

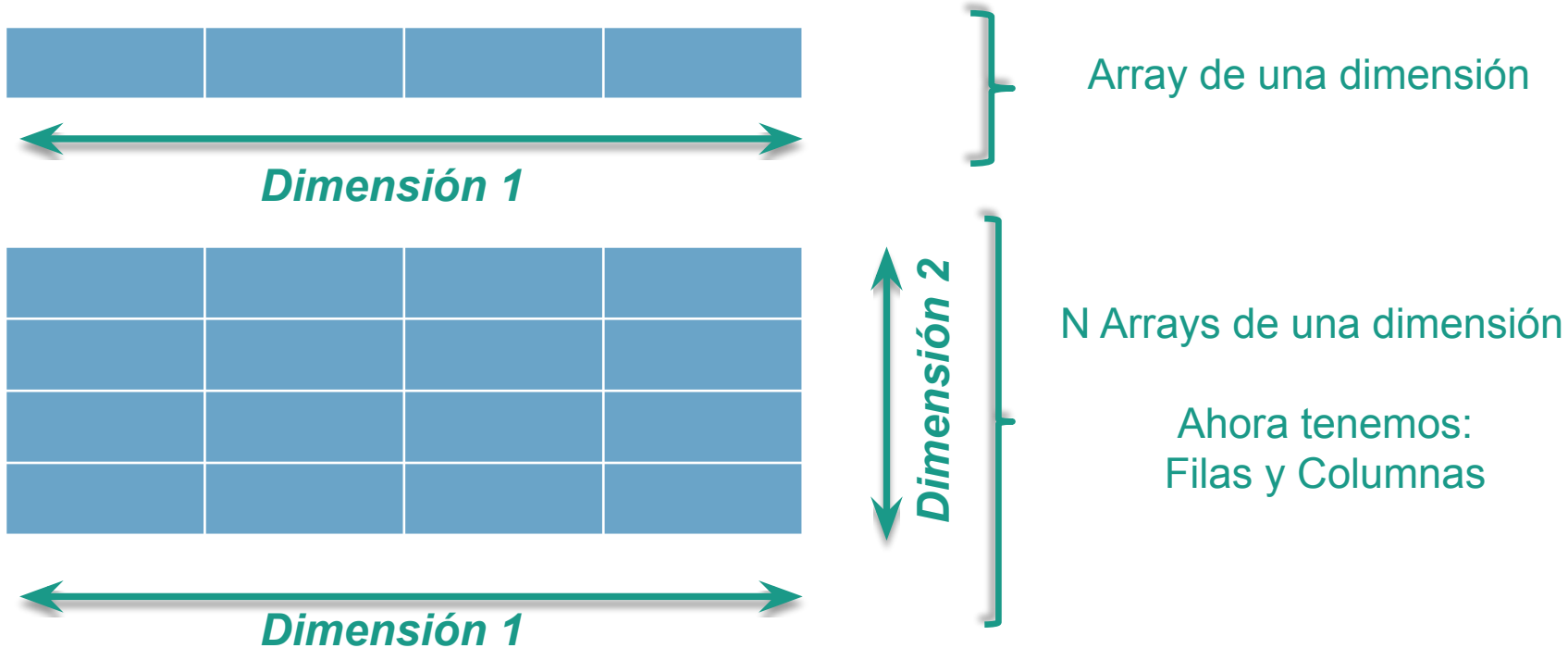


Programación 1

Matrices

Introducción


- Las matrices son arrays de 2 o más dimensiones



Partes de una matriz

- Por ej, una matriz de 2 dimensiones (con las que vamos a trabajar) con 16 enteros puede ser:

	Columna 0	Columna 1	Columna 2	Columna 3
Fila 0	1	5	2	34
Fila 1	45	3	11	46
Fila 2	76	66	21	6
Fila 3	53	43	42	13



Celda (2,3)

Declaración de una matriz

- La declaración tiene la forma:

`tipo_de_dato [][] nombre_matriz`

por ejemplo:

`int [][] matEntero` *//matEntero es una matriz de enteros*

- Para hacer espacio para los datos en la memoria se debe hacer la declaración:

`matEntero = new int [MAXFILAS][MAXCOLUMNAS]`

*//se supone que MAXFILAS y MAXCOLUMNAS son constantes que se definieron antes,
y son cualquier número entero mayor a 0*

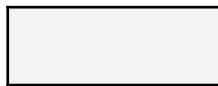
podríamos escribir todo junto:

`int [][] matEntero = new int [MAXFILAS][MAXCOLUMNAS]`

Declaración de una matriz

- Ej de declaración sin espacio para datos:

```
int [][] matEntero //matEntero es una matriz de enteros
```



matEntero

- Ej de declaración con espacio para datos:

```
final int MAXFILAS = 2;
```

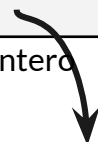
```
final int MAXCOLUMNS = 2;
```

```
...
```

```
int [][] matEntero = new int [MAXFILAS][MAXCOLUMNS];
```



matEntero



0	0
0	0

Declaración de una matriz

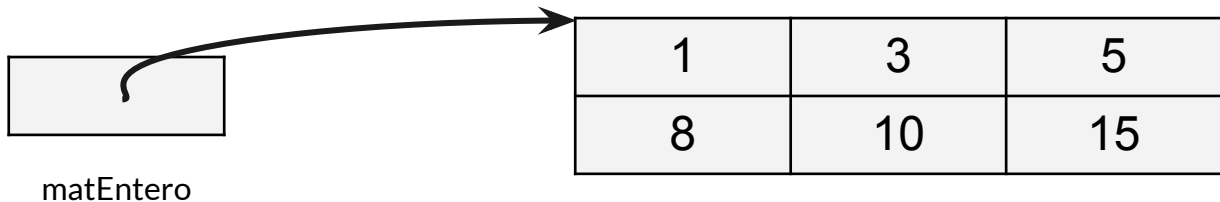
- Al igual que en arreglos, podemos hacer una declaración explícita a partir de los valores:

```
int [][] matEntero = { {1 , 3 , 5},  
                      {8 , 10 , 15}  
                      };
```

Las comas separan las filas

- ¿Qué tendríamos en memoria?

Una variable **matEntero** que apunta a un espacio de 2 dimensiones



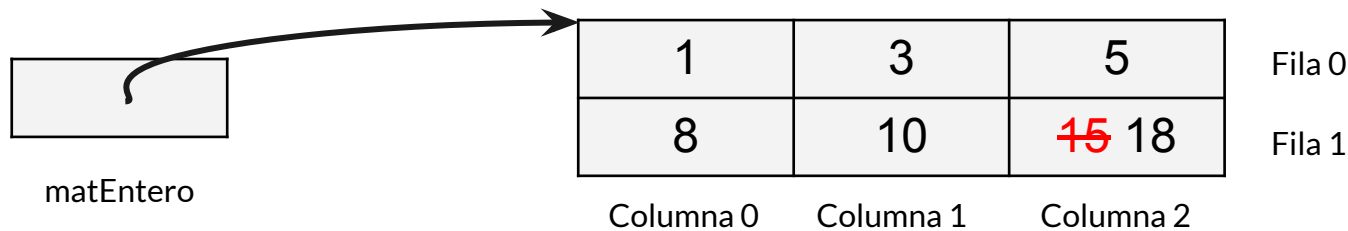
Acceso a una matriz

- Al igual que en arreglos, la posición comienza desde 0 a MAX-1, solo que ahora tenemos para recorrer filas y columnas:

```
int fila = 1;
```

```
int columna = 2;
```

```
matEntero [fila][columna] = 18 //Se reemplaza el valor previo por 18
```



¿Y si a columna le hubiésemos puesto 3, qué hubiese ocurrido?

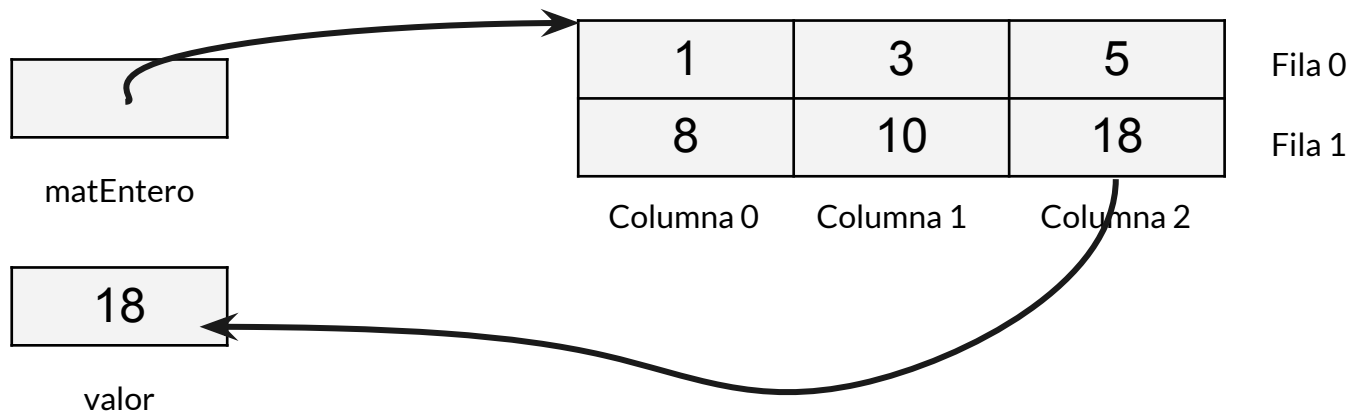
Acceso a una matriz

- Ej acceso:

```
int fila = 1;
```

```
int columna = 2;
```

```
int valor = matEntero[fila][columna]; //Se copia en la variable valor el número 18
```



Acceso a una matriz



Siempre debemos verificar que el acceso a una matriz
sea con variables que

estén dentro del espacio asignado a la dimensión que se quiera acceder

Pasaje por parámetros

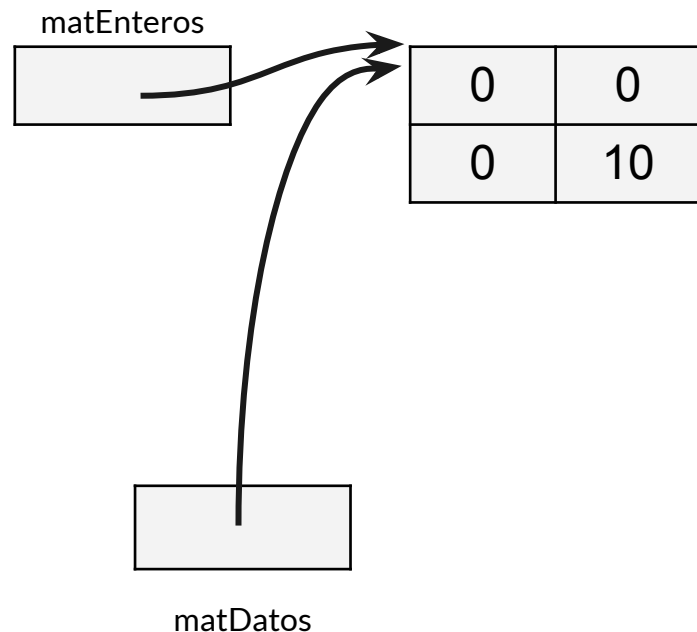


En el pasaje por parámetros, ocurre lo mismo que con arreglos: como son tipos NO PRIMITIVOS, el método que recibe una matriz por parámetro, apunta a la misma dirección de memoria en dónde están realmente los datos

Pasaje por parámetros

```
final int maxFil = 2; final int maxCol = 2;
public static void main(String[] args){
    ...
    int [][] matEnteros = new int[maxFil][maxCol];
    cargarRandom(matEnteros);
    ...
}

public static void cargarRandom(int [][] matDatos){
    ...
    matDatos[fil][col] = 10; //sup. que fil = 1 y col = 1
}
```



Métodos con matrices




- Java y otros lenguajes, permiten resolver una fila de la matriz como si fuera un arreglo
- Cada fila de la matriz es un arreglo del mismo tipo de la matriz y de tamaño MAXCOLUMNAS.
- Así, por ej si tenemos:

```
int[][] matEnt = new int [MAXFILAS][MAXCOLUMNAS];
```

 - Luego podríamos decir que:
matEnt[nroFila] es un arreglo de enteros, al que le podemos aplicar todos los métodos desarrollados anteriormente sobre arreglos.
- En recorridos por fila esto simplifica mucho la labor del programador.
- En recorridos por columna no. Es decir, en JAVA no se puede acceder a una columna, sin antes acceder a una fila


Métodos con matrices (sin arreglos)



```
public static void imprimir_promedios_filas (int[][] mat){
    for (int fila = 0 ; fila < MAXFILA; fila++){
        System.out.println("Promedio de la fila "+fila+" es "+promedio_fila(mat,fila));
    }
}
```

```
public static int promedio_fila (int[][] mat, int fila){
    int promedio;
    int suma = 0;
    for (int columna = 0 ; columna < MAXCOLUMNA; columna++) {
        suma+=mat[fila][columna];
    }
    promedio = suma/MAXCOLUMNA;
    return promedio;
}
```

Métodos con matrices (con arreglos)



```
public static void imprimir_promedios_filas (int[][] mat){  
    for (int fila = 0 ; fila < MAXFILA; fila++){  
        System.out.println("Promedio de la fila "+fila+" es "+promedio_arreglo(mat[fila]));  
    }  
}
```

```
public static int promedio_arreglo (int[] arr){  
    int promedio;  
    int suma = 0;  
    for (int columna = 0 ; columna < MAXCOLUMNA; columna++) {  
        suma+=arr[columna];  
    }  
    promedio = suma/MAXCOLUMNA;  
    return promedio;  
}
```

Métodos con matrices

- Como se mencionó anteriormente, en JAVA no se puede acceder primero a una columna
- Por ej: Hacer un programa que dada una matriz de enteros de tamaño 4x5 (precargada), imprima por pantalla el promedio de cada una de sus columnas.

```
public static void imprimir_promedios_matriz (int[][] mat){
    int promedio;
    for (int col = 0 ; col < MAXCOLUMNA; col++){
        System.out.println("Promedio de la columna "+col+" es "+obtener_promedio_columna(mat,col));
    }
}

public static int obtener_promedio_columna(int[][] mat,int col){
    int promedio; int suma = 0;
    for (int fila = 0 ; fila < MAXFILA; fila++) {
        suma+=mat[fila][col];
    }
    promedio = suma/MAXFILA;
    return promedio;
}
```

Buscar un elemento 1/2

//Hacer un programa que dado una matriz de enteros de tamaño 4*5, encuentre la posición
//fila,columna de un número entero ingresado por el usuario. Si existe, muestre esa posición por
//pantalla, o indique que no existe.

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class Clase_6_Ejemplo_1 {
    public final static int MAXFILAS = 4, MAXCOLUMNA = 5;
    public static void main(String[] args) {
        int numero;
        int [][] matint = new int [MAXFILAS][MAXCOLUMNA];
        cargar_matriz_aleatorio_int(matint);
        BufferedReader entrada = new BufferedReader(new InputStreamReader(System.in));
        try {
            System.out.println("ingrese un numero entero: ");
            numero = Integer.valueOf(entrada.readLine());
            imprimir_fila_columna_matriz(matint, numero);
        }
        catch (Exception exc){
            System.out.println(exc);
        }
    }
}
```


Buscar un elemento 2/2

```
public static void imprimir_fila_columna_matriz(int[][] mat, int numero){
    int fila = 0;
    int columna = MAXCOLUMNA;
    while ((fila < MAXFILA) && (columna == MAXCOLUMNA)){
        columna = obtener_pos_arreglo(mat[fila],numero);
        if (columna == MAXCOLUMNA){
            fila++;
        }
    }
    if ((fila < MAXFILA) && (columna < MAXCOLUMNA)){
        System.out.println(numero+" se encuentra en "+fila+" y "+columna);
    }
    else {
        System.out.println(numero+" no se encuentra en la matriz");
    }
}

public static int obtener_pos_arreglo(int [] arr, int numero){
    int posicion = 0;
    while ((posicion < MAXCOLUMNA) && (arr[posicion] != numero)){
        posicion++;
    }
    return posicion;
}

}
} // fin del class
```

Ejemplo corrimiento



Hacer un programa que dado una matriz de enteros de tamaño 4x5 que se encuentra precargada, solicite al usuario una posición fila, columna y realice un corrimiento a derecha. Además, imprima la matriz antes y después del corrimiento.

```

import java.io.BufferedReader;
import java.io.InputStreamReader;
public class EjemploMat {
    public static final int MAXFILA = 4;
    public static final int MAXCOLUMNA = 5;
    public static void main (String[] args) {
        int [][] matint;
        int fila, columna;
        matint = new int[MAXFILA][MAXCOLUMNA];
        cargar_matriz_aleatorio_int(matint);
        imprimir_matriz_int(matint);
        fila = LeerFilaPorTeclado();
        columna = LeerColumnaPorTeclado();
        if ((0<=fila)&&(fila < MAXFILA)&&(0<=columna)&&(columna < MAXCOLUMNA)){
            corrimiento_der_fila_columna(matint[fila],columna);
            imprimir_matriz_int(matint);
        }
    }

    public static void corrimiento_der_fila_columna(int [] arrenteros, int pos){
        int indice = MAXCOLUMNA-1;
        while (indice > pos){
            arrenteros[indice] = arrenteros[indice-1];
            indice--;
        }
    }
}

```

Ejemplo de una corrida:

3	8	4	7	5
4	9	6	1	7
1	1	1	6	2
5	5	7	1	2

Ingrese una fila:

1

Ingrese una columna:

2

3	8	4	7	5
4	9	6	6	1
1	1	1	6	2
5	5	7	1	2

Matrices - Ordenamientos y secuencias

- Una matriz puede estar ordenada:
 - En cada una de sus filas
 - En cada una de sus columnas
 - Toda la matriz, con algún sentido particular (no se verá en esta cursada)
- En la cátedra trabajaremos con matrices ordenadas por filas y por columnas.

Ordenado por fila:

4	5	6	9	9
1	4	5	7	7
1	1	1	6	8
5	5	7	8	9

Ordenado por columna:

4	5	2	1	5
5	6	4	3	5
7	6	8	9	6
9	7	9	9	6

Matrices - Ordenamientos y secuencias



- Para ordenar una matriz por fila o columna, vamos a aplicar las mismas técnicas/algoritmos que aplicamos para arreglos:
 - Selección, inserción, burbujeo.
- Para ordenar por filas directamente aplicamos el mismo método de arreglo.
- Para ordenar por columnas, no se puede reutilizar los métodos de arreglos

//Ejemplo Ordenar por Columna con Selección

```
import java.io.BufferedReader;
import java.io.InputStreamReader;

public class EjemploMat {
    public static final int MAXFILA = 4;
    public static final int MAXCOLUMNA = 5;
    public static void main (String[] args) {
        int [][] matint;
        int columna;
        matint = new int[MAXFILA][MAXCOLUMNA];
        cargar_matriz_aleatorio_int(matint);
        imprimir_matriz_int(matint);
        columna = LeerColumnaPorTeclado();
        if ((0<=columna)&&(columna < MAXCOLUMNA)){
            ordenar_matriz_columna_seleccion(matint,columna);
            imprimir_matriz_int(matint);
        }
    }

    public static void ordenar_matriz_columna_seleccion(int [][] mat, int col){
        int pos_menor, tmp;
        for (int i = 0; i < MAXFILA; i++) {
            pos_menor = i;
            for (int j = i + 1; j < MAXFILA; j++){
                if (mat[j][col] < mat[pos_menor][col]) {
                    pos_menor = j;
                }
            }
            if (pos_menor != i){
                tmp = mat[i][col];
                mat[i][col] = mat[pos_menor][col];
                mat[pos_menor][col] = tmp;
            }
        }
    }
}
```

Ejemplo de una corrida:

9	5	3	5	2
3	9	9	9	6
5	2	3	6	9
8	1	3	9	7

Ingrese una columna:

1

9	1	3	5	2
3	2	9	9	6
5	5	3	6	9
8	9	3	9	7

Matrices de Secuencias

- Al igual que con los arreglos, las matrices de secuencias también sirven para **representar estructuras dentro de una estructura**. Por ejemplo, patrones de colores dentro de una foto (la foto es una matriz de números que hacen referencia a colores).
- En este caso solo vamos a trabajar con matrices de secuencias de números enteros y matrices de secuencias de caracteres, considerando que estas secuencias solo aparecen en las filas (**no habrá secuencias por columnas**).



Matrices de Secuencias



- Los métodos para trabajar con estas estructuras implican:
 - Recorrer secuencias,
 - Procesar las secuencias,
 - Incorporar secuencias,
 - Eliminar secuencias.
- Como con arreglos, vamos a partir de una solución para cargar de forma aleatoria una matriz de secuencias por fila, que nos va a permitir utilizarlo para probar nuestros ejercicios.
- A continuación se dejan ejemplos de códigos (carga e impresión) para reutilizar. Los mismos se basan en las soluciones de arreglos.


```

import java.io.BufferedReader;
import java.io.InputStreamReader;

public class EjemploMat {

    public static final int MAXFILA = 4;
    public static final int MAXCOLUMNA = 20;

    public static void main(String[] args) {
        char [][] matchar;
        int [][] matint;
        matchar = new char[MAXFILA][MAXCOLUMNA];
        matint = new int[MAXFILA][MAXCOLUMNA];
        cargar_matriz_aleatorio_secuencias_char(matchar);
        imprimir_matriz_char(matchar);
        cargar_matriz_aleatorio_secuencias_int(matint);
        imprimir_matriz_int(matint);
    }

    public static void imprimir_matriz_char(char [][] mat){
        for (int fila = 0; fila < MAXFILA; fila++){
            imprimir_arreglo_char(mat[fila]);
        }
    }

    public static void imprimir_matriz_int(int [][] mat){
        for (int fila = 0; fila < MAXFILA; fila++){
            imprimir_arreglo_int(mat[fila]);
        }
    }
}

```

```

public static void cargar_matriz_aleatorio_secuencias_int(int [][] mat){
    for (int fila = 0; fila < MAXFILA; fila++){
        cargar_arreglo_aleatorio_int(mat[fila]);
    }
    System.out.println("");
}

public static void cargar_matriz_aleatorio_secuencias_char(char [][] mat){
    for (int fila = 0; fila < MAXFILA; fila++){
        cargar_arreglo_aleatorio_char(mat[fila]);
    }
    System.out.println("");
}
}

```

Ejemplo de una corrida:

```

| | | |f| |w|y|i|x|l|p|a|r| |o| | | | | |
| |j| |w| |o| | |x|g| |t|g|e|v| |q|d| | |
| |m|y| |c| | |l| |e|d|q| | | |v| |x|f| |

```

```

|0|6|6|0|4|4|0|0|3|9|0|8|5|7|0|6|6|0|1|0|
|0|3|0|5|0|9|0|5|0|0|0|0|3|0|9|3|4|0|0|0|
|0|2|0|0|0|7|0|0|7|0|4|0|0|0|2|0|8|4|5|0|
|0|5|0|3|1|9|0|5|7|0|0|4|0|0|0|9|9|0|0|0|

```

Algunos tips

- Cuando trabajo sobre una matriz en un método no se debe retornar la misma que se pasa por parámetro ya que se modifica esa misma. Si se puede devolver otra matriz con resultados calculados a partir de esa original.
- El pasaje de parámetros es por copia y cuando trabajo con tipos de referencia lo que se copia es la dirección de memoria no el valor.
- No se puede acceder a una columna sin antes acceder a una fila.
- Siempre verificar los límites tanto de filas como de columnas para acceder a posiciones válidas.
- Siempre que quiero insertar un elemento en una matriz, desplazar a derecha en el lugar que debe ir dentro de la fila. Si es una secuencia se debe desplazar tantas veces como el tamaño de la secuencia que se quiere insertar.
- Siempre que quiero eliminar un elemento de una matriz se debe desplazar a izquierda y queda duplicado el último elemento dentro de la fila. Si es una secuencia se debe desplazar tantas veces como el tamaño de la secuencia y quedará duplicado n veces el último elemento.
- Siempre que tengo una estructura ordenada tengo que aprovechar esa característica para cuando quiero trabajar con ella. No es recomendable insertar y luego ordenar porque se desperdician recursos computacionales.
- Las secuencias están separadas por “separadores” que pueden variar dependiendo de la aplicación.