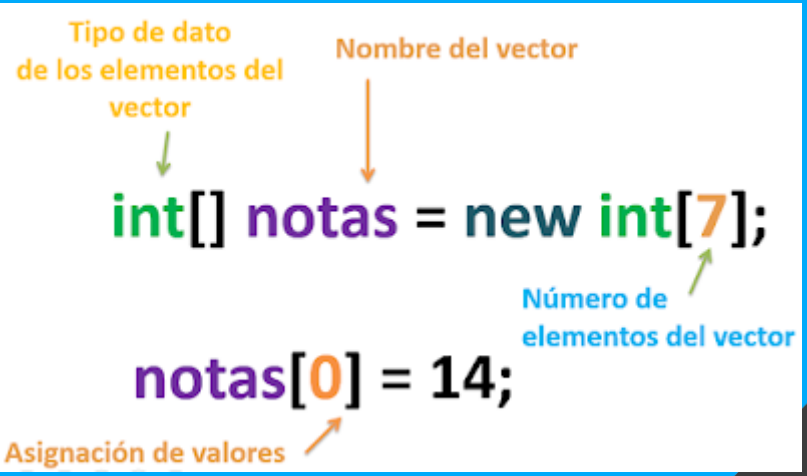
//Arrays (vectores)



UTEC
Universidad Tecnológica

LICENCIATURA EN TECNOLOGÍAS DE LA INFORMACIÓN

Arrays (vectores) en JAVA

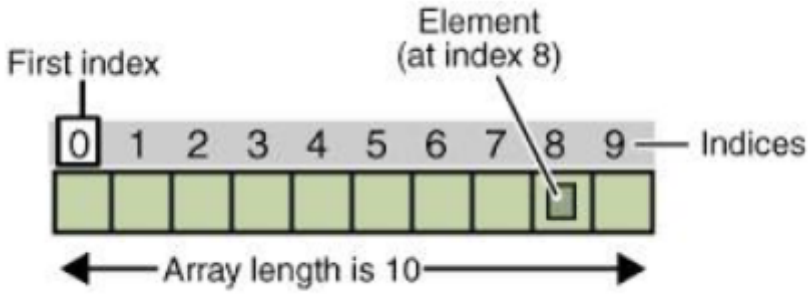
Introducción

Un array (arreglo) es una estructura de datos que contiene una colección de datos del mismo tipo, estas son usadas como contenedores que almacenan uno o más datos relacionados, en lugar de declarar cada dato de manera independiente. Por medio de los arreglos veremos cómo crear un elemento que nos permite definir una “variable” que contenga diferentes datos del mismo tipo asociados al mismo identificador.

Puedes pensar en él como un conjunto de celdas numeradas. Puedes colocar datos en cada celda.

Índice de un array

El índice o index de un array es la posición que se encuentra ubicado el valor almacenado.



En este ejemplo anterior vemos cómo se manejan los índices en Java, el primer dato arranca el índice 0, y así sucesivamente. En este ejemplo tenemos un array de 10 elementos, donde los índices van desde 0 a 9. Recuerden que el último elemento siempre va a ser igual al largo del array menos 1, en la imagen anterior sería (largo)=10-1 = 9 último índice.

Veamos un ejemplo de un array de int:

0	***1***	***2***	***3***	***4***	***5***	***6***	***7***	***8***	***9***
45	10	19	245	78	5	-5	87	103	589

La primera fila indica el índice, y la segunda fila tiene los valores de los elementos del array para ese índice.

¿Qué valor tiene el elemento de dicho array en el índice 4? 78.

¿Y en el índice 2? 19.

Resumiendo lo visto hasta ahora de los arrays.

El tipo de dato Array nos permite almacenar un conjunto de valores dentro de la misma variable.

Un array o arreglo, es un tipo de dato estructurado que permite almacenar un conjunto de datos homogéneos, es decir, todos los elementos deben ser del mismo tipo.

A los datos almacenados en un array se les denomina elementos.

El largo del array es la cantidad de elementos que este almacena.

La posición (o índice) de los elementos en los array arranca en 0.

Por lo tanto, la posición del primer elemento de un array es 0.

La posición del último elemento del array es el largo -1.

Un array siempre tiene un tamaño fijo, que lo definimos cuando lo declaramos (ahora veremos bien cómo es que esto se hace) y una vez definido el tamaño no puede cambiar durante la ejecución del programa.

Nunca puedo acceder a un índice que se escape del rango del array (de 0 a n-1, siendo n el tamaño total del array).

Si hago esto, cuando el programa esté corriendo e intente acceder a un índice que se exceda del rango, el programa lanzará un error.

Los elementos almacenados en un array deben ser del mismo **tipo de datos.**

Sintaxis de Array

Veamos como es la sintaxis en Java para declarar un array .

tipoDeDato [] nombreVariable = new tipoDeDato [LargoArray];

donde:

tipoDeDato es el tipo de dato de Java que va almacenar el array.

nombreVariable es el nombre de la variable en este caso un array

largoArray es la cantidad de elementos que puede contener. (es fija)

Veremos algunos ejemplos de cómo definir un array:

```
String[] primerArray = new String[20];
int segundoArray [] = new int[10];
float[] otroArray = new float[5];
```

En estos ejemplos tenemos primero un array de String de 20 elementos, luego tenemos un array de enteros(int) de 10 elementos y por último un array de float de 5 elementos.

Después de crear un array utilizando el operador new, sus celdas contienen valores por defecto. Para tipos numéricos, el valor por defecto es 0, para el tipo booleano es false, y para tipos de referencia es null, en el ejemplo anterior String sería null y los numéricos 0 y 0.0 respectivamente.

También, se puede definir el array, indicando directamente los elementos que quiero que el mismo contenga. Por ejemplo:

```
String[] primerArray = {"Gaby","Guille","Nico","Juan"};
int segundoArray [] = {1,5,10,-65,1000,23};
float[] otroArray = {10.5f,21.5f,587.23f,34f};
```

Vemos que no se define explícitamente el largo del array, sino que se le están asignando los elementos de cada uno. Esto se hace con {} y dentro de los corchetes, van los elementos del array separados por coma (,).

De esta manera, ¿Qué tamaño tendrán los tres arrays definidos?
La respuesta es 4, 6 y 4 respectivamente (su cantidad de elementos).

Si ahora preguntamos,
¿Qué valor tiene el array "otroArray" en el índice 3?
La respuesta correcta sería 34 Ya que, el índice se arranca a contar desde 0. Por lo tanto el índice 3 corresponde al cuarto elemento o posición del array.

Declaración de los tipos de datos

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

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

Aclarar que los valores por defecto son los siguientes:

a) Para números el valor cero “0”.

b) Para cadenas y letras el valor vacío.

c) Para booleanos el valor false.

Accediendo a los valores

Ahora que tenemos la noción y el concepto de lo que es un array, veamos cómo podemos acceder a los elementos del mismo, ya sea para obtener un valor, o guardar un valor en un índice.

Al array debemos tratarlo como una variable de las ya vistas. La única diferencia es que debemos siempre indicarle el índice del elemento al que queremos acceder o guardar el valor.

Esto lo hacemos de la siguiente manera:

● varArreglo[indice] : devuelve el valor del elemento en indice del array nombre varArreglo

● varArreglo[indice] = valor : asigna valor al elemento en indice del array de nombre varArreglo

Ejemplo1

```
public class Ej1 {
    public static void main(String[] args) {
        int[] varArray ={2,3,7,1,10,34,665};
        int suma = varArray[0]+varArray[4];
        System.out.println(suma);
    }
}
```

¿Qué valor tendrá la variable suma al finalizar la ejecución del programa?

Estamos sumando, de la variable de tipo array llamada varArray, los valores que se encuentran almacenados en el índice 0 y en el 4 (2 y 10 respectivamente), por lo tanto el valor que quedará almacenado será la suma de los mismos: 12.

```
"C:\Program Files\Java\jdk-17.0.2\b
12

Process finished with exit code 0
```

Que se ve en la salida por la consola si ejecutamos el programa anterior.

Como se puede ver, para acceder al valor de un elemento en particular, se pone el nombre de la variable del array, seguido por paréntesis rectos y dentro de los mismos, el número de índice que nos interesa acceder.

Luego, si queremos modificar algún valor almacenado en un índice, debemos utilizar una sentencia de asignación, donde la variable que queremos asignarle es un elemento del array.

Ejemplo2

```
public class Ej1 {
    public static void main(String[] args) {
        int[] varArray ={2,3,7,1,10,34,665};
        // Le asignamos al array un valor al elemento que esta en indice 1
        varArray[1]=100;
        // Le asignamos al array un valor al elemento que esta en indice 3
        varArray[3]=35;
        System.out.println("muestro un elemento del array el indice 1 "+varArray[1]);
        System.out.println("-".repeat( count: 70));
        System.out.println("Todos los elementos del array ");
        System.out.println("-".repeat( count: 70));
        for(int i=0; i< varArray.length; i++){
            System.out.println("indice "+i+" valor: "+varArray[i]);
        }
    }
}
```

En este ejemplo, hemos modificado los valores de los índices 1 y 3 del array “varArray”. De esta manera, ahora en el índice 1 tenemos almacenado el valor 100 y no el 3 y lo mismo sucede con el índice 3, donde no tendremos más almacenado el valor 1 y ahora tendremos el 35.

En este programa además estamos mostrando uno de los elementos que está en el índice 1.

Y luego con un bucle for, muy usado para trabajar con arrays muestra todos los elementos del array con su índice y valor.

Esta sería la salida por consola de ejecutar este programa.

```
muestro un elemento del array el indice 1 100
------------------------------------------------------------------------
Todos los elementos del array
------------------------------------------------------------------------
indice 0 valor: 2
indice 1 valor: 100
indice 2 valor: 7
indice 3 valor: 35
indice 4 valor: 10
indice 5 valor: 34
indice 6 valor: 665
```

Operaciones con Array

El array es una variable que almacena múltiples valores en vez de uno solo, pero tiene el mismo comportamiento que una variable.

La gran diferencia es que no puedo hacer operaciones con arrays, como las que hago con números o strings.

Por ejemplo, no puedo sumar dos variables de tipo int[] con el operador +.

Pero Java nos ofrece métodos propios de los array

length

Existe el método length, que nos devuelve el largo total del arreglo. Se utiliza nombreArray.length.

Ejemplo 3

```
public class Ej3 {
    public static void main(String[] args) {
        int arrayUno[] = new int[25];
        String arrayDos[] = new String[30];
        System.out.println("Largo arrayUno :"+arrayUno.length);
        System.out.println("Largo arrayDos :"+arrayDos.length);
    }
}
```

```
Largo arrayUno :25
Largo arrayDos :30

Process finished with exit code 0
```

Ya que length devuelve el largo total de elementos. Y como vimos anteriormente en este caso todos los elementos quedan inicializados con el valor por defecto de int (ya que en este caso era un array de int) que es 0.

Veamos por último un ejemplo un poco más complejo. Dado un array de números enteros, queremos tener un algoritmo que nos diga si cierto número se encuentra o no en el array.

En el ejemplo, tendremos explícitamente el número y el array como variables.

Ejemplo 4

```
public class Ej4 {
    public static void main(String[] args) {
        int numeros[] ={23,34,54,-23,-342,45,65,10,101};
        int numero= 45;
        boolean encontre=false;
        int i=0;
        while(i<numeros.length && !encontre){
            if(numeros[i]==numero){
                encontre=true;
            }
            i++;
        }
        System.out.println("Encontre el 45 "+encontre);
    }
}
```

```
Encontre el 45 true

Process finished with exit code 0
```

Asumamos que tenemos una variable de tipo array de int “numeros” que no sabemos lo que contiene y además un número entero “numero”.

Primero que nada, inicializamos la variable encontre como false, ya que de recorrer el array y no encontrar el número, la variable encontre debe tener valor false.

Luego, con una sentencia while, recorremos todos los índices del array: desde 0

hasta su largo menos uno (que es el último índice) i< numeros.length.

La condición podría ser también: i <= numeros.length - 1. Ya que significaría

lo mismo. La condición es compuesta, ya que si encuentra el numero buscado al cambiar la variable booleana salimos del while, y tenemos un índice i que se va incrementando en cada iteración o loop del while.

En cada iteración, se va a preguntar si el elemento del array correspondiente

al índice i, es igual al elemento que buscamos (numero). Si es, cambiamos la variable "encontré" a true y si es así salimos del bucle.

Importante:

Al ejecutar un programa, si se intenta acceder a un índice que está fuera del rango

del array esto dará error en el tiempo de ejecución.

Ejemplo 5

```
public class Ej5 {
    public static void main(String[] args) {
        int numeros[] = new int[20];
        int valor = numeros[30];
        System.out.println(valor);
    }
}
```

Al ejecutar este programa, donde se accede al elemento de índice 30, que está fuera

de rango, ya que el largo es 20. Este programa da error de ejecución esto se puede

ver en la consola, ya que imprime en rojo el siguiente texto:

```
"C:\Program Files\Java\jdk-17.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.3\lib\idea_rt.jar
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException Create breakpoint : Index 30 out of bounds for length 20
    at utec.Ej5.main(Ej5.java:6)

Process finished with exit code 1
```

Qué quiere decir esto:

Exception in thread "main"

Nos dice que ocurrió un error en el método main

java.lang.ArrayIndexOutOfBoundsException: Index 30 out of bounds for length 20.

Es una excepción (que veremos en el siguiente módulo) indice fuera de rango.

Ahora veremos que pasa en el siguiente ejemplo de un caso límite:

Ejemplo 6

```java
public class Ejó {
    public static void main(String[] args) {
        int vector[] = new int[20];
        int var= vector[vector.length];
    }
}
```

¿Qué pasa en este caso ? También tenemos un caso de fuera de rango, ya que el último elemento del vector en este caso seria 19 (largo - 1) .

```
"C:\Program Files\Java\jdk-17.0.2\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.3\lib\idea_rt.
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException Create breakpoint : Index 20 out of bounds for length 20
    at utec.Ejó.main(Ejó.java:6)

Process finished with exit code 1
```

Veremos algunos otros ejemplos que pueden ser de utilidad con los arrays, en este caso de una dimensión.

Ejemplo 7

```java
public class Ej7 {
    public static void main(String[] args) {
        int numeros[] = {15,23,24,58,-95,245};
        // sumar los elementos de un array
        int suma=0; // inicializo una variable tel tipo acumulador
        for(int i=0; i<numeros.length; i++)
            suma+=numeros[i];
        System.out.println("La suma de numeros es :"+suma);
    }
}
```

```
La suma de numeros es :270

Process finished with exit code 0
```

Ejemplo 8

```java
public class Ej8 {
    public static void main(String[] args) {
        int numeros[] = {15,23,24,58,-95,245};
        // mostrar por consola los numeros pares

        for(int i=0; i<numeros.length; i++) {
            if(numeros[i]%2==0){
                System.out.println(numeros[i]);
            }
        }
    }
}
```

```
24
58

Process finished with exit code 0
```

En este ejemplo, se recorre el array con un for, y se muestran los números que son pares, para saber si son pares hacemos el módulo de 2 (resto de dividir por 2) si nos da 0 es par, y en ese caso lo mostramos por consola.

Ejemplo 9

```java
public class Ej9 {
    public static void main(String[] args) {
        String nombres[] = {"Ana","Karina","Hugo","Diego"};
        for(int i=0; i<nombres.length; i++){
            System.out.println(nombres[i]+" largo nombre: "+nombres[i].length());
        }
    }
}
```

La salida por consola es:

```
Ana largo nombre: 3
Karina largo nombre: 6
Hugo largo nombre: 4
Diego largo nombre: 5
```

En este caso es un array de String que tiene almacenado nombres, y luego lista cada nombre y el largo del mismo.

Ejemplo 10

```
public class Ej10 {
    public static void main(String[] args) {
        Persona p1 = new Persona( nombre: "Juan", edad: 45);
        Persona p2 = new Persona( nombre: "Adriana", edad: 22);
        Persona p3 = new Persona( nombre: "Luis", edad: 28);
        Persona [] personas = {p1,p2,p3};
        double suma=0;
        for (int i=0; i< personas.length;i++){
            System.out.println(personas[i]);
            suma+=personas[i].getEdad();
        }
        double promedio=suma/ personas.length;
        System.out.println("El promedio de las edades es "+promedio);
    }
}
```

```
Persona{nombre='Juan', edad=45}
Persona{nombre='Adriana', edad=22}
Persona{nombre='Luis', edad=28}
El promedio de las edades es 31.666666666666668
```

Como cuando hacemos un array de String (que es una clase) podemos hacer un array de una clase creada por nosotros, en este caso de Persona. La persona tiene nombre y edad, creamos 3 instancias de Persona y luego las colocamos dentro un array de 3 elementos.
Recorrimos el array y mostramos los datos de todas las personas, y calculamos el promedio de las edades de las personas. El promedio se calcula sumando todas las edades y dividendo por la cantidad de personas, esto lo tenemos con el largo del array.

El mismo ejemplo anterior, podemos recorrer el array de personas, usando un for-each

Ejemplo 11

```java
public class Ej11 {
    public static void main(String[] args) {
        Persona p1 = new Persona( nombre: "Juan", edad: 45);
        Persona p2 = new Persona( nombre: "Adriana", edad: 22);
        Persona p3 = new Persona( nombre: "Luis", edad: 28);
        Persona [] personas = {p1,p2,p3};
        double suma=0;
        for (Persona p: personas){ // Utilizando un for-each
            System.out.println(p);
            suma+=p.getEdad();
        }
        double promedio=suma/ personas.length;
        System.out.println("El promedio de las edades es "+promedio);
    }
}
```

Aca solo cambiamos el for con los índices, por el for-each, de esta forma se puede recorrer un array para poder visualizar sus datos, y poder operar con ellos.

Arrays Multidimensionales

Un arrray unidimensional es por ejemplo **int[] numeros = new int[5];**

Un array cuyos elementos son arrays, es decir, un array de arrays, es bidimensional. Es como un tablero que tiene un número de fila y un número de columna. Los array bidimensionales se pueden aplicar a diferentes juegos como por ejemplo el ajedrez, el buscaminas, el ta-te-ti, la batalla naval entre otros, donde se utilizan filas y columnas.

Ejemplo de un array bidimensional:

```
int[][] nombre = new int[ancho][alto];
```

int : es el tipo de dato del array

nombre: es el nombre a la variable array

ancho: es la cantidad de filas.

alto: la cantidad de columnas.

El array de dos dimensiones es conocido matemáticamente como matriz.

Vemos ejemplos de cómo se definen los array de múltiples dimensiones:

Ejemplo 12

```
public class Ej12 {
    public static void main(String[] args) {
        // arrays multidimensionales
        int[] numeros = new int[5]; // array unidimensional
        int[][] datos = new int[5][3]; // array bidimensional o matriz
        int[][][] arrayTri = new int[5][4][2]; // array tridimensional


    }
}
```

Veamos ahora otro ejemplo si tenemos un array de dos dimensiones:

Ejemplo 13

```java
public class Ej13 {
    public static void main(String[] args) {
        int numeros[][] = new int[2][5];
        numeros[1][1]=5;
        // recorremos el array y mostramos todos los datos
        for(int i=0; i<numeros.length;i++){
            for(int j=0; j< numeros[0].length;j++){
                System.out.print(numeros[i][j]+" -- ");
            }
            System.out.println();
        }
    }
}
```

Salida por la consola

```
0 -- 0 -- 0 -- 0 -- 0 --
0 -- 5 -- 0 -- 0 -- 0 --
```

En este ejemplo tenemos 2 filas y 5 columnas, al definirlo de esta forma, asignamos en 0 todos los elementos por defecto, ya que el tipo de dato es int. Luego hacemos una asignación en el índice 1,1 que equivalen fila 1 columna 1 le asignamos el valor 5.

Luego mostramos con un for anidado la forma de recorrer la estructura de dos dimensiones, esta forma de usar el for es nuevo, por lo que tenemos que usar dos índices, uno para las filas y otro para las columnas, el for que está dentro del otro, se recorre varias veces, en este ejemplo se usa 2 veces. En este caso arranca el primer for en i=0 , y para i igual 0 recorre el for de las j desde 0 a 4, luego pone el i=1, y recorre nuevamente el for de j desde 0 a 4, y el programa termina.

También se puede realizar una inicialización rápida para los array bidimensionales.

Veamos un ejemplo:

Ejemplo 14

```java
public class Ej14 {
    public static void main(String[] args) {
        // inicializar array bidimensionales
        int[][] matriz = {
                {1, 2, 3},
                {4, 5, 6},
                {7, 8, 9}
        };
        int primerElemento = matriz[0][0]; // Accediendo al primer elemento de la matriz
        int elementoCentral = matriz[1][1]; // Accediendo al elemento central de la matriz
        int ultimoElemento = matriz[matriz.length - 1][matriz[0].length - 1]; // Accediendo al último elemento de la matriz
        System.out.println("Primer elemento: " + primerElemento);
        System.out.println("Elemento central: " + elementoCentral);
        System.out.println("Último elemento: " + ultimoElemento);
        // Mostando todos los elementos de la matriz
        for(int i=0;i< matriz.length;i++){
            System.out.println("Para i ="+i);
            for(int k=0;k<matriz[0].length;k++){
                System.out.print(matriz[i][k]+" - ");
            }
            System.out.println();
            System.out.println("-----------------------");
        }
    }
```

Aca vemos con se inicializa el array bidimensional, luego se muestran algunas posiciones en particular y por último se muestra toda la estructura del array llamada matriz.

La salida por la consola sería :

```
Primer elemento: 1
Elemento central: 5
Último elemento: 9
Para i =0
1 - 2 - 3 -
-----------------------
Para i =1
4 - 5 - 6 -
-----------------------
Para i =2
7 - 8 - 9 -
-----------------------

Process finished with exit code 0
```

Ejemplo 15

```java
public class Ej15 {
    public static void main(String[] args) {
        int[][] matriz = {
                {1, 2, 3, 4},
                {5, 6, 7, 8},
                {9, 10, 11, 12}
        };

        int filas = matriz.length; // Obteniendo el número de filas de la matriz
        int columnas = matriz[0].length; // Obteniendo el número de columnas de la matriz

        System.out.println("La matriz tiene " + filas + " filas y " + columnas + " columnas.");
    }
}
```

Salida por la consola

```
La matriz tiene 3 filas y 4 columnas.

Process finished with exit code 0
```

Vamos a ver otro ejemplo cargando en una matriz las tablas de multiplicar del 0 al 9, en cada elemento va a estar la multiplicación de los índices.

Ejemplo 16

```java
package utec;

public class Ej16 {
    public static void main(String[] args) {
        // Cargamos en las tablas de 0 a 9
        int tablas[][]= new int [10][10];
        for (int i=0;i<tablas.length;i++){
            for (int k=0;k<tablas[0].length;k++){
                tablas[i][k]=i*k;
            }
        }
        // Cuanto es 8 x 7
        System.out.println(" 8 x 7 = "+tablas[8][7]);
    }
}
```

```
C:\Program Files\Java\jdk-17.0.2\
8 x 7 = 56

Process finished with exit code 0
```

Arrays Escalonados en Java

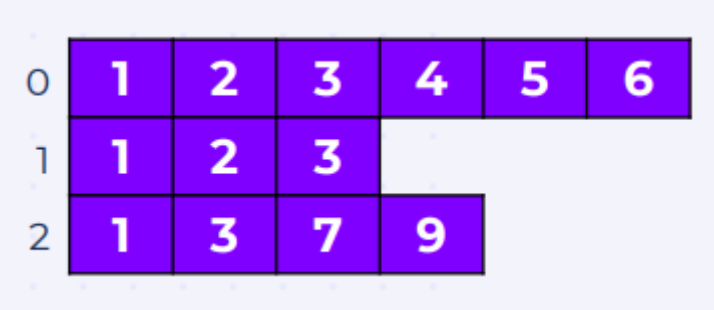
Si necesitas crear una matriz bidimensional cuyas columnas varían en longitud: Primero, necesitas crear un 'contenedor de contenedores', es decir, el primer arreglo que almacenará referencias a arreglos unidimensionales.

int[][] nombre = new int[filas][];

```java
int[][] matrix = new int[3][];
matrix[0] = new int[]{1, 2, 3, 4, 5, 6};
matrix[1] = new int[]{1, 2, 3};
matrix[2] = new int[]{1, 3, 7, 9};
for (int i = 0; i < matrix.length; i++) {
    for (int j = 0; j < matrix[i].length; j++) {
        System.out.print(matrix[i][j] + " ");
    }
    System.out.println();
}
```

Crea un arreglo
Llena el arreglo con valores

Bucle externo que itera sobre las filas del arreglo.
Bucle interno que itera sobre las celdas de una sola fila.



```java
public class Ej17 {
    public static void main(String[] args) {
        // array escalonados en Java
        int [][] matriz = new int[3][];
        matriz[0] = new int[]{1,2,3,4,5,6};
        matriz[1]= new int[]{1,2,3};
        matriz[2] = new int[]{1,3,7,9};
        for(int i=0; i< matriz.length;i++){
            for(int j=0;j < matriz[i].length;j++){
                System.out.print(matriz[i][j]+" ");
            }
            System.out.println();
        }
    }
}
```

```
1 2 3 4 5 6
1 2 3
1 3 7 9
```

Veamos ahora un ejemplo un array multidimensional, en este caso un ejemplo de un cine, en el cual tiene 2 salas , en las cuales tiene asientos ubicados con filas y columnas, que nos indican si tiene disponible en cada cine disponibilidad en sus butacas.

Este código primero inicializa todos los asientos como disponibles(true). Luego, cambia el estado de algunos asientos específicos a ocupados(false) para el ejemplo. Finalmente, imprime el estado de cada asiento en cada sala para demostrar como se puede acceder y manipular los elementos en un arreglo tridimensional.

Ahora les mostraremos el ejemplo:

```java
public class Eje18 {
    public static void main(String[] args) {
        // Crear e inicializar el arreglo tridimensional para las salas de cine
        // cine[sala][fila][asiento]
        boolean[][][] cine = new boolean[2][3][4];

        // Supongamos que todos los asientos están inicialmente disponibles, los inicializamos a true
        for (int sala = 0; sala < cine.length; sala++) {
            for (int fila = 0; fila < cine[sala].length; fila++) {
                for (int asiento = 0; asiento < cine[sala][fila].length; asiento++) {
                    cine[sala][fila][asiento] = true;
                }
            }
        }
       // Ocupamos algunos asientos para el ejemplo
        // Ocupar el asiento en la primera sala, primera fila, primer asiento
        cine[0][0][0] = false;
        cine[0][0][1] = false;
        cine[0][1][2] = false;


       // Mostrar el estado de los asientos en las salas
        for (int sala = 0; sala < cine.length; sala++) {
            System.out.println("Sala " + (sala + 1) + ":");
            for (int fila = 0; fila < cine[sala].length; fila++) {
                for (int asiento = 0; asiento < cine[sala][fila].length; asiento++) {
                    System.out.print(cine[sala][fila][asiento] ? "Disponible " : "Ocupado ");
                }
                System.out.println("\n");
            }
            System.out.println();
        }

    }
}
```

La salida por consola sería:

```
Sala 1:
Ocupado Ocupado Disponible Disponible

Disponible Disponible Ocupado Disponible

Disponible Disponible Disponible Disponible


Sala 2:
Disponible Disponible Disponible Disponible

Disponible Disponible Disponible Disponible

Disponible Disponible Disponible Ocupado
```

La Clase Array

La clase `java.util.Arrays` proporciona varios métodos de utilidad que facilitan el manejo y la manipulación de arrays en Java.

A continuación, se presentan algunos ejemplos prácticos y bien de cómo utilizar estos métodos en programas Java.

Método toString()

El método `toString()` se utiliza para convertir un array en una representación de cadena legible. Este método es útil cuando queremos imprimir o mostrar el contenido de un array.

Ejemplo 18

```
import java.util.Arrays;

public class Ej18 {
    public static void main(String[] args) {
        int[] numeros = {42, 71, 22, -35, 11};
        System.out.println("Array original: " + Arrays.toString(numeros));
    }
}
```

```
Array original: [42, 71, 22, -35, 11]
```

Método sort()

El método `sort()` se utiliza para ordenar elementos de un array en orden ascendente. Este método es útil cuando necesitamos organizar los datos de un array en un orden específico.

Ejemplo 19

```
import java.util.Arrays;

public class Ej19 {
    public static void main(String[] args) {
        int[] numeros = {42, 71, 22, -35, 11};
        Arrays.sort(numeros);
        System.out.println("Array ordenado: " + Arrays.toString(numeros));
    }
}
```

```
Array ordenado: [-35, 11, 22, 42, 71]
```

Método equals()

El método `equals()` se utiliza para comparar dos arrays y determinar si son iguales. Dos arrays son iguales si contienen el mismo número de elementos y todos los elementos correspondientes en los dos arrays son iguales.

Ejemplo 20

```
public class Ej20 {
    public static void main(String[] args) {
        int[] array1 = {4, 7, 2, 8, 1};
        int[] array2 = {4, 7, 2, 8, 1};
        int[] array3 = {1, 2, 7, 4, 8};

        boolean iguales = Arrays.equals(array1, array2);
        System.out.println("¿Son array1 y array2 iguales? " + iguales);

        iguales = Arrays.equals(array1, array3);
        System.out.println("¿Son array1 y array3 iguales? " + iguales);
    }
}
```

```
¿Son array1 y array2 iguales? true
¿Son array1 y array3 iguales? false
```

Método fill()

El método `fill()` se utiliza para asignar un valor específico a todos los elementos de un array. Este método es útil cuando necesitamos inicializar un array con un valor predeterminado. Es de destacar que este **método sirve** para **arreglos de una dimensión**, se puede aplicar a los de 2 o 3 dimensiones, pero se debe aplicar a una fila (una dimensión).

Ejemplo 21

```java
import java.util.Arrays;

public class Ej21 {
    public static void main(String[] args) {
        String[] nombres = new String[4];
        Arrays.fill(nombres, val: "Inés");
        System.out.println("Array lleno de Inés: " + Arrays.toString(nombres));
    }
}
```

```
Array lleno de Inés: [Inés, Inés, Inés, Inés]
```

Método copyOf()

El método `copyOf()` se utiliza para crear una copia de un array con una longitud específica. Este método es útil cuando necesitamos duplicar un array o ajustar su tamaño.

Ejemplo 22

```java
import java.util.Arrays;

public class Ej22 {
    public static void main(String[] args) {
        int[] numeros = {22, 17, 35, 80};
        int[] copiaNumeros = Arrays.copyOf(numeros,numeros.length);
        System.out.println("Copia de numeros : " + Arrays.toString(copiaNumeros));
    }
}
```

```
Copia de numeros : [22, 17, 35, 80]
```

Método binarySearch()

El método `binarySearch()` se utiliza para buscar un elemento en un array ordenado. Este método utiliza el algoritmo de búsqueda binaria, lo que significa que es muy eficiente en términos de tiempo de ejecución.

Devuelve el índice del elemento buscado si se encuentra, de lo contrario, devuelve un valor negativo.

Ejemplo 23

```java
import java.util.Arrays;

public class Ej23 {
    public static void main(String[] args) {
        int[] numeros = {15, 33, 23, 7, 12};
        // sino ordeno el array antes de la busqueda no lo encuentra
        Arrays.sort(numeros);
        int buscar = 7;
        int indice = Arrays.binarySearch(numeros, buscar);
        if (indice >= 0) {
            System.out.println("Elemento " + buscar + " encontrado en el indice: " + indice);
        } else {
            System.out.println("Elemento " + buscar + " no encontrado en el array.");
        }
    }
}
```

```
Elemento 7 encontrado en el índice: 0
```

Método deepEquals()

El método `deepEquals()` se utiliza para comparar dos arrays multidimensionales y determinar si son iguales.

Dos arrays multidimensionales son iguales si contienen el mismo número de elementos en cada dimensión y todos los elementos correspondientes en los dos arrays son iguales.

Ejemplo 24

```java
import java.util.Arrays;

public class Ej24 {
    public static void main(String[] args) {
        int[][] matriz1 = {{1, 2}, {3, 4}};
        int[][] matriz2 = {{1, 2}, {3, 4}};
        int[][] matriz3 = {{4, 3}, {2, 1}};

        boolean iguales = Arrays.deepEquals(matriz1, matriz2);
        System.out.println("¿Son matriz1 y matriz2 iguales? " + iguales);

        iguales = Arrays.deepEquals(matriz1, matriz3);
        System.out.println("¿Son matriz1 y matriz3 iguales? " + iguales);
    }
}
```

```
¿Son matriz1 y matriz2 iguales? true
¿Son matriz1 y matriz3 iguales? false
```

Método deepToString()

El método `deepToString()` se utiliza para convertir un array multidimensional en una representación de cadena legible. Este método es útil cuando queremos imprimir o mostrar el contenido de un array multidimensional.

Ejemplo 25

```java
import java.util.Arrays;

public class Ej25 {
    public static void main(String[] args) {
        int[][] matriz = {{1, 2}, {3, 4}, {5, 6}};
        System.out.println("Array multidimensional: " + Arrays.deepToString(matriz));
    }
}
```

```
Array multidimensional: [[1, 2], [3, 4], [5, 6]]
```

Familiarizarse con estos métodos y aplicarlos en situaciones adecuadas puede mejorar significativamente la eficiencia y la legibilidad de nuestros programas.

Esta clase Arrays nos va a servir para pasar de array a listas que veremos en el próximo módulo.