Sesión 5:
Arrays

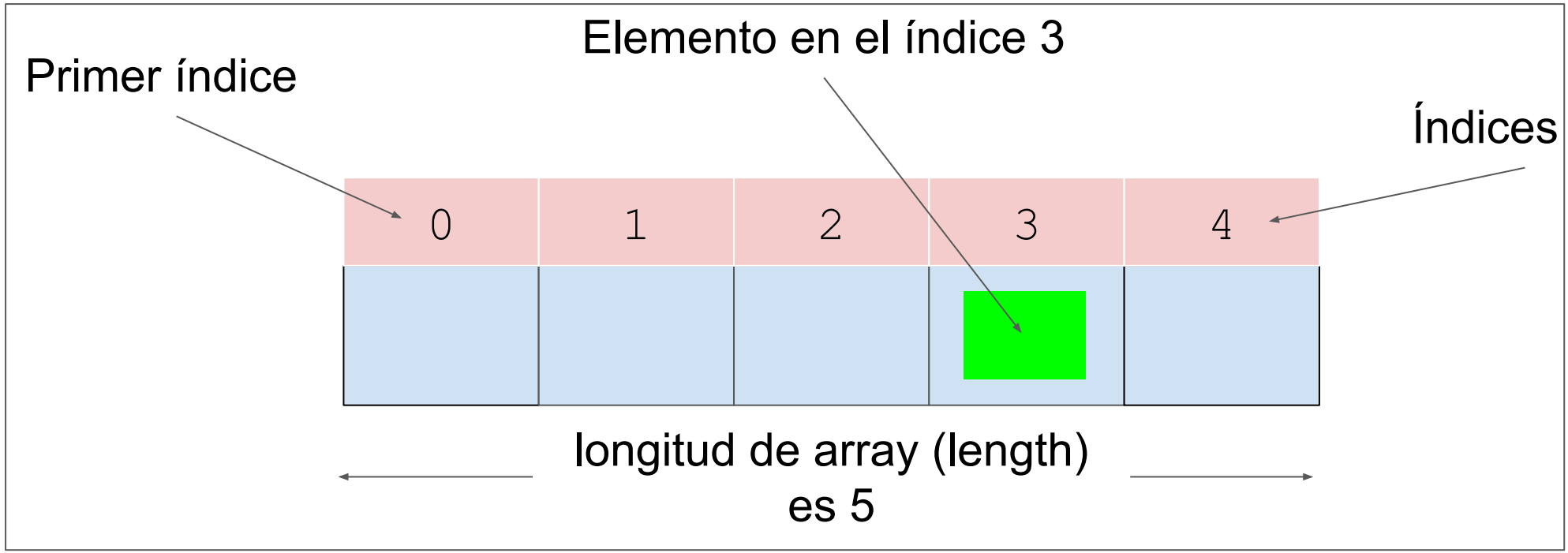
Índice

Arrays

- Un **array** es una *estructura de datos* que nos permite almacenar una serie de datos o elementos.
- Simplemente es una **variable que puede contener múltiples valores.**
- Estos valores tienen que ser del mismo **tipo (int, char, byte…)**
- Como todas las variables, se hace referencia al array por medio de un **nombre común**, y se accede a un valor concreto a través de un **índice**.
- Los arrays pueden tener una **longitud fija igual o mayor a cero**.

Arrays

- Cada ítem del array se conoce como *elemento*
- Se accede al valor de cada elemento a través de un *índice numérico*.
- Los índices de un array comienzan en 0
- Java soporta arrays unidimensionales o multidimensionales.



Declarar un array

- La sintaxis para declarar un array es:

  tipo nombre_array[]*;*

  - *tipo* indica el tipo de elementos que va a almacenar el array.
  - *nombre_array* es el nombre del array

  Otra forma: *tipo[] nombre_array;*

- Por ejemplo, un array de tipo int almacenará una lista de valores enteros:

  int dias_del_mes[];

  - *Este array tiene valor null y no se ha especificado su tamaño*

- Un array de caracteres no es un String..

Inicializar array

- Para inicializar un array real de enteros, se utiliza el operador **new** y se asigna un tamaño de elementos al array.
- El operador **new** es un operador especial para reservar espacio de memoria. En el caso de los arrays se usa para reservar espacio en memoria.

- Se utiliza así:

nombre_array = new *tipo* [*tamaño*] ⟶ `dias_del_mes = new int[12];`

- En el ejemplo, se reserva espacio para un array de 12 elementos enteros y los asigna a la variable *dias_del_mes*.
- Todos los elementos se inicializan a 0. Si fueran objetos serían null.

- Con la otra sintaxis: `int[] dias_del_mes = new int[3];`

Inicializar array

- Otra forma de hacerlo, indicando el valor inicial en cada posición:

```
int unArray[] = new int[] {1, 2, 3, 4, 5};
```

No es necesario indicar el tamaño, pues es implícito.

- Podemos obviar la declaración de su tipo si lo estamos inicializando:

```
int array[] = {1, 2, 3, 4, 5};
```

- La Clase *Java.utils.array* contiene otros métodos que nos ayudan a ello:

  fill, copyof, setAll…

https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/util/Arrays.html

Acceder a un elemento del array

- Se puede acceder a un elemento del array usando su índice entre corchetes:

```
dias_del_mes[1];
```

Por ejemplo:

```
int dias_del_mes = new int[12];
System.out.println(dias_del_mes[3]);
```

- En nuestro ejemplo, se crea un array de 12 elementos, el índice del primer elemento será el 0 y el índice del último elemento será 11.

  Se accede al cuarto elemento del array usando el índice 3 entre corchetes.

- Cabe destacar que **los índices de un array empiezan en 0.**

- **No se deben usar índices negativos, ni mayores o iguales a su tamaño.**

Añadir valores a un array

Se puede dar valor a los elementos de un array de dos formas:

- Accediendo a un elemento del array y dándole valor con el operador '=':

```
int dias_del_mes = new int [12];

dias_del_mes[0] = 31;
```

- Asignando los valores de cada elemento entre llaves y separados por coma. En este caso, no será necesario utilizar el operador **new**.

```
int dias_del_mes[] = {31, 28, 31, 30, 31, 30, 31, 31,
30, 31, 30, 31};
```

Ejemplo de array

- El siguiente programa crea un array, reserva espacio para 12 elementos, y asigna un valor a cada elemento. Al final, se imprime el número de días del mes Abril.

```
int dias_del_mes[];
dias_del_mes = new int[12];
dias_del_mes[0] = 31;
dias_del_mes[1] = 28;
dias_del_mes[2] = 31;
dias_del_mes[3] = 30;
dias_del_mes[4] = 31;
dias_del_mes[5] = 30;
dias_del_mes[6] = 31;
dias_del_mes[7] = 31;
dias_del_mes[8] = 30;
dias_del_mes[9] = 31;
dias_del_mes[10] = 30;
dias_del_mes[11] = 31;
System.out.println("Abril tiene " + dias_del_mes[3] + " dias.");
```

Más info sobre arrays

- **Rango del array:**
  - Cuando se crea un array de un tamaño, los índices deben estar dentro de ese rango. Java comprueba estrictamente que los índices del array están dentro del rango correcto en tiempo de ejecución.

    En nuestro ejemplo, comprobará el valor de cada índice de *dias_del_mes* para asegurarse de que están comprendidos entre 0 y 11 y que no se usa incorrectamente.
  - Si se usa un índice fuera del rango, se producirá un error en tiempo de ejecución: **IndexOutOfBoundsException**

    En nuestro ejemplo, se producirá este error si se escribe:

```
System.out.println(dias_del_mes[12]);
```

Más info sobre arrays

- **Tamaño del array: .length**

  length es un atributo de los arrays que devuelve el número de elementos que tiene un array. Es muy útil, si queremos conocer el tamaño del array, por ejemplo, en bucles. Se utiliza:

```
int tamaño = a2.length;   // devolvería 3
```

- **Ordenar un array:**

  Método **sort** en java.util.Arrays.

  Hay varias versiones. Los de tipos primitivos se ordenan ascendentemente.

- **Buscar en un array**

  Tendríamos que recorrer el array comparando con cada elemento. Si el array está ordenado, podemos hacer una búsqueda binaria.

Iterar sobre un array

Se puede iterar sobre los elementos de un array usando un bucle *for* o *while*.

```
    for (int i = 0; i < dias_del_mes.length; i++) {
        System.out.println("El mes " + (i + 1) + " tiene " +
        dias_del_mes[i] + " días.");
    }

/*****************************************************************/

    int j = 0;
    while (j < dias_del_mes.length) {
        System.out.println("El mes " + (j + 1) + " tiene " +
        dias_del_mes[j] + " días.");
      j++;
    }
```

Tener en cuenta que el contador del bucle siempre tiene que ser menor que el tamaño del array. **Los índices comienzan en 0 y terminan en length-1** *(tamaño menos 1).*

Iterar sobre un array

El bucle for tiene una sintaxis alternativa para recorrer arrays, el bucle *forEach:*

```
int[] anArray = new int[] {1, 2, 3, 4, 5};
for (int element : anArray) {
    System.out.println(element);
}
```

Así nos ahorramos todo lo relativo a índices, es una buena opción en estos casos:

1. No necesitamos modificar el array (cambiar el valor de un elemento no tendrá efecto)

2. No necesitamos hacer nada que involucre usar índices.

Este bucle recorre todos los elementos del array.