ARREGLOS MULTIDIMENSIONALES-JAVA

MATRICES EN JAVA (Arreglos Bidimensionales)

Claro que sí. Basándome en la estructura, el estilo y los conceptos pedagógicos del documento PDF que subiste, he elaborado esta guía adaptada específicamente para el tema de **Matrices (Arreglos Bidimensionales)** utilizando tu script como base y añadiendo nuevos ejemplos prácticos.

MATRICES EN JAVA (Arreglos Bidimensionales)

Conceptos Fundamentales

Al igual que los arreglos unidimensionales descritos en tu documento, las matrices son objetos contenedores. Sin embargo, en lugar de una sola fila de datos, funcionan como una "tabla" con filas y columnas.

Sus características esenciales se mantienen:

- **Tamaño Fijo:** Una vez creada la matriz (ej. 3 filas, 4 columnas), su estructura no cambia.
- **Mismo Tipo:** Todos los elementos deben ser del mismo tipo de dato (`int`, `boolean`, etc.).
- **Acceso por Coordenadas:** Se accede mediante dos índices: el primero para la **fila** y el segundo para la **columna**, ambos iniciando en 0.

El siguiente código ilustra la declaración, inicialización, modificación y recorrido de matrices.

```
import java.util.Arrays;

public class Matrices {
    public static void main(String[] args) {
        // 1. Declaración
        int[][] nombreDeLaMatriz;

        // 2. Inicialización (3 filas, 4 columnas)
        nombreDeLaMatriz = new int[3][4];
```

```
        // 3. Impresión de la estructura completa
        // IMPORTANTE: deepToString para matrices
        System.out.println(Arrays.deepToString(nombreDeLaMatriz));

        // 4. Acceso y Lectura
        System.out.println(nombreDeLaMatriz[0][3]);

        // 5. Asignación / Modificación
        nombreDeLaMatriz[0][3] = 34;

        // Lectura tras modificación
        System.out.println(nombreDeLaMatriz[0][3]);

        // 6. Declaración e Inicialización Directa
        boolean[][] f = {{true, false},{false, true}};

        System.out.println(f[0][1]);
        System.out.println("------" + Arrays.deepToString(f));

        // 7. Recorrido con For Anidado
        for (int i = 0; i < f.length; i++) {          // Recorre
filas
            for (int j = 0; j < f[i].length; j++) {   // Recorre
columnas
                System.out.println(f[i][j]);
            }
        }

        // Para imprimir matrices usamos deepToString
        System.out.println(Arrays.deepToString(f));
    }
}
```

1. Declaración e Inicialización

En las matrices, agregamos un segundo par de corchetes [][] para indicar la segunda dimensión.

Concepto	Código de Ejemplo	Descripción
Declaración	int[][] matriz;	Declara una variable capaz de apuntar a un arreglo de arreglos (matriz) de enteros.
Inicialización por Tamaño	matriz = new int[3][4];	Reserva memoria para **3 filas** y **4 columnas**. Los valores inician en 0 (int) o null (objetos) por defecto.
Inicialización Directa	boolean[][] f = {{true, false}, ...};	Crea la matriz y asigna los valores inmediatamente. Java calcula el tamaño automáticamente basándose en los datos entre llaves {}.

2. Clase Arrays y Representación

A diferencia de los arreglos simples donde usábamos toString[7], las matrices requieren un método "profundo" para verse correctamente.

Concepto	Código de Ejemplo	Descripción
Arrays.deepToString()	Arrays.deepToString(f)	Método vital para matrices. Si usas solo toString, verás referencias de memoria. deepToString recorre recursivamente la matriz para mostrar los valores reales.

3. Acceso y Modificación

Los elementos se ubican mediante coordenadas [fila][columna].

Concepto	Código de Ejemplo	Descripción

Lectura	val = matriz[0][3];	Obtiene el valor ubicado en la **Fila 0**, **Columna 3**.
Asignación	matriz[0][3] = 34;	Sobrescribe el valor en esa posición específica con el número 34.
Longitud de Filas	matriz.length	Devuelve la cantidad de **filas** que tiene la matriz.
Longitud de Columnas	matriz[i].length	Devuelve la cantidad de columnas que tiene la fila i.

4. Recorridos (Bucles Anidados)

Para visitar cada celda, necesitamos un bucle dentro de otro (uno para bajar por las filas, otro para recorrer las columnas).

Tipo de Recorrido	Código de Ejemplo	Descripción

For Anidado	for (int i=0; i < f.length; i++) {
	
	for (int j=0; j < f[i].length; j++) {
	
	// codigo
		El bucle externo (i) controla la fila actual. El bucle interno (j) recorre todas las columnas de esa fila antes de que i incremente.

	} } 	

Ejemplos Adicionales

Ejemplo A: Matriz de String (Cine)

Simulación de una sala de cine pequeña (3 filas, 3 asientos) donde marcamos uno como "OCUPADO".

```
public class Cine {
    public static void main(String[] args) {
        // Declaración de 3x3
        String[][] sala = new String[3][3];

        // Llenado inicial
        for (int i = 0; i < sala.length; i++) {
            for (int j = 0; j < sala[i].length; j++) {
                sala[i][j] = "LIBRE";
            }
        }

        // Simulamos una reserva en Fila 1 (indice 0), Asiento 2
(indice 1)
        sala[0][1] = "OCUPADO";

        // Imprimir visualmente la sala
        for (int i = 0; i < sala.length; i++) {
            for (int j = 0; j < sala[i].length; j++) {
                System.out.print("[" + sala[i][j] + "] ");
            }
            System.out.println(); // Salto de línea al terminar
una fila
        }
    }
}
```

Ejemplo B: Suma de una Matriz (Acumuladores)

Recorrer una matriz numérica para sumar todos sus valores.

```
public class SumatoriaMatriz {
    public static void main(String[] args) {
        int[][] numeros = {
            {10, 20, 30},
            {5, 5, 5},
            {1, 1, 1}
        };

        int sumaTotal = 0;

        // Recorrido para sumar
        for (int i = 0; i < numeros.length; i++) {
            for (int j = 0; j < numeros[i].length; j++) {
                sumaTotal += numeros[i][j]; // Acumulador
            }
        }

        System.out.println("La suma total de la matriz es: " +
sumaTotal);
    }
}
```

Ejercicios:

1. Practicar los bucles anidados (`for` dentro de `for`) para llenar una matriz.

Enunciado: Crea una matriz de caracteres (`char`) de 5 filas y 5 columnas.

1. Usa dos bucles para rellenar **todas** las posiciones con el símbolo `'#'`.
2. Imprime la matriz para ver un cuadrado perfecto de símbolos.

2. Entender cómo acceder a una posición específica usando coordenadas `[fila][columna]`.

Enunciado: Simula un tablero de Triqui (o Gato) vacío de 3×3 usando `String`.

1. Inicializa la matriz explícitamente con guiones `"-"` (indicando vacío).

2. **Sin usar bucles**, coloca manualmente una `"X"` en el centro del tablero (fila 1, columna 1).
3. Coloca una `"O"` en la esquina superior izquierda (fila 0, columna 0).
4. Imprime el tablero usando `Arrays.deepToString()` para ver el resultado rápido.

3. Recorrer una matriz pequeña para sumar sus valores.

Enunciado: Tienes una caja con 2 filas y 2 columnas de compartimentos con monedas.

1. Crea la matriz `int[][] caja = {{5, 10}, {20, 50}};`.
2. Crea una variable entera `total = 0`.
3. Recorre la matriz sumando cada moneda a la variable `total`.
4. Imprime cuánto dinero hay en total.