Arrays unidimensionales en Java

ÍNDICE

1. Introducción

Un array es una estructura de datos que nos permite almacenar un conjunto de datos de un **mismo tipo**. En **Java**, el tamaño de los arrays se declara en un primer momento (en tiempo de compilación) y no puede cambiarse en tiempo de ejecución.

Por ejemplo, si quiero almacenar los nombres de todos los alumnos de la clase, puedo crear una variable de tipo cadena de texto por cada uno de los alumnos del grupo. Con el uso de arrays, con una única variable puedo almacenar los nombres de todos, cada uno en una posición del array.

La primera posición del array será la 0 y así sucesivamente. Por tanto, la última posición del array será la correspondiente al tamaño-1. Si tenemos un array de diez elementos, el primer elemento sería el cero y el último elemento sería el nueve.

Durante el presente tema vamos a aprender a implementar el manejo de **arrays unidimensionales**.

2. Arrays unidimensionales o vectores

Son aquellos que tienen una única dimensión, es decir, para acceder a una posición de memoria únicamente necesito un subíndice.

2.1 Declaración

Para declararlos lo podemos hacer de la siguiente forma:

```
TipoDato[] nombreArray;
```

Por ejemplo, para declarar un array que va a almacenar el nombre de los alumnos podemos hacerlo de la siguiente forma:

```
String[] alumnos;
```

2.2 Inicialización

Para poder comenzar a utilizarlos, debemos inicializarlos. Lo podemos hacer de la siguiente forma:

```
nombreArray = new TipoDato[tamaño];
```

Para nuestro ejemplo de los alumnos, si deseamos inicializar un array para almacenar el nombre de 4 alumnos, lo podemos hacer así:

```
alumnos = new String[4];
```

Podemos combinar la declaración y la inicialización de la siguiente forma:

```
String[] alumnos = new String[4];
```

En Java, tenemos una forma especial de inicializar y asignar valores directamente. Es la siguiente:

```
String[] alumnos = {"Ana", "Miguel", "Juan", "Cristina"};
```

2.3 Asignación de valores

Hemos visto que el primer elemento del array ocupa la posición 0, el segundo ocupa la posición 1 y así, sucesivamente, hasta la posición "tamaño-1" que será la última.

```
nombreArray[posición] = valor
```

Por ejemplo, si la alumna número 4 (última posición del array, es decir, la 3) se llama Cristina, puedo asignar dicho valor de la siguiente forma haciendo uso de los corchetes

```
alumnos[3] = "Cristina";
```

2.4 Acceso

Para acceder a un elemento específico utilizaremos los corchetes de la misma forma que lo hacíamos para la asignación.

```
variable <- nombreArray[posición]
```

Para nuestro ejemplo de los alumnos sería:

```
String nombre = alumnos[3];
```

2.5 Resumen de los pasos

De esta manera, un código completo podría ser el siguiente:

```
// Forma 1
String[] alumnos = new String[4]; // Declaración + inicialización
alumnos[3] = "Cristina"; // Asignación de valor
String nombrePrimerAlumno = alumnos[0]; // Lectura de valor

// Forma 2
String[] alumnos = {"Ana", "Miguel", "Juan", "Cristina"};
```

3. Recorrer arrays unidimensionales

Para recorrer arrays es importante saber su longitud. Todos los lenguajes de programación permiten obtener el tamaño de las distintas dimensiones de un array. En el caso de Java debemos hacer uso de la propiedad **`length`** del array:

```
nombreArray.length
```

Una vez que sabemos la longitud (tamaño) de un array podemos utilizar un `bucle n` o un bucle `foreach` para recorrer todas sus posiciones de forma secuencial.

3.1 Bucle for

Para poder recorrer un array unidimensional podemos hacer uso de la siguiente estructura de código:

```
String[] alumnos = {"Ana", "Miguel", "Juan", "Cristina"};

for(int i=0; i < alumnos.length; i++) {
   System.out.println((i+1) + ". " + alumnos[i]);
}
```

Con dicha estructura tenemos acceso tanto a los valores del propio array como al índice en el que se encuentra cada uno de los valores.

3.2 Bucle foreach

Para poder recorrer un array también podemos hacer uso de foreach. Un ejemplo de uso es el siguiente:

```
String[] alumnos = {"Ana", "Miguel", "Juan", "Cristina"};

for(String nombreAlumno : alumnos) {
   System.out.println(nombreAlumno);
}
```

En este caso únicamente tenemos acceso a los valores de cada posición del array pero no a la posición o índice en el que se encuentra cada valor.

4. Parámetro del main

Si nos fijamos en la firma o interfaz del método `main` podemos ver que el parámetro `args` es un array de `String`:

```
public static void main(String[] args) {
	// …
}
```

Con el conocimiento que ya tenemos sobre programación y sobre Java podemos darnos cuenta que el `main` es un método que recibe un único parámetro, con lo cuál… ¿podríamos pasarle valores en el array y que los procese? La respuesta es sí.

Para saber cómo implementar esta funcionalidad y poder ejecutar el programa desde un terminal (siempre y cuando tengamos bien configurado el `PATH`) podemos partir del siguiente código fuente que recibe dos números por parámetro y nos devuelve la suma de ambos.

```
public class MainArgsApp {

	public static void main(String[] args) {
		int numero1 = Integer.parseInt(args[0]);
		int numero2 = Integer.parseInt(args[1]);

		System.out.printf("%d + %d = %d", numero1, numero2, numero1 + numero2);
	}
}
```

Para ejecutar el programa, deberemos pasarle los dos números como argumento. Antes de nada habrá que compilar el código y, después, ejecutarlo desde el terminal de la siguiente forma:

```
prompt> java MainArgsApp 7 4
```

La salida por pantalla será la siguiente:

```
7 + 4 = 11
```

5. Actividad de investigación

Investiga las siguientes cuestiones:

- ¿Cómo podemos ordenar un array?
- ¿Cómo podemos hacer una copia exacta de un array?
- ¿Qué significa el paso por valor y por referencia de los parámetros?