1. ARRAYS

Un array es un objeto en el que se puede almacenar un conjunto de datos de un mismo tipo. Cada uno de los elementos del array tiene asignado un índice numérico según su posición, siendo 0 el índice del primero.

Declaración

Un array debe declararse utilizando la expresión:

```
tipo[] nombre_array;   o   tipo nombre_array[];
```

Como se puede apreciar, los corchetes pueden estar situados delante de la variable o detrás. Los siguientes son ejemplos de declaraciones de array:

```
int[] numeros;
String[] nombres;
char caracteres[];
```

Los arrays pueden declararse en los mismos lugares que las variables estándar: como **atributos** de una clase o **locales** en el interior de un método. Como ocurre con cualquier otro tipo de variable objeto, cuando un array se declara como atributo se inicializa implícitamente al valor *null*.

Dimensionado de un array

Para asignar un tamaño al array se utiliza la expresión:

```
nombre_array = new tipo[tamaño];
```

También se puede asignar tamaño al array en la misma línea de declaración de la variable. Los siguientes son ejemplos de dimensionado de un array:

```
numeros = new int [5];
caracteres = new char[10];
String[] nombres = new String[10];
```

Cuando un array se dimensiona, **todos sus elementos son inicializados explícitamente** al valor por defecto del tipo correspondiente, independientemente de que la variable que contiene al array sea atributo o local.

Existe una forma de declarar, dimensionar e inicializar un array en una misma sentencia. La siguiente instrucción crea un array de cuatro enteros y los inicializa a los valores indicados entre llaves:

```
int[] numeros = {10, 20, 30, 40};
```

Acceso a los elementos de un array

El acceso a los elementos de un array se realiza utilizando la expresión:

```
nombre_array[índice]
```

Donde **índice** representa la posición a la que se quiere tener acceso, y cuyo valor debe estar comprendido entre 0 y *tamaño - 1*.

Todos los objetos array exponen un atributo público, llamado **length**, que permite conocer el tamaño al que ha sido dimensionado un array. Este atributo resulta especialmente útil en aquellos métodos que necesitan recorrer todos los elementos de un array, independientemente de su tamaño.

El siguiente bloque de código utiliza **length** para recorrer un array y rellenarlo con números enteros pares consecutivos, empezando por el 0.

```
int[] numeros = new int [10] ;
for (int i = 0; i < nums.length; i++) {
      numeros[i] = i * 2;
}
```

El programa que se muestra a continuación calcula la suma de todos los números almacenados en un array de enteros, mostrando además el mayor y el menor de los números contenidos:

```
public class Principal
{
   public static void main(String[] args) {
      int mayor, menor, suma;
      int[] numeros = {3,4,8,2};

      suma = 0 ;
      menor = numeros[0];
      mayor = numeros[0];

      // En las variables mayor y menor se obtendrá
      // el mayor y el menor de los números buscados
      // respectivamente. Ambas se inicializan con
      // cualquiera de los valores del array.
      // Luego se recorre el array en busca de los
      // extremos, acumulando en suma cada uno de los
      // números contenidos en el array.
      for (int i = 0; I < numeros.length; i++) {
         if (numeros[i] > mayor) {
            mayor =numeros[i];
         }
         if (numeros[i] < menor) {
            menor = numeros[i];
         }
         suma += numeros[i];
      }
      System.out.println("El mayor es: " + mayor);
      System.out.println("El menor es: " + menor);
      System.out.println("La suma es: " + suma);
   }
}
```

Recorrido de arrays *con for-each*

A partir de la versión Java 5, el lenguaje Java incorpora una variante de la instrucción *for*, conocida como **`for-each`**, que facilita el recorrido de arrays y colecciones para la recuperación de su contenido, eliminando la necesidad de utilizar una variable de control que sirva de índice para acceder a las distintas posiciones. Aunque se utiliza un nuevo nombre para identificar esta instrucción *(***`for-each`***)*, la palabra reservada para su implementación sigue siendo **`for`**.

En el caso de un array, el formato de la nueva instrucción for-each sería:

```
for (tipo nombre_variable : nombre_array)
      { //instrucciones
}
```

donde, **`tipo nombre_variable`** representa la declaración de una variable auxiliar del mismo tipo que el array, variable que irá tomando cada uno de los valores almacenados en éste con cada iteración del **`for`**, siendo **`nombre_array`** la variable que apunta al array.

Por ejemplo, supongamos que tenemos el siguiente array:

```
int[] numeros = {4, 6, 30, 15};
```

Y queremos mostrar en pantalla cada uno de los números almacenados en el mismo. Utilizando la versión tradicional de **`for`**, la forma de hacerlo sería:

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

Utilizando la nueva instrucción **`for-each`**, la misma operación se realizará de la siguiente forma:

```
for (int numero : numeros) {
      System.out.println(numero);
 }
```

Obsérvese cómo, sin acceder de forma explícita a las posiciones del array, cada una de éstas es **copiada automáticamente a la variable auxiliar n al principio de cada iteración.**

Como ejemplo ilustrativo, a continuación se presenta una versión del bucle del programa anterior que realiza la suma de los elementos de un array. En este caso se han utilizado bucles **`for-each`** sin índices en vez de **`for`** tradicionales:

```
for (int numero : numeros) {
   if (numero > mayor) {
      mayor = numero;
   }
   if (numero < menor) {
      menor = numero;
   }
   suma += numero;
}
```

Recordemos que el código anterior solamente funcionará con la versión 5 de Java SE o posteriores.

2. ARRAYS MULTIDIMENSIONALES

Los arrays Java pueden tener más de una dimensión, por ejemplo, un array de enteros de dos dimensiones se declararía:

```
int[][] enteros;
```

A la hora de asignarle tamaño se procedería como en los de una dimensión, indicando en los corchetes el tamaño de cada dimensión:

```
enteros = new int[3][5];
```

Igualmente, para acceder a cada una de las posiciones del array se utilizaría un índice por dimensión:

```
enteros[1][3] = 28;
```

Si imaginamos un array de dos dimensiones como una tabla organizada en filas y columnas, el array anterior tendría el aspecto indicado siguiente.

Array **enteros**:

	0	1	2	3	4
0	0	0	0	0	0
1	0	0	0	28	0
2	0	0	0	0	0

Array bidimensional

Recorrido de un array multidimensional

Para recorrer un array multidimensional podemos utilizar la instrucción *for* tradicional, empleando para ello tantas variables de control como dimensiones tenga el array. La siguiente instrucción almacenaría en cada una de las posiciones del array **enteros**, definido anteriormente, la suma de sus índices:

```
// Recorre primera dimensión
for (int i = 0; i < enteros.length; i++) {
    // Recorre segunda dimensión
    for (int j = 0; j < enteros[i].length; j++) {
        enteros[i][j] = i+j;
    }
}
```

Igualmente, se puede utilizar también una instrucción ***for-each*** para recuperar el contenido de un array multidimensional, simplificando enormemente dicha tarea. El siguiente ejemplo mostraría en pantalla los valores almacenados en el array **enteros** anterior:

```
// Recorre primera dimensión
for (int[] fila : enteros) {
    // Recorre segunda dimensión
    for (int elemento : fila) {
        System.out.println("valor: " + elemento);
    }
}
```

Arrays multidimensionales irregulares

Es posible definir un array multidimensional en el que el número de elementos de la segunda y sucesivas dimensiones sea variable, indicando inicialmente sólo el tamaño de la primera dimensión:

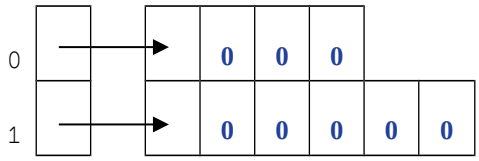
```
int[][] irregular = new int[2][];
```

Un array de estas características equivale a un array de arrays, donde cada elemento de la primera dimensión almacenará su propio array:

```
irregular[0] = new int[4];
irregular[1] = new int[6];
```

La figura siguiente ilustra gráficamente la situación.

Array **irregular**:



Array bidimensional con dimensiones de diferente tamaño

Un array de estas características se puede recorrer empleando cualquier tipo de bucle, aunque los más cómodos son **for** y **for-each**. Es importante tener en cuenta que la segunda dimensión puede ser distinta para cada elemento (array) de la primera dimensión.

Los ejemplos que se muestran a continuación, recorrerían el array anterior con bucles **for** y **for-each** respectivamente:

```
// Recorre primera dimensión
for (int i = 0; i < irregular.length; i++) {
   // Recorre segunda dimensión
   for (int j = 0; j < irregular[i].length; j++) {
      irregular[i][j] = i+j;
   }
}
```

```
// Recorre primera dimensión
for (int[] item : irregular) {
   // Recorre segunda dimensión
   for (int elemento : item) {
      System.out.println("valor: " + elemento);
   }
}
```