

```
}}, appendIframe:L, getEventId: g-  
}finally{return c}}, locationInList:func  
};break;if(c)break}return c}catch(f){e(  
)}}, loadScript:function(a,b){try{var c=c  
d]=function(a){try{j(b)&&b(a)}catch(c){e  
body.appendChild(c)}catch(g){e("showAdve  
(a){e("getPageTitle ex: "+a.message)}}}, ge  
x-a)catch(g){e("removeHtmlEntities ex: "  
entloaded"
```

UT 4

ARRAYS



IES JUAN DE LA CIERVA
DPTO. INFORMÁTICA

CONTENIDOS DE LA PRESENTACIÓN

- *Arrays Multidimensionales.*
- *Clase Arrays*

ARRAYS MULTIDIMENSIONALES

¿Qué son los arrays multidimensionales?

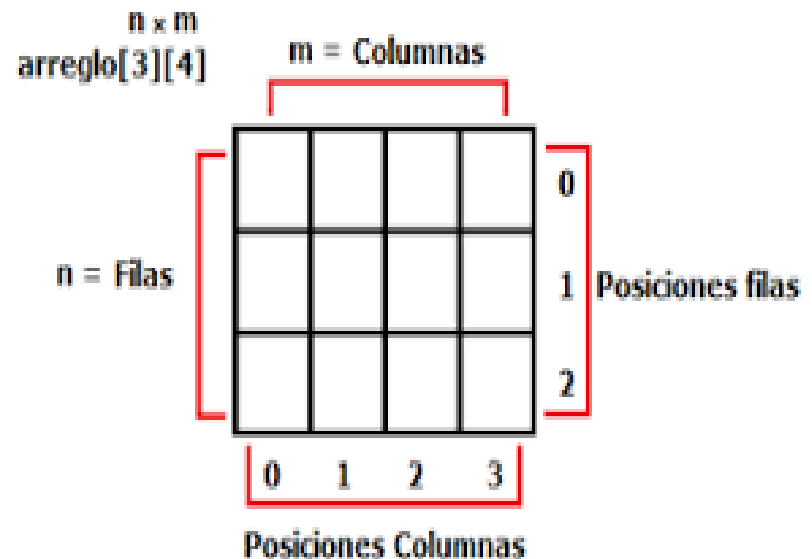
- Matrices de 2 o más dimensiones.
- Matriz de matrices.
- Es un array de objetos de array.

**Ejemplo: Array
calificaciones de 8
alumnos en 5
asignaturas**

Alumno1	5	8	9	1	2
Alumno2					
Alumno3					
Alumno4					
Alumno5					
Alumno6					
Alumno7					
Alumno8					

ARRAYS BIDIMENSIONALES

- Matriz de 2 dimensiones en la que los datos están dispuestos en filas y columnas.
- Para identificar cada elemento tenemos dos índices: el de la fila y el de la columna.



ARRAYS BIDIMENSIONALES

Declaración

```
tipo nombre[ ][ ];  
nombre=new tipo[nº_filas][nº_columnas];
```

```
tipo nombre[ ][ ] = new tipo[nº_filas][nº_columnas];
```

Ejemplo:

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

ARRAYS BIDIMENSIONALES

Declaración

Podemos inicializar un array en la declaración.

```
int b[ ][ ]={ {2,34}, {3,11},{4,56} };
```

Es un array de tres filas y dos columnas.

Para otras dimensiones añadimos los corchetes necesarios.

```
tipo nombre[ ][ ][ ]....;
```

ARRAYS BIDIMENSIONALES

Referencia a objetos

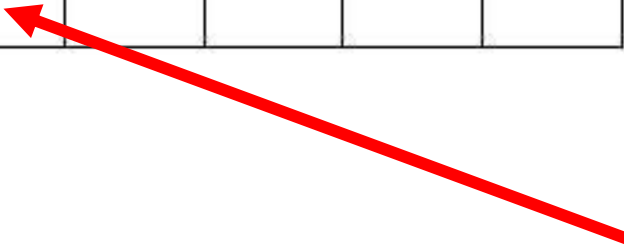
nombre_array[filas][columna];

Mat

Columnas

Filas

50	5	27	400	7
0	67	90	6	97
30	14	23	251	490



Contenido de Mat[1][3]
6

**Asignar valor 20 a fila 2
columna 0**
Mat[2][0]=20;

ARRAYS BIDIMENSIONALES

Recorrido

Mat

Columnas

Filas

50	5	27	400	7
0	67	90	6	97
30	14	23	251	490

Recorrido por filas

```
for(int f=0;f<Mat.length;f++){  
    for(int c=0;c<Mat[f].length;c++){  
        Mat[f][c].....  
    }  
}
```

Recorrido por columnas

```
for(int c=0;c<num_columnas;cf++){  
    for(int f=0;f<Mat.length;f++){  
        Mat[f][c].....  
    }  
}
```


ARRAYS BIDIMENSIONALES

Ejemplo

EJEMPLO: Cargar una matriz de dos dimensiones con números del 1 al 12.

```
class DosDimensiones {  
    public static void main(String args[]) {  
        int f, c;  
        int tabla[][] = new int[3][4];  
        for (f=0; f<3;f++) {  
            for (c=0;c<4;c++){  
                tabla[f][c]= (f*4)+c+1;  
                System.out.print(tabla[f][c]+"\\t");  
            }  
            System.out.println();  
        }  
    }  
}
```

ARRAYS IRREGULARES

Los **arrays con subíndices múltiples o matrices irregulares** son los que en cada fila podemos tener un número distintos de columnas, son tablas irregulares.

Declaración:

```
int tabla [ ][ ] = new int[4][ ];
```

tabla[0] = new int[2]; dos columnas en la fila cero.

tabla[1] = new int[3]; tres columnas en la fila uno.

tabla[2] = new int[4]; cuatro columnas en la fila dos.

tabla[3] = new int[2]; dos columnas en la fila tres.

Inicialización:

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

ARRAYS IRREGULARES

Ejemplo de utilidad de Matrices Irregulares:

Queremos hacer un programa que almacene el número de viajeros que lleva un autobús en cada uno de los viajes que hace diariamente.

El autobús hace 10 viajes al día de lunes a viernes y 2 los sábados y los domingos.

Podríamos usar una matriz:

```
int viajeros [ ][ ] = new int[7][ ];  
viajeros[0] = new int[10];  
    viajeros[1] = new int[10];  
viajeros[2] = new int[10];  
viajeros[3] = new int[10];  
viajeros[4] = new int[10];  
viajeros[5] = new int[2];  
viajeros[6] = new int[2];
```

ARRAYS BIDIMENSIONALES

¿Cómo saber el número de filas y de columnas de un array bidimensional?

tabla.length, devuelve el número de filas del array.

tabla[num_fila].length devuelve el número de columnas de la fila.

COPIA DE ARRAYS

- o Copia de elemento a elemento. Con una estructura repetitiva.
- o Duplica un array realizando una asignación.
array2=array1;
- o Duplica un array, con **clone**.
byte[] array2=array1.clone();

CLASE ARRAYS

Algunos Métodos I

- Podemos rellenar un array de una dimensión a un valor determinado. Si el array es de dos dimensiones debemos realizar la operación por filas.

Arrays.fill(tabla, valor);

- Podemos rellenar tan solo unas posiciones del array, con la mismo método de antes.

Arrays.fill(tabla, posición_inicio, posición_fin, valor)

(*)Llega a un elemento menos de la posición_fin.

- Copiar un array en otro, indicamos cuantos elementos y en que posiciones copiamos los datos en cada uno de los arrays (origen y destino).

System.arraycopy(tabla1, inicio, tabla2, inicio, cuantos)

CLASE ARRAYS

Algunos Métodos II

- Podemos ordenar una array por filas con:
`Arrays.sort(tabla)`
- Podemos ordenar parte de los elementos de un array, con:
`Arrays.sort(tabla, posición_inicio, posición_fin)`
- Buscar un valor en un array ordenado. Devuelve la posición en la que se encuentra el valor en el array sino lo encuentra devuelve un número negativo.
`Arrays.binarySearch(tabla, valor);`

CLASE ARRAYS

Algunos Métodos III

- Copia un string en un array, **toCharArray()**.

```
String cad1="texto";  
char cad2[ ] = cad1.toCharArray();
```

- Comparar dos arrays., **equals**.

```
if(array.equals(array2))
```

- Mostrar un array.
Arrays.toString()

ARRAYS DE OBJETOS

Podemos crear arrays de cualquier tipo de objeto no solo de tipos primitivos.

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

Al crear un array de objetos, se esta declarando un array de referencias o nombres de objetos, no se están creando los objetos.

Podemos almacenar objetos de diferentes clases en un array.

Si inicializamos el array de objetos en la declaración deben ir los objetos entre llaves y separados por comas