ARRAYS (ARREGLOS) MULTIDIMENSIONALES

Vamos a realizar un repaso sobre conocimientos que debemos tener relativos a arrays multidimensionales. En Java es posible crear arrays con más de una dimensión, pasando de la idea de lista, vector o matriz de una sola fila a la idea de matriz de m x n elementos, estructuras tridimensionales, tetradimensionales, etc. La sintaxis será:

```
Tipo_de_variable[ ][ ]… [ ] Nombre_del_array = new  Tipo_de_variable[dimensión1][dimensión2]…[dimensiónN];
```

También podemos alternativamente usar esta declaración:

```
Tipo_de_variable[ ][ ] … [ ]   Nombre_del_array;

Nombre_del_array = new  tipo_de_variable[dimensión1][dimensión2]…[dimensiónN];
```

El tipo de variable puede ser cualquiera de los admitidos por Java y que ya ha sido explicado. Ejemplos de declaración e inicialización con valores por defecto de arrays, usando los distintos tipos de variables Java, serían:

```
byte[][] edad = new byte[4][3];
short ][] edad = new short[4][3];
int[][] edad = new int[4][3];
long[][] edad = new long[4][3];
float[][] estatura = new float[3][2];
double[][] estatura = new double[3][2];
boolean[][] estado = new boolean[5][4];
char[][] sexo = new char[2][1];
String[][] nombre = new String[2][1];
```

La declaración de una matriz tradicional de m x n elementos podría ser:

```
int[][]  matriz = new int[3][2];
      O alternativamente
int[][]  matriz;
matriz = new int[3][2];
```

El número de elementos sería: 3 x 2 = 6, dónde 3 es el número de filas y 2 es el número de columnas.

Ahora procedemos a cargar la matriz con valores:

matriz[0][0] = 1; matriz[0][1] = 2; matriz[1][0] = 3; matriz[1][1] = 4; matriz[2][0] = 5; matriz[2][1] = 6;

Hay que recordar que los elementos empiezan a numerarse por 0. Así, la esquina superior izquierda de la matriz será el elemento [0][0] y la esquina inferior derecha será el [2][1]. Hay que prestar atención a esto porque en otros lenguajes de programación la numeración puede empezar por 1 en vez de por 0.

También se pueden cargar directamente los elementos, durante la declaración de la matriz de la siguiente manera:

```
int[][]  matriz = {{1,2},{3,4},{5,6}};
```

donde {1,2} corresponde a la fila 1, {3,4} a la fila 2 y {5,6} a la fila 3, y los números separados por coma dentro de cada fila, corresponden a las columnas. En este caso, los números (1, 3, 5) de cada una de las filas corresponden a la primera columna y los números (2, 4, 6) atañen a la segunda columna.

Para obtener el número de filas de la matriz, podemos recurrir a la propiedad "length" de los arrays, de la siguiente manera:

```
int filas = matriz.length;
```

Para el caso del número de columnas sería de la siguiente forma:

```
int columnas = matriz[0].length;
```

También Java nos permite la posibilidad de clonar una matriz, es decir, crear una matriz nueva a partir de otra matriz, siguiendo esta sintaxis:

```
String[][] nuevaMatriz = matriz.clone();
```

donde clone() es un método especial, que permite la clonación de arrays de cualquier dimensión en Java. De esta manera "nuevaMatriz" y "matriz" son 2

matrices distintas pero con los mismos valores. Hablaremos del método clone más adelante.

EJERCICIO RESUELTO

Vamos a plantear y resolver un ejercicio: queremos almacenar en una matriz el número de alumnos con el que cuenta una academia, ordenados en función del nivel y del idioma que se estudia. Tendremos 3 filas que representarán al Nivel básico, medio y de perfeccionamiento y 4 columnas en las que figurarán los idiomas (0 = Inglés, 1 = Francés, 2 = Alemán y 3 = Ruso). Se pide realizar la declaración de la matriz y asignarle unos valores de ejemplo a cada elemento.

SOLUCIÓN

La declaración de la matriz sería:

```
int[][]  alumnosfxniveleidioma= new int[3][4];
```

Podríamos asignar contenidos de la siguiente manera:

alumnosfxniveleidioma[0][0] = 7; alumnosfxniveleidioma[0][1] = 14;

alumnosfxniveleidioma[0][2]= 8; alumnosfxniveleidioma[0][3] = 3;

alumnosfxniveleidioma[1][0] = 6; alumnosfxniveleidioma[1][1]= 19;

alumnosfxniveleidioma[1][2] = 7; alumnosfxniveleidioma[1][3] = 2

alumnosfxniveleidioma[2][0] = 3; alumnosfxniveleidioma[2][1]= 13;

alumnosfxniveleidioma[2][2] = 4; alumnosfxniveleidioma[2][3] = 1

También, podríamos asignar contenido de esta otra forma, como ya se ha explicado anteriormente:

int[][] alumnosfxniveleidioma = {{7,14,8,3},{6,19,7,2},{3,13,4,1}};

La representación gráfica que podríamos asociar a esta asignación de datos sería esta matriz:

$$\begin{pmatrix} 7 & 14 & 8 & 3 \\ 6 & 19 & 7 & 2 \\ 3 & 13 & 4 & 1 \end{pmatrix}$$

La organización de la información en matrices, nos generará importantes ventajas a la hora del tratamiento de datos en nuestros programas.

Para terminar en cuanto a multidimensionalidad, veamos casos de declaraciones con más de dos dimensiones. Para ello supongamos que estamos realizando un "conteo de coches", es decir, que estamos contando los coches que pasan por un determinado lugar en un periodo de tiempo que puede ser un día, varios días, varios meses, etc. La forma de declarar esos arrays podría ser la siguiente:

Duración del conteo	**Tipo de array**	**Declaración con Java** **(nc es Número de coches)**
Un día	Array de una dimensión (hora)	int[] nc = new int[24];
Varios días	Array de dos dimensiones (hora y día)	int[][] nc = new int[24][31];
Varios meses	Array de tres dimensiones (hora, día y mes)	int[][][] nc = new int[24][31][12];
Varios años	Array de cuatro dimensiones (hora, día, mes y año)	Int[][][][] nc = new int[24][31][12][2999];

Varios siglos	Array de cinco dimensiones (hora, día, mes, año y siglo)	Int[][][][][] nc = new int[24][31][12][2999][21];

Veamos lo que sería un ejemplo de programa con array multidimensional, usando un tipo String.

```
public class MatrizAlumnos {

 public static void main(String arg[]) {

      String[ ][ ]  nombreAlumno = new String[5][25];

      nombreAlumno[2][23] = "Pedro Hernández González";

      System.out.println("El alumno número 24 del curso tercero se llama
      "+nombreAlumno[2][23]);

 }

}
```

El resultado del programa es la aparición del mensaje "El alumno número 24 del curso tercero se llama Pedro Hernández González.

En este ejemplo, [5] representa a los cursos. Hablamos de 5 cursos que son identificados con 0, 1, 2, 3, 4, por lo que [2] hace mención al tercer curso;lo mismo podemos decir de [23], que corresponde al alumno número 24. Hay que recordar que siempre en Java tenemos que contar el cero, ya que si no lo hacemos podemos cometer errores.

Trabajo Práctico:

EJERCICIO 1

Crea un programa que pida por pantalla cuatro países y a continuación tres ciudades de cada uno de estos países. Los nombres de ciudades deben almacenarse en un array multidimensional cuyo primer índice sea el número asignado a cada país y el segundo índice el número asignado a cada ciudad. Es decir el array deberá tener un tamaño de 4x4

Ejemplo de resultados que debe mostrar el programa:

País: Argentina	Ciudades:	Buenos Aires	Cordoba	La Plata
País: España	Ciudades:	Madrid	Lugo	Sevilla
País: Francia	Ciudades:	Paris	Niza	Lyon
País: Italia	Ciudades:	Roma	Napoles	Sicilia

EJERCICIO 2

Crea un programa que pida por pantalla 2 valores numéricos enteros X e Y. Cree un primer array de tamaño [X,Y] de tipo entero y almacene en cada posición un valor entero que deberá ser ingresado por el usuario. A continuación cree un segundo array de tamaño [Y,X] y almacene en cada posición un valor entero que deberá ser ingresado por el usuario. Finalmente cree un tercer array de tamaño [X,Y] que será el resultante de multiplicar cada una de las posiciones de las filas del array uno por cada una de las posiciones de las columnas del array 2.

Muestre por pantalla cada uno de los arrays con sus valores.

EJERCICIO 3

Codifique un programa que solicite un valor entero X mayor o igual a 3 y menor o igual a 10. Cree un arreglo de tamaño [X,X] de tipo int. Posteriormente solicite los

valores necesarios para cargar cada una de las celdas de la matriz. Muestre por pantalla la matriz resultante.

Al finalizar la carga sume cada una de las columnas del array y asigne los resultados en una nueva matriz de una dimensión, finalmente sume los valores de esta última matriz y muestre el resultado por pantalla.

Ejemplo: X=5

3	5	8	88	7
45	34	67	87	54
34	43	23	44	55
6	0	45	4	66
56	44	32	12	54

144	126	175	235	236

Total: 916

EJERCICIO 4

Codifique un programa que solicite un valor entero X mayor o igual a 4 y menor o igual a 10. Cree un arreglo de tamaño [X,X] de tipo int, nos piden hacer un menú con estas opciones:

a) Rellenar TODA la matriz de números, debes pedírselo al usuario.
b) Suma de una fila que se pedirá al usuario (validar que elija una correcta)
c) Suma de una columna que se pedirá al usuario (controlar que elija una correcta)
d) Sumar la diagonal principal
e) Sumar la diagonal inversa
f) La media de todos los valores de la matriz

IMPORTANTE: hasta que no se haga la primera opción **a**, el resto de opciones no se deberán de ejecutar, simplemente mostrar un mensaje que diga que debes rellenar la matriz. Mostrar por pantalla el resultado de la ejecución de cada una de las opciones del menú.

EJERCICIO 5

Codifique la siguiente matriz de 2 dimensiones, que se corresponde a una máquina expendedora de golosinas donde la columna 1 es la golosina, la columna 2 el precio y la columna 3 la cantidad (stock) actual de golosinas

KitKat	32	10
Chicles	2	50
Caramelos de Menta	2	50
Huevo Kinder	25	10
Chetoos	30	10
Twix	26	10
M&M'S	35	10
Papas Lays	40	20
Milkybar	30	10
Alfajor Tofi	20	15
Lata Coca	50	20
Chitos	45	10

Tendremos un pequeño menú con las siguientes opciones:

a. **Pedir golosina**: pedirá la posición de la golosina que quiera. Esta máquina tiene golosinas en cada posición, identificados por su fila y columna, que será lo que introduzca el usuario al pedir una golosina, por ejemplo si el usuario teclea 2 significa que está pidiendo la golosina que está en la fila 2. Al seleccionar una golosina debe disminuir la cantidad disponible de la golosina. En caso de agotar el stock de la golosina deberá informar de la situación al cliente y solicitarle que seleccione otra golosina.
b. **Mostrar golosinas**: mostrara todas las golosinas con la cantidad actual disponible.
c. **Rellenar golosinas**: esta es una función exclusiva de un técnico por lo que para su ejecución nos pedirá una contraseña, si el usuario escribe "AdminXYZ" nos autorizara y pedirá la posición de la golosina y la cantidad a recargar. La cantidad ingresada se sumara a la cantidad actual existente.
d. **Apagar maquina**: sale del programa, antes de salir mostrara las ventas totales durante la ejecución del programa. Es decir la suma de todos los precios de las golosinas seleccionadas desde el inicio del programa.