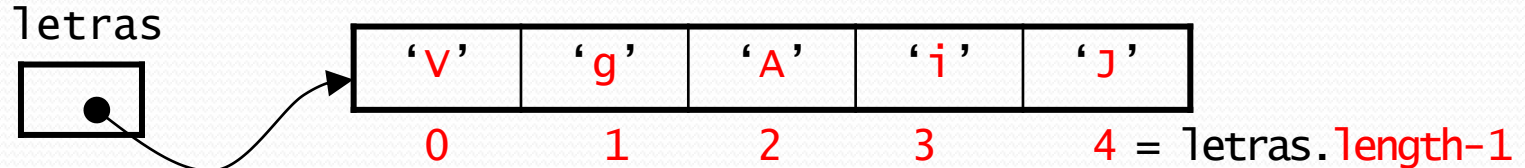
- Array de **5** letras:

// declaración, creación e inicialización a ' ' de todos los elementos

```
char[] letras = new char[5];
```

// declaración, creación e inicialización de cada elemento a cierto valor char

```
char[] letras = {'v', 'g', 'A', 'i', 'J'}
```



- Arrays de enteros:

```
int[] numeros1 = new int[5000], numeros2 = new int[50];
```

- Array de **20** nombres:

```
final static int NUM = 10;
```

```
String[] nombres = new String[NUM * 2];
```

- Array cuyo **TAMAÑO**, o nº de componentes, se lee del teclado:

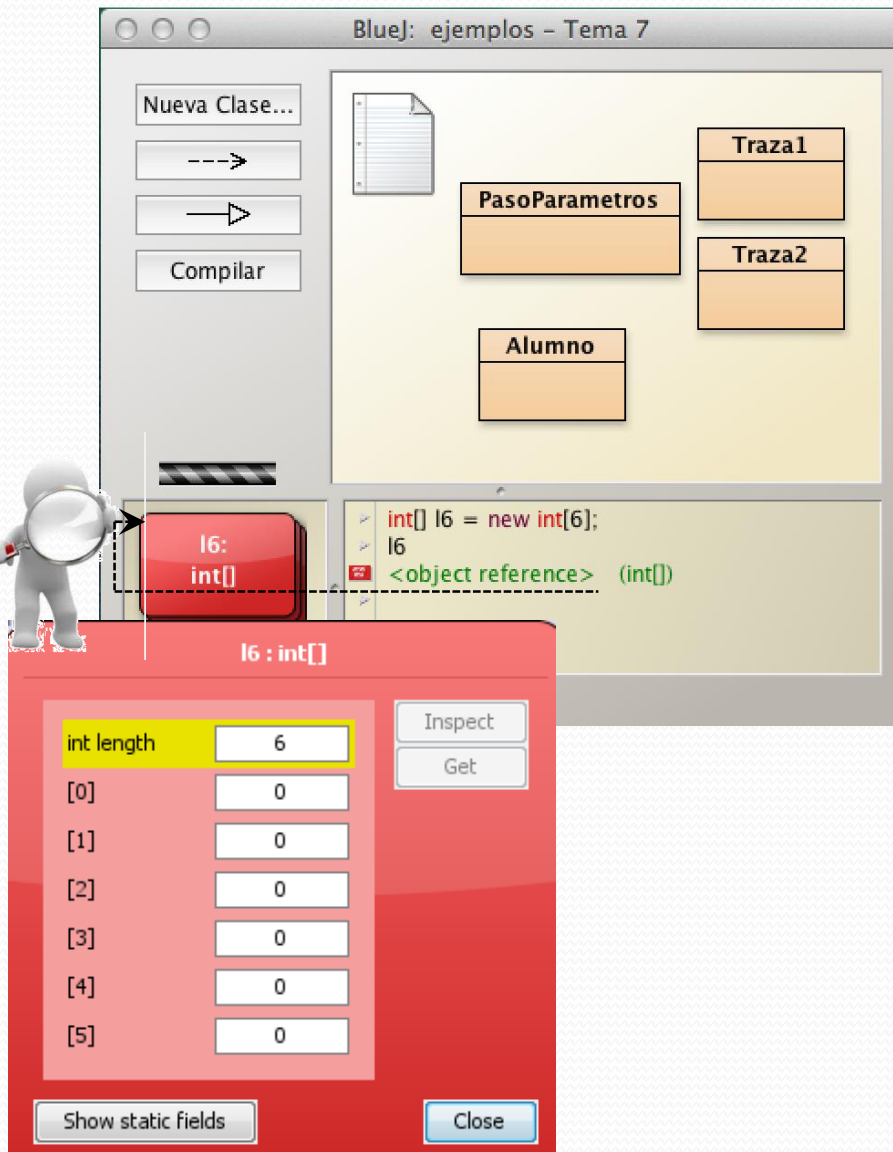
```
double[] precios = new double[teclado.nextInt()];
```

- Array de **4** enteros con los valores **-5, 6, 10** y **3**:

```
int[] v = {-5, 6, 10, 3};
```

# Introducción: declaración, creación e inicialización de arrays

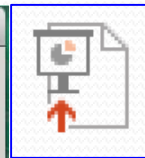
## Ejemplos con arrays de tipo base PRIMITIVO (II)



The screenshot shows the BlueJ IDE interface. In the center, there's a diagram with boxes for 'PasoParametros', 'Alumno', 'Traza1', and 'Traza2'. On the left, there are buttons for 'Nueva Clase...', 'Compilar', and a magnifying glass icon. Below the diagram, a red box highlights the variable 'l6: int[]'. A small figure with a magnifying glass is pointing at this box. Below the red box, a red dialog box titled 'l6 : int[]' is open, showing the array's contents. The dialog has a table with 'int length' and '6', and a list of indices from [0] to [5], each with a value of 0. There are 'Inspect' and 'Get' buttons, and 'Show static fields' and 'Close' buttons at the bottom.

```
int[] l6 = new int[6];  
l6  
<object reference> (int[])
```

int length	6
[0]	0
[1]	0
[2]	0
[3]	0
[4]	0
[5]	0



La variable **l6** **referencia** al objeto  
... **PERO NO ES** el objeto

**SINO una variable Referencia**

El **objeto array** al que referencia **l6** **REPRESENTA** en memoria **un agregado** (o grupo, o conjunto, o lista o secuencia) de **6** elementos de tipo **int**...

- En posiciones **contiguas** de memoria (+1 posición a partir de la primera)
- Con **acceso directo a cada elemento**, que puede ser tratado a todos los efectos como cualquier otra variable de tipo **int**



# Introducción: declaración, creación e inicialización de arrays

## Ejemplo con array de tipo base REFERENCIA – array de objetos

The screenshot shows the BlueJ IDE interface. At the top, the title bar reads "BlueJ: ejemplos - Tema 7". On the left, there are buttons for "Nueva Clase...", a dashed arrow, a solid arrow, and "Compilar". The main workspace displays a class hierarchy with boxes for "PasoParametros", "Alumno", "Traza1", and "Traza2". A small icon of a document with a red arrow is highlighted in the top right corner. Below the workspace, a code editor shows the following Java code:

```
Alumno[] grupo = new Alumno[10];
grupo
<object reference> (Alumno[])
grupo[0] = new Alumno("Gutierrez Sendra, Joaquin", 21544000);
```

Below the code editor, two variable inspection windows are open. The first window, titled "grupo : Alumno[]", shows the array's state:

int length	10
[0]	
[1]	null
[2]	null
[3]	null
[4]	null
[5]	null
[6]	null

The second window, titled "grupo\_0 : Alumno", shows the details of the object at index 0:

private long dni	21544000
private double nota	0.0
private String nombre	"Gutierrez Sendra, Joaquin"
private boolean asistencia	true
private char grupo	' '

Both windows have buttons for "Show static fields" and "Cerrar".

La variable **grupo** referencia al objeto... **PERO NO ES** el objeto **SINO** una variable Referencia

El **objeto** array al que referencia **grupo** REPRESENTA en memoria un **agregado** (o grupo, o conjunto, o lista o secuencia) de **10** elementos de tipo **Alumno**...

- En posiciones **contiguas** de memoria (+1 posición a partir de la primera)
- Con **acceso directo a cada elemento**, que puede ser tratado a todos los efectos como cualquier otra variable de tipo **Alumno**

# Introducción

## Manipulación de arrays: Excepciones

Los **índices válidos** para el acceso a las componentes de un array pertenecen al intervalo  $[0, a.length - 1]$ . Un acceso con cualquier índice fuera de este intervalo provocará el error de ejecución, o excepción, **ArrayIndexOutOfBoundsException**

The screenshot shows the BlueJ IDE window titled "BlueJ: ejemplos - Tema 7". The interface includes a left sidebar with buttons for "Nueva Clase...", a dashed arrow, a solid arrow, and "Compilar". Below these is a red button labeled "l6: int[]". The main workspace displays a diagram with four objects: "PasoParametros", "Alumno", "Traza1", and "Traza2". The "Alumno" object is connected to "PasoParametros". The bottom panel shows the following code:

```
int[] l6 = new int[6];  
l6  
<object reference> (int[])  
l6[6]  
Exception: java.lang.ArrayIndexOutOfBoundsException (6)  
l6[-1]  
Exception: java.lang.ArrayIndexOutOfBoundsException (-1)  
|
```

At the bottom of the window, the status bar indicates "Compilando... Listo".

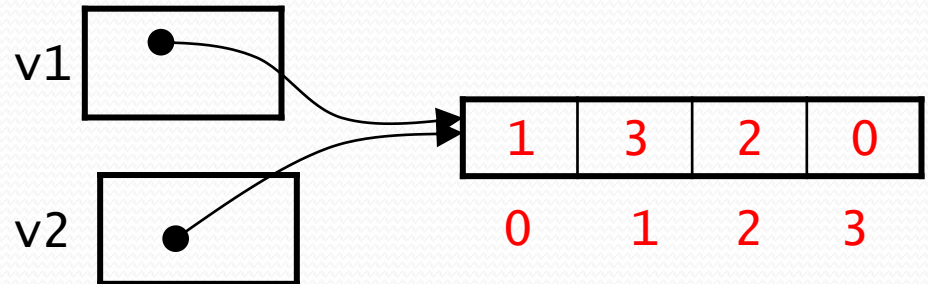
## Manipulación de arrays: asignación y copia

- La  $i$ -ésima componente de un array  $v$  es a todos los efectos una variable del tipo base de  $v$ , por lo que se le puede asignar cualquier expresión del mismo tipo o compatible:  $v[i] = \text{expresión}$ ;
- La asignación entre arrays **sólo afecta a las referencias**.

```
int[] v1 = {1, 3, 2, 0};
```

```
int[] v2;
```

```
v2 = v1;
```

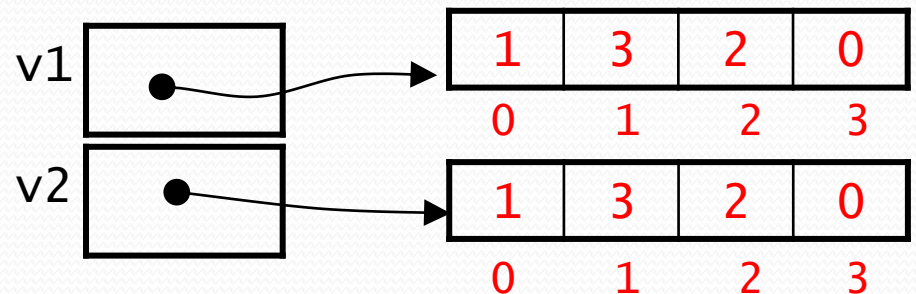


- Si se desea una **copia de un array**  $v1$  se debe:
  - (1) Crear uno nuevo  $v2$ ;
  - (2) Vía asignación, copiar en el nuevo, una a una, las componentes de  $v1$

```
v2 = new int[4];
```

```
v2[0] = v1[0]; v2[1] = v1[1];
```

```
v2[2] = v1[2]; v2[3] = v1[3];
```



## Manipulación de arrays como parámetros de un método (I)

- Un array puede ser un parámetro formal de un método:

```
public static int metodo1(int[] v1, int[] v2) { ... }  
public static void main(String[] args) { ... }
```

- En la llamada al método sólo se pasa el nombre de la variable como argumento. Como el paso de parámetros es por valor, los parámetros (v1 y v2) se inicializan a las referencias de los argumentos (a1 y a2)

```
int[] a1 = new int[10], a2 = new int[5];  
...  
int i = metodo1(a1, a2);
```

- Los métodos pueden devolver como resultado (la referencia a) un array:

```
public static char[] metodo2(int[] v1) {  
    char[] res = new char[v1.length + 10];  
    ...  
    return res;  
}
```

**Invocación:**

```
char[] a = metodo2(a1);
```

## Manipulación de arrays como parámetros de un método (II)



BlueJ: ejemplos - Tema 7

- Establece puntos de ruptura en las llamadas a los métodos, ejecuta el main de la clase PasoParametros y observa el estado de la variable elArray para comprobar que **no** se modifica en la llamada a metodo1 y que, sin embargo, **sí** se modifica su primera componente en la llamada a metodo2.

```
public class PasoParametros {  
    public static void main(String[] args) {  
        double[] elArray = {5.0, 6.4, 3.2, 0.0, 1.2};  
        metodo1(elArray);  
        // elArray no se ha modificado  
        metodo2(elArray);  
        // elArray[0] vale ahora 0.1  
    }  
    public static void metodo1(double[] copia) {  
        copia = new double[5]; // desaparece al acabar el método  
    }  
    public static void metodo2(double[] copia) {  
        copia[0] = 0.1; // modificado elArray[0]  
    }  
}
```

# Introducción: Ejercicios propuestos

## Nº 1 Transparencias:



BlueJ: ejemplos - Tema 7

- **Ejecuta**, en este orden, los programas **Traza1** y **Traza2** del proyecto y **argumenta** por qué su resultado es el que es –te puede ayudar poner puntos de ruptura en la invocación a los métodos y observar el estado de las variables durante su ejecución

## Nº 2 Transparencias:



### La clase Reloj (clave CCDJG4ai)

Completa los huecos de la clase Java que se te proporciona para que represente un reloj digital. Al hacerlo ten muy presentes los comentarios que preceden a estos huecos y las convenciones de código Java



### La clase TestReloj (clave CCDJH4ai)

Completa los huecos del programa Java que se te proporciona para que muestre por pantalla el funcionamiento de un reloj durante un día, segundo a segundo. Al hacerlo ten muy presentes los comentarios que preceden a estos huecos y las convenciones de código Java

# Representación y tratamiento de datos con arrays

## Cuando la posición SÍ es relevante...

**PROBLEMA:** introducir en la máquina Java un pastillero semanal y hacer que funcione como tal.



- A Necesitamos un array para representar los datos (declararlo de un cierto tipo)
- B ¿Qué operaciones – básicas- se realizan sobre las componentes del array?



# Representación y tratamiento de datos con arrays

## Cuando la posición SÍ es relevante...

**PROBLEMA:** : introducir en la máquina Java un pastillero semanal y hacer que funcione como tal.

### A ¿Necesitamos un array para representar los datos? (tipo y tamaño)

Para una solución **eficiente**, **SÍ**, necesitamos un array de “**contadores**”

(a) Tipo base: `int`

(b) Tamaño: nº de contadores a representar, uno por elemento en  $[0, 6]$ :  $6 - 0 + 1 = 7$

Ya podemos declarar e inicializar un array de `C_P_D = 7` elementos de tipo `int`:

```
int[] elArray = new int[C_P_D];
```

(c) ¿Qué representa CADA componente del array

`elArray[i]` representa el **contador *i***  
es la **variable** que gestiona las pastillas del día ***i***

Truco que “**aprovecha**” el **acceso directo**  
Observa que la **posición *i*** es **relevante**

### B ¿Qué operaciones – básicas- se realizan sobre las componentes del array?

```
Declarar e inicializar elArray
insertar(i)
eliminar(i)
esVacio(i)
...
```



# Representación y tratamiento de datos con arrays

## Ejercicio Nº 1 Transparencias: implementación de contadores en Java

**PROBLEMA:** contar el nº de veces que un usuario ha escrito en el teclado un nº entero en el intervalo [0, 5]



**PARTE 1:** representación de los datos del problema mediante un array (tipo y tamaño) y operaciones básicas

**PARTE 2:** implementación -eficiente- de un array de contadores en Java - **La clase ArrayDeContadores**

**NOTA:** una vez explicado su diseño (ver la siguiente transparencia) escribe esta clase en el proyecto BlueJ  
*Ejemplos - Tema 7*

# Representación y tratamiento de datos con arrays

## Ejercicio Nº 1 Transparencias – Parte 2: implementación en Java

**PROBLEMA:** contar el nº de veces que un usuario ha escrito en el teclado un nº entero en el intervalo [0, 5]

**SOLUCIÓN eficiente:** array de contadores

```
public class ArrayDeContadores {  
    private static final int C_P_D = 6;  
    public static void main(String[] args) {  
        Scanner teclado = new Scanner(System.in);  
        int[] elArray = new int[C_P_D];  
        int i;  
        do {  
            i = teclado.nextInt();  
            if (i > -1 && i < 6) { elArray[i]++; }  
        } while(i != -1);  
        for (i = 0; i < elArray.length; i++) {  
            System.out.println("Leídos " + elArray[i] + " " + i + "s");  
        }  
    }  
}
```

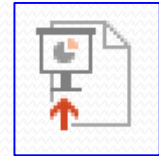
Piensa cómo modificarías este programa si el problema fuese...

Cuenta el nº de veces que a un jugador le ha salido una de las 6 caras de un dado (numeradas del 0 al 5)

## Ejercicio Nº 2 Transparencias: implementación de un conjunto de naturales en Java

**PROBLEMA:** representa un conjunto de números naturales con valores en el intervalo  $[0, n]$ . Por ejemplo, si  $n = 5$ , podríamos representar cualquiera de los siguientes conjuntos:  $\{0, 1, 2, 4, 5\}$ ,  $\{5\}$ ,  $\{0, 1\}$ ,  $\{0, 1, 2, 3, 4, 5\}$

**PARTE 1:** representación de los datos del problema mediante un array (tipo y tamaño) y operaciones básicas



**PARTE 2:** implementación -eficiente- de un conjunto de números naturales en Java - **La clase ConjuntoDeN**

**NOTA:** puedes ver el código de la clase en  
El proyecto BlueJ *Ejemplos - Tema 7*

# Representación y tratamiento de datos con arrays

## Ejercicio Nº 2 Transparencias – Parte 2: implementación en Java

**PROBLEMA:** representa un conjunto de números naturales con valores en  $[0, n]$

**SOLUCIÓN:** array “conjunto”

```
public class ConjuntoDeN {
    private boolean[] elArray;

    /** PRECONDICION:  $n > 0$ . Crea un conjunto vacío de naturales en  $[0, n]$  */
    public ConjuntoDeN(int n) {
        elArray = new boolean[n + 1];
    }

    public int maximo() { return elArray.length - 1; }

    /** PRECONDICION:  $0 \leq i \leq \text{maximo}()$ . ... */
    public void insertar(int i) { elArray[i] = true; }

    /** PRECONDICION:  $0 \leq i \leq \text{maximo}()$ . ... */
    public void eliminar(int i) { elArray[i] = false; }

    /** PRECONDICION:  $0 \leq i \leq \text{maximo}()$ . ... */
    public boolean pertenece(int i) { return elArray[i]; }

    public int cardinal() {
        int talla = 0;
        for (int j = 0; j < elArray.length; j++) {
            if (elArray[j]) { talla++; }
        }
        return talla;
    }
}
```

¿Se puede calcular **más rápido**?

# Representación y tratamiento de datos con arrays

## Ejercicio Nº 2 Transparencias – Parte 2: implementación EFICIENTE

**PROBLEMA:** representa un conjunto de números naturales con valores en  $[0, n]$

**Solución EFICIENTE:**

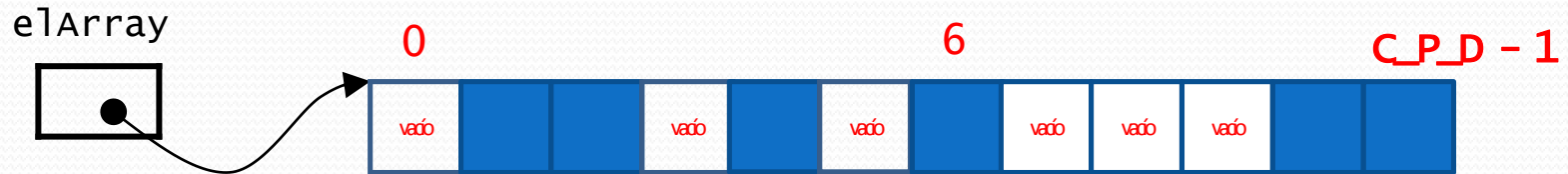
```
public class ConjuntoDeN {
    private boolean[] elArray; private int talla;
    /** PRECONDICION:  $n > 0$ . Crea un conjunto vacío de naturales en  $[0, n]$  */
    public ConjuntoDeN(int n) {
        elArray = new boolean[n + 1]; talla = 0;
    }
    public int maximo() { return elArray.length - 1; }
    /** PRECONDICION:  $0 \leq i \leq \text{maximo}()$ . ... */
    public void insertar(int i) { if (!elArray[i]) { elArray[i] = true; talla++; } }
    /** PRECONDICION:  $0 \leq i \leq \text{maximo}()$ . ... */
    public void eliminar(int i) { if (elArray[i]) { elArray[i] = false; talla--; } }
    /** PRECONDICION:  $0 \leq i \leq \text{maximo}()$ . ... */
    public boolean pertenece(int i) { return elArray[i]; }
    public int cardinal() {
        return talla;
    }
}
```

¿Se puede calcular **más rápido**?

**Sí**, usando el atributo auxiliar **talla**

# Representación y tratamiento de datos con arrays

En resumen, si la posición de los datos en el grupo es relevante...



```
private tipoBase[] elArray; private static final int C_P_D = ...;  
private int talla; // alternativamente, declarar libres
```

- Para **crear** un grupo **vacío**, con **0** datos:

```
elArray = new tipoBase[C_P_D]; talla = 0; // o libres = C_P_D;
```

- Para **insertar** un dato **x** en el grupo:

**buscar** **i**, la 1ª posición libre en `elArray` (`elArray[i] = vacío`)

**si** (**existe**(**i**)): `elArray[i] = x`; `talla++`; **sino** tratar el caso de array **lleno**

- Para **borrar** el dato del grupo en posición **i**:

**si** `elArray[i] != vacío`: `elArray[i] = vacío`; `talla--`;

# Representación y tratamiento de datos con arrays

## Ejercicio Nº 3 Transparencias: implementación de un Hospital en Java

**PROBLEMA:** representa un hospital (de una sola planta) con  $n$  camas; considera que un paciente solo ocupará una cama si está libre y que la cama nº 0 nunca se usará

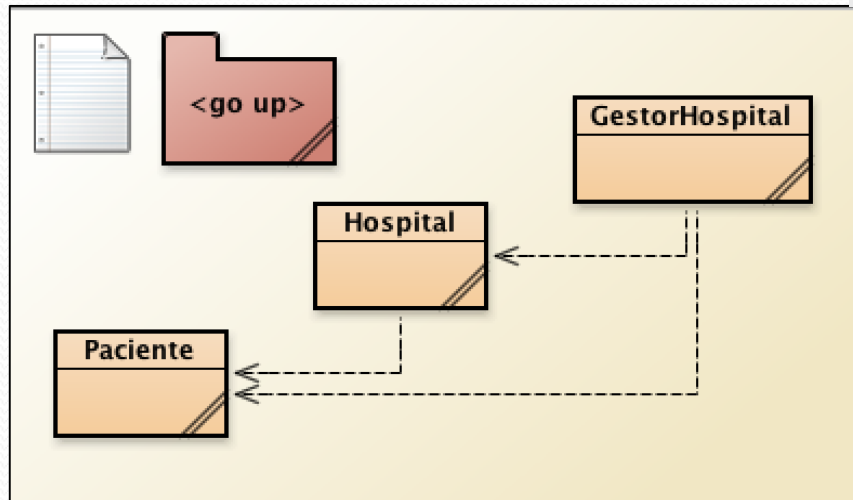
### PARTE 1:

- A. ¿Necesitamos un array para representar los datos? (tipo y tamaño)
- B. ¿Qué operaciones básicas se realizan sobre las componentes del array?

### PARTE 2: implementación -BÁSICA- de un Hospital en Java



BlueJ: ejercicios – Tema 7



**Abre** el *package* `eHospital` del proyecto; encontrarás las clases del problema pero sin completar, excepto `Paciente`

- **Crea** un objeto de la clase `Paciente`, **inspecciónalo** y **prueba** sus métodos
- **Edita** la clase `Hospital`, **define** sus atributos y todos los métodos **hasta ingresar incluido y exceptuando primeraLibre**