3. Creación de arrays

Tal como hemos comentado, un array es una colección de elementos de un mismo tipo. Por tanto, el primer paso a la hora de crear un array es definir de qué tipo es, al tiempo que le damos un identificador para poder referirnos a él posteriormente. Por ejemplo:

```
int[] arrayEnteros;
```

Esto define un array llamado arrayEnteros (de momento, sin indicar su tamaño) preparado para almacenar una colección de datos de tipo entero. El array, sin embargo, todavía no existe. Debemos crearlo definiendo, ahora sí, su tamaño:

```
arrayEnteros = new int[100];
```

Acabamos de crear el array arrayEnteros, con capacidad para almacenar hasta 100 números enteros. Esa capacidad, definida por la cantidad que aparece entre corchetes, es fija desde el momento de la creación del array, es decir:

- No la podemos superar, luego no caben más de 100 números en el array.
- Las 100 posiciones, aunque de momento estén vacías —en Java, al crear un Array se inicializan todas sus posiciones a cero—, ya están reservadas desde el principio.

La declaración y la creación del array pueden hacerse a la vez mediante una única línea de código:

```
int[] arrayEnteros = new int[100];
```

La principal ventaja de usar nuestro array de 100 posiciones frente a emplear 100 variables distintas se refleja en dos aspectos fundamentales:

- El array tiene un nombre único, mientras que las 100 variables tendrían nombres diferentes.
- Cada posición del array viene dada por un número correlativo, que empieza en 0 y, en este caso, acabaría en 99.

En este ejemplo, tras crear el array se han insertado estos valores:

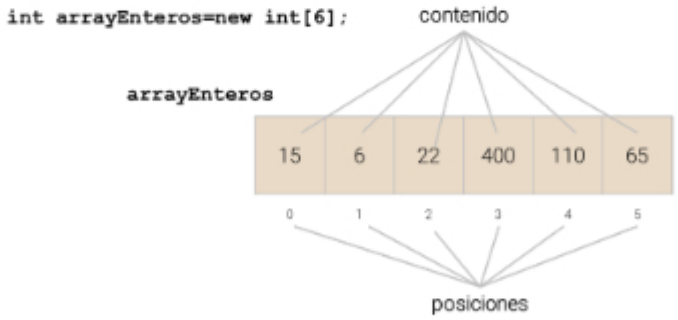

Figura 4.1. Array de 6 posiciones con su contenido y sus posiciones

IMPORTANTE

Dado que la primera posición de un array es la 0, la última siempre es el número de elementos que contiene menos uno. Es decir, en un array de 10 posiciones, la primera sería la 0 y la última la 9. Si tuviese 15 posiciones, la primera seguiría siendo la 0 y la última sería la 14. En definitiva, en un array con ***n*** **posiciones** la primera es la 0 y **la última es la** ***n***–1.

Para hacer referencia al contenido de una posición, pondremos el nombre del array y, entre corchetes, la posición deseada. Por ejemplo, si tomamos el array de la figura anterior y hacemos:

```
System.out.println("El elemento de la posición 4 es " + arrayEnteros[4]);
```

al poner arrayEnteros[4] estamos haciendo referencia al contenido de la posición 4 del array arrayEnteros. Por tanto, la respuesta por pantalla será 110.

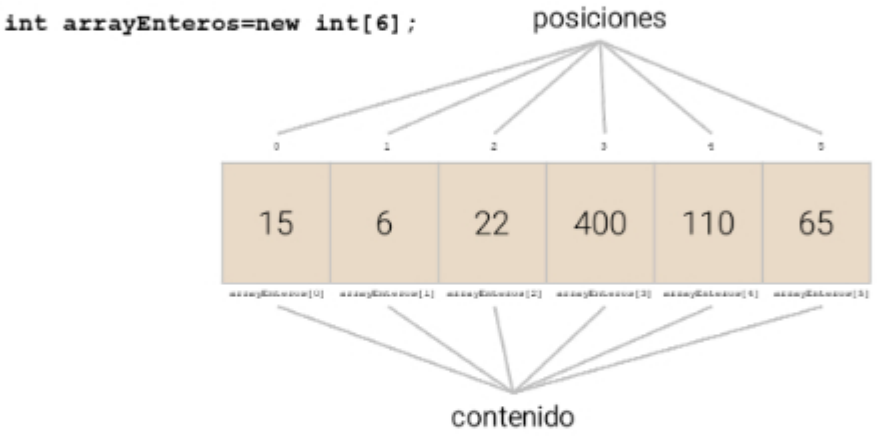

Figura 4.2. Array de 6 posiciones con referencias a cada posición, así como a su contenido

Supongamos que queremos mostrar el contenido de la posición 2, modificarlo, y después volverlo a mostrar:

```
System.out.println("Contenido de la posición 2: " + arrayEnteros[2]); // muestra 22
arrayEnteros[2] = 23; // le estamos diciendo que guarde un 23 en la posición 2 del array
System.out.println("Contenido de la posición 2: " + arrayEnteros[2]); // muestra 23
```

Lógicamente, en un array con muchas posiciones, resultaría largo y pesado rellenarlas de una en una. Por suerte, al existir un índice correlativo que empieza desde 0 para referenciar cada posición, es posible (y además recomendable) usar bucles para trabajar con los arrays. Para ello, usaremos una variable que haga de índice (pongamos que sea i), y haremos un bucle que recorra el array desde el principio (0) hasta el final (*n*–1, donde *n* es el tamaño del array):

```
for(int indice = 0; indice < n; indice++) // desde la posición 0 hasta la n-1 (indice < n)
```

Dentro del bucle, ya sabemos que el índice hará referencia a una posición. Además, suponiendo que el array se llame arrayEnteros, entonces arrayEnteros[indice] hace referencia al contenido de esa

posición. Así pues, mostrar el contenido de un array es muy sencillo con ayuda de un bucle. Recordemos que *n* representa el tamaño de un array llamado arrayEnteros:

```
for(int indice=0; indice<n; indice++) {
   System.out.println("El contenido de la posición " + indice + " es " +
   arrayEnteros[indice]);
}
```

Asignándole a *n* el valor del tamaño del array, el bucle nos mostrará el contenido de todas las posiciones de un array, sea cual sea su tamaño.

Por otro lado, para inicializar un array (es decir, para establecer sus valores iniciales) tenemos varias opciones. Si hablamos de un array de pocas posiciones cuyo contenido sabemos de antemano, podemos rellenarlo al tiempo que lo declaramos.

```
int[] arrayEnteros = {15, 6, 22, 400, 110, 65};
```

Lo más habitual, sin embargo, será que los datos del array deban rellenarse desde una fuente externa. En ese caso, deberemos recurrir de nuevo a un bucle:

```
int[] arrayEnteros = new int[6];
for(int indice = 0; indice < 6; indice++) {
      arrayEnteros[indice] = valor;  // ponemos el valor que queremos guardar
}
```

De esa forma, sin embargo, solo podemos rellenar un array en el que todas sus posiciones tengan el mismo contenido, salvo que la variable valor sea el resultado de alguna fórmula, de haber leído un dato de otra fuente, o de haber pedido al usuario que introduzca un valor.

```
Scanner teclado = new Scanner(System.in);
int[] arrayEnteros = new int[6];
for(int indice = 0; indice < 6; indice++) {
      System.out.print("Introduzca el valor para la posición " + indice + ": ");
      arrayEnteros[indice] = teclado.nextInt();
}
```

Si no sabemos cuántas posiciones tiene un array, o si queremos que el bucle funcione igualmente aunque en programación se cambie su tamaño, podemos utilizar la propiedad `length`, que nos devuelve el tamaño del array.

```
for(int indice = 0; indice < arrayEnteros.length; indice++)
```

De esa forma, sea cual sea el tamaño de arrayEnteros, no tendremos que cambiar nuestro bucle.

IMPORTANTE

Recuerda que, en un bucle del tipo for(int i = 0; i < a.length; i++), la **variable** i hace referencia a la **posición** del array en la que nos encontramos (0, 1, 2…), mientras que **a[i]**, donde *a* es el nombre del array, haría referencia al **contenido** de dicha posición. Teniendo esto siempre presente, te será más sencillo resolver algunos de los ejercicios propuestos.