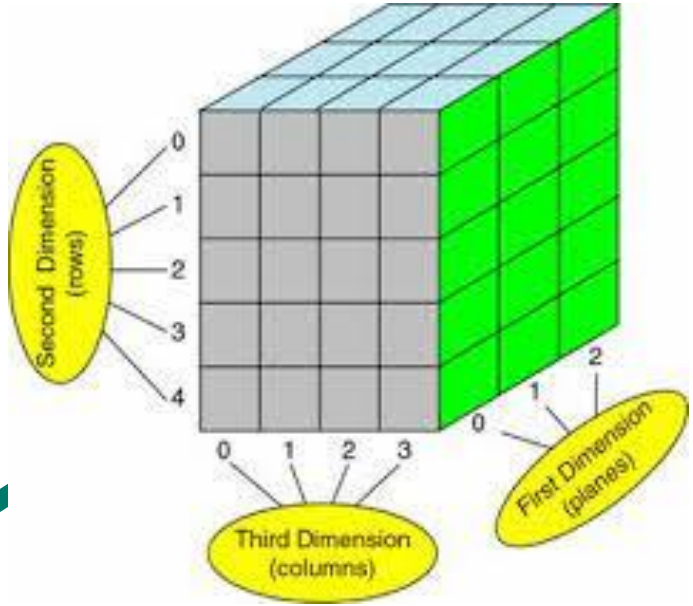
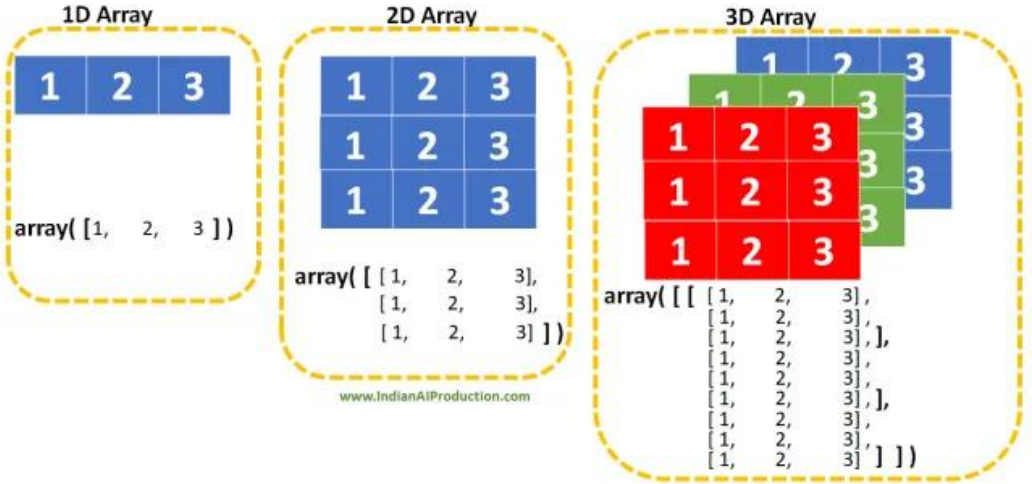
ARRAYS
MULTIDIMENSIONALES



Encarnación Sánchez Gallego

ÍNDICE

1. Arrays multidimensionales.
2. Asignación de arrays
3. Clonación de arrays.

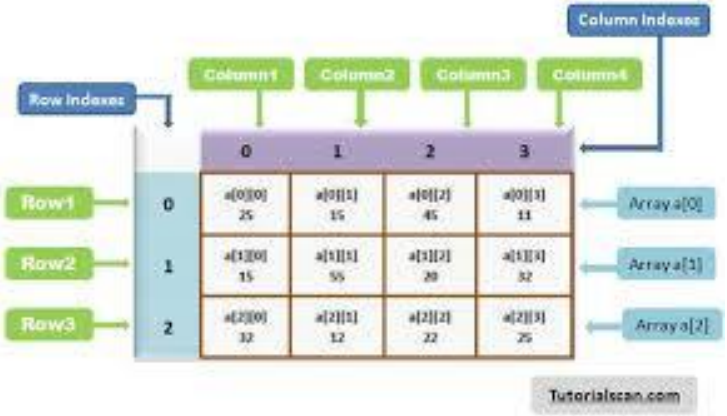


ARRAYS MULTIDIMENSIONALES

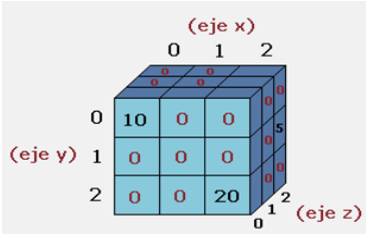
Al declarar un array se indica entre corchetes la cantidad de posiciones que deseamos tener. Esto marca el máximo número de posiciones que contiene el array y es llamada una dimensión. Es posible declarar los arrays con más dimensiones. Las matrices multidimensionales son arrays de arrays, donde cada elemento del array contiene la referencia de otro array. Se crea una matriz multidimensional al agregar un conjunto de corchetes ([]) por dimensión.

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

El código anterior creará un array de dos dimensiones, o lo que es lo mismo, creará un array que contendrá 3 arrays de 5 números cada uno.



ARRAYS MULTIDIMENSIONALES



Ejemplos de arrays multidimensionales:

```
int[][] intArray = new int[2][2];                  // Un array 2D o matriz de enteros

float[][] floatArray = new float[3][2];            // Un array 2D o matriz de float

String [][] stringArray = new String[4][3]         // Un array 2D o matriz de String

int[][][] intArray = new int[2][3][4];             // Un array 3D de enteros

int[][][] floatArray = new float[2][3][4];         // Un array 3D de float

String [][] stringArray = new String[2][3][4]    // Un array 3D de String
```

ARRAYS MULTIDIMENSIONALES

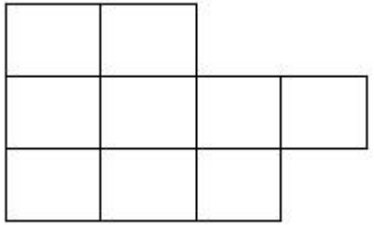
Si declaramos 2 dimensiones, entonces estamos creando una matriz bidimensional con varias filas y cada fila tendremos una cantidad de columnas. De hecho, es posible que en cada fila se tengan un número distinto de elementos o columnas. Por ejemplo, podemos declarar e inicializar la siguiente matriz bidimensional

```
int[][] matrizIrregular={{1,2},{3,4,5,6},{7,8,9}};
```

La primer fila tiene cuatro elementos {1,2}.

La segunda fila tiene dos elementos {3,4,5,6}.

La tercera fila tiene seis elementos {7,8,9}.



ARRAYS MULTIDIMENSIONALES

Para mostrar los elementos de este array bidimensional irregular escribimos el siguiente código:

```
for (int i=0; i < matrizIrregular.length; i++) {
        for (int j=0; j < matrizIrregular[i].length; j++) {
                System.out.print(matrizIrregular[i][j]+"\t"); //Usamos tabulador para separar
        }
        System.out.println("");
}
```

Como podemos apreciar, matrizIrregular.length nos proporciona el número de filas (tres), y luego matrizIrregular[i].length, nos proporciona el número de elementos en cada fila.

ARRAYS MULTIDIMENSIONALES

Al igual que en los arrays unidimensiones, en los arrays multidimensionales se puede utilizar la propiedad length para saber a longitud, pero con la salvedad como vimos antes de que existirán varios “niveles” de longitudes. Veamos a qué correspondería cada nivel:

```
int tabla[][]={{1,2,3},{4,5},{6,7,8,9}};

System.out.println("Longitud de tabla nivel 1: "+tabla.length);

System.out.println("Longitud de tabla nivel 2.0: " +tabla[0].length);

System.out.println("Longitud de tabla nivel 2.1: " +tabla[1].length);

System.out.println("Longitud de tabla nivel 2.2: " +tabla[2].length);
```

ARRAYS MULTIDIMENSIONALES

```
class multiDimensional {                                    // Ejemplo
    public static void main(String args[]) {
        int multiArray[][] = { {2,7,9},{3,6,1},{7,4,2} }; // declarar e inicializar array 2D
        for (int i=0; i< 3 ; i++) {                         // imprimir array 2D
            for (int j=0; j < 3 ; j++)
                System.out.print(multiArray[i][j] + " ");
            System.out.println();
        }
    }
}
```

ARRAYS MULTIDIMENSIONALES

Ejemplo de recorrido para sumar todos los elementos de la matriz:

```
static int sumaMatriz(int[][] matriz) {
        int suma = 0;
        for (int i = 0; i < matriz.length; i++) {
                for (int j = 0; j < matriz[i1].length; j++) {
                        suma += matriz[i][j];
                }
        }
        return suma;
}
```

ARRAYS MULTIDIMENSIONALES

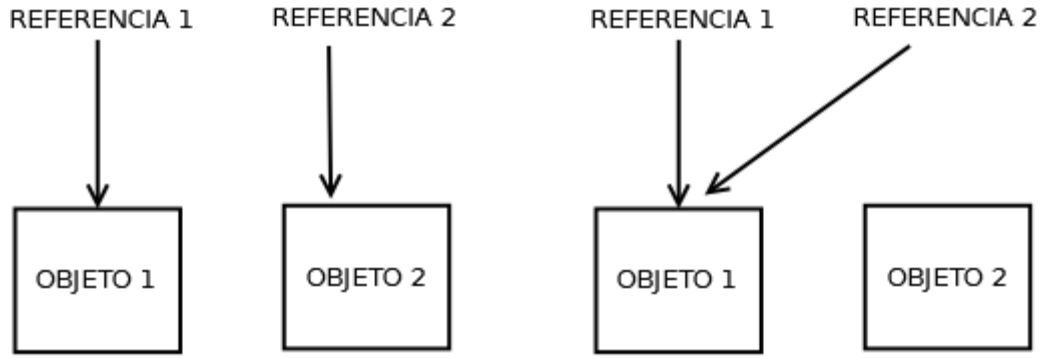
Ejemplos de inicialización:

```
int[][] array1={{0,1,2},{3,4,5},{6,7,8},{9,10,11}};      // Declaración e inicialización fija
int[][][] array2={{{0,1},{2,3}},{{0,1},{2,3}}};          // Declaración e inicialización fija
int[][] inicializarArray (int n, int m){                 // Función que crea y devuelve una matriz
    int[][] ret=new int[n][m];                           // Crea la matriz con los valores dados
    for (int i=0;i<n;i++)
        for (int j=0;j<m;j++)
            ret[i][j]=n*m;                               //Inicializa cada posición
    return ret;
}
```

ASIGNACIÓN DE ARRAYS

Al igual que con otros objetos, cuando se asigna una variable de referencia de un array a otra, simplemente está cambiando a qué objeto se refiere dicha variable.

Al hacerlo, no está causando que se realice una copia del array, ni hace que el contenido de un array se copie en el otro, sino que se crea una nueva referencia al mismo objeto.



Al asignar los arrays, realmente se está haciendo una referencia nueva al mismo objeto, como en este segundo caso.

ASIGNACIÓN DE ARRAYS. EJEMPLO

```
//Asignación de variables de referencias en array
class AsignacionArray {
    public static void main(String args[]) {
        int i;
        int num1[] = new int;
        int num2[] = new int;
        for (i=0; i<10 ; i++)
            num1=i;
        for (i=0; i<10 ; i++)
            num2=-i;
        System.out.print("num1: ");
        for (i=0; i<10 ; i++)
            System.out.print(num1  + " ");
        System.out.println();
        System.out.print("num2: ");
        for (i=0; i<10 ; i++)
            System.out.print(num2  + " ");
        System.out.println();
        num2=num1; //Asignación de referencias
        System.out.print("num2 después: ");
        for (i=0; i<10 ; i++)
            System.out.print(num2  + " ");
        System.out.println();
        //ahora opera el array num1 a través de
num2
        num2=99;
        System.out.print("num1 después: ");
        for (i=0; i<10 ; i++)
            System.out.print(num1  + " ");
        System.out.println();
    }
}
```

CLONACIÓN DE ARRAYS

Para clonar un array dimensional único, como Object[], se debe realizar una “copia profunda” para que el nuevo array contenga copias de los elementos del array original en lugar de referencias. Por ejemplo:

```
class Test {                    // Programa para clonar de arrays unidimensionales
        public static void main(String args[])  {
                int intArray[] = {1,2,3};                       //Array original
                int cloneArray[] = intArray.clone();                    /Array para clonar
                System.out.println(intArray == cloneArray); // Será false al estar clonado
                for (int i = 0; i < cloneArray.length; i++) {
                        System.out.print(cloneArray[i]+" ");
                }
        }
}
```

CLONACIÓN DE ARRAYS

Existe otra forma de hacer una copia pero sólo de parte de los arrays mediante la función System.arraycopy() que lleva 5 parámetros, el array original, la posición a empezar a copiar del original, el array destino, la posición a empezar a copiar en el destino y el número de elementos a copiar:

```
int intArray[] = {9,8,7,6,5};                    //Array original
int cloneArray[] = new int[4];                   //Array destino que no tiene la misma longitud
System.arraycopy(intArray, 1, cloneArray, 1, 2);          //Copiado del array
System.out.println(intArray == cloneArray);               // Será false al ser una copia
for (int i = 0; i < cloneArray.length; i++) {
        System.out.print(cloneArray[i]+" ");
}
```

WEBGRAFÍA

- https://w3api.com/Java/System/out/
- https://www.discoduroderoer.es/arrays-de-objetos-en-java/
- https://javadesdecero.es/arrays/unidimensionales-multidimensionales/