

JAVA

TEMA 06:

ARRAYS



ÍNDICE

1. Introducción
2. Vectores
3. Matrices
4. Arrays multidimensionales
5. Arrays y el método main
6. Ejercicios de consolidación



JAVA

ARRAYS

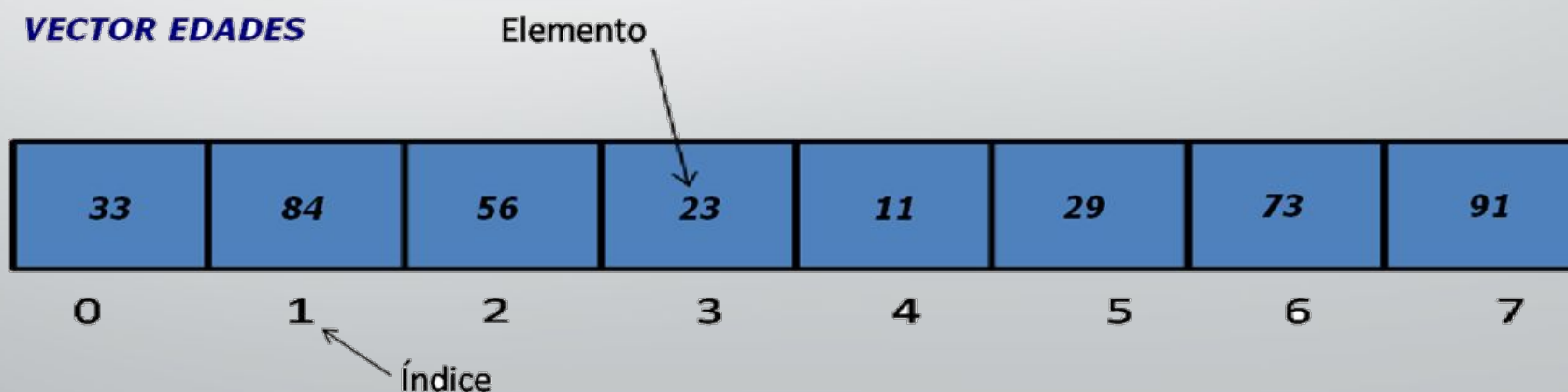
1. Introducción



1.- Introducción

- Un array es un identificador que referencia un conjunto de datos del mismo tipo.
- Podremos referenciar a un dato o a otro del conjunto mediante un índice, que comenzará por el valor 0 y será un valor entero y positivo.

- Ejemplo:



1.- Introducción

- Los arrays podrán ser de 3 tipos:

- **Vectores** (Array de una dimensión)

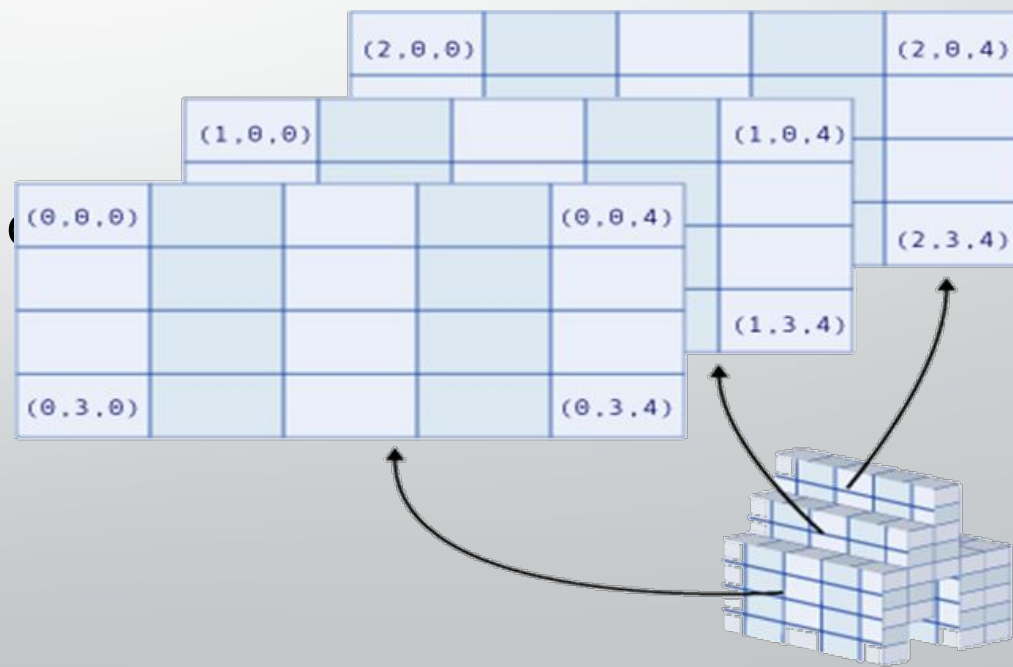
- **Matrices** (Array de 2 dimensiones)

- **Multi-dimensionales** (Array de 3 dimensiones)

0	20	10	9	3	1
1	0	2	8	55	4
2	99	28	30	91	6
3	5	12	13	14	7
4	22	39	21	40	67

3 DIMENSIONES □

□ 2 DIMENSIONES



JAVA

ARRAYS



2. Vectores

2.- Vectores

- Un **vector** es un array unidimensional, es decir, sólo utiliza un índice para referenciar a cada uno de los elementos.
- Su definición será de la siguiente manera:

tipo [] nombre; *por ejemplo: int [] edades;*

- **tipo:** Podrá ser cualquiera de los conocidos (int, char, double, float, String...e incluso un objeto)
- **nombre:** Un identificador representativo para el vector.

2.- Vectores

- Un array en JAVA no deja de ser un objeto. Por ello deberemos instanciarlo, esto es, crearlo y reservar memoria para él:

nombre = new **tipo** [**tamaño**]; *por ejemplo: **edades** = new int [8];*

- **tipo**: Podrá ser cualquiera de los conocidos (int, char, double, float, String...e incluso un objeto)
- **nombre**: Un identificador representativo para el vector.
- **tamaño**: Número de celdas del vector.

2.- Vectores

- Como ya hemos visto con los objetos, lo más habitual será realizar la definición y la instanciación juntas:

tipo [] **nombre** = new **tipo** [**tamaño**];

- tipo: Podrá ser cualquiera de los conocidos (int, char, double, float...)
- nombre: Un identificador representativo para el vector.
- tamaño: número de elementos del vector.

Ejemplo: `int [] calificaciones = new int [25]`

`// He creado el array calificaciones que albergará 25 enteros`

2.- Vectores

• Ejemplo:

```
package vector1;
import java.util.Scanner;
/**
 * @author Oscar Laguna Garcia
 */
public class Vector1 {

    public static void rellenaVector(int vector[]) {
        Scanner entrada = new Scanner(System.in);
        int i;
        System.out.println("Vamos a rellenar un array de 5 posiciones");
        for (i = 0; i < 5; i++) {
            System.out.print("Introduce un valor para la posicion "+i+": ");
            vector[i]=entrada.nextInt();
        }
    }

    public static void muestraVector(int vector[]) {
        int i;
        for (i = 0; i < 5; i++) {
            System.out.println("El valor guardado en la posicion "+i+" es de: "+vector[i]);
        }
    }

    public static void main(String[] args) {
        int[] vector= new int [5];
        rellenaVector(vector);
        muestraVector(vector);
    }
}
```

2.- Vectores

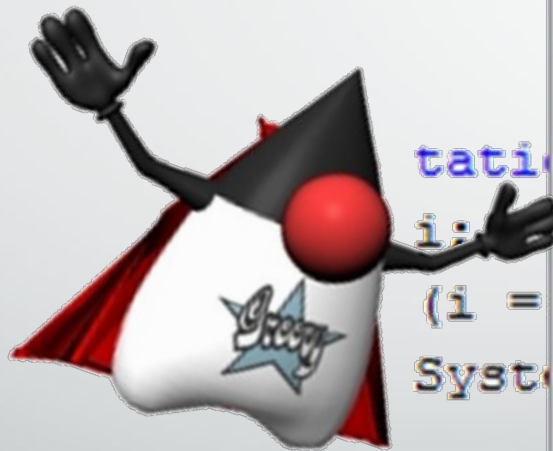
- Un vector también podemos inicializarlo (esto es, asignarle valores) en el momento de declararlo. Si lo hacemos así ya no es necesario indicar el tamaño. Ejemplo:

```
String [] dias = {"Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo"};
```



2.- Vectores

- Como los arrays son objetos, también tienen, por defecto, una serie de métodos:



vector.

<input type="checkbox"/> length	int
<input type="radio"/> clone()	Object
<input type="radio"/> equals(Object obj)	boolean
<input type="radio"/> getClass()	Class<?>
<input type="radio"/> hashCode()	int
<input type="radio"/> notify()	void
<input type="radio"/> notifyAll()	void
<input type="radio"/> toString()	String
<input type="radio"/> wait()	void
<input type="radio"/> wait(long timeout)	void
<input type="radio"/> wait(long timeout, int nanos)	void

2.- Vectores

- Además, dispone del atributo **length**, que nos indica el número de elementos de nuestro array.
- Saber las posiciones del array es muy útil, para recorrerlo con un bucle for y, sobretodo, cuando en vez del programador, es el usuario en tiempo de ejecución quien determina el tamaño del array.
- En el siguiente ejemplo será el usuario quien elija el tamaño del vector, por lo que siempre controlaremos que nos introduzca un número entero y que las posiciones sean un número mayor que 0:

2.- Vectores

```
package vector2;
import java.util.Scanner;
/**
 * @author Oscar Laguna Garcia
 */
public class Vector2 {

    public static int elegirLongitud() {
        Scanner entrada = new Scanner(System.in);
        int longitud=0;
        do{
            System.out.print("Por favor, introduzca la longitud del vector: ");

            try{
                longitud = entrada.nextInt();
            }
            catch (Exception e){
                System.out.println(e);
                System.out.println("La longitud del vector debe de ser un entero");
            }
            finally{
                entrada.nextLine();
            }

            if (longitud<1){
                System.out.println("La longitud del vector debe de mayor que cero");
            }

        }while(longitud<1);

        return longitud;
    }
}
```


2.- Vectores

```
public static void rellenarVector(int vector[]){
    Scanner entrada = new Scanner(System.in);
    int i;
    System.out.println("Vamos a rellenar el array de "+vector.length+" posiciones");
    for (i = 0; i < vector.length; i++) {
        System.out.print("Introduce un valor para la posicion "+i+": ");
        vector[i]=entrada.nextInt();
    }
}

public static void mostrarVector(int vector[]){
    int i;
    System.out.println("\n -- Vector Resultante -- ");
    for (i = 0; i < vector.length; i++) {
        System.out.println("El valor guardado en la posicion "+i+" es de: "+vector[i]);
    }
}

public static void main(String[] args) {
    int[] vector= new int [elegirLongitud()];
    rellenarVector(vector);
    mostrarVector(vector);
}
```

EJERCICIOS

- **Ejercicio 01.- (OPTATIVO)** Escribir un programa que contenga un método que pida al usuario 10 números enteros y los guarde en un array unidimensional. Luego, otro método, mostrará solo los números pares que contiene el array.

EJERCICIOS

- **Ejercicio 02.- (OPTATIVO)** Realizar un programa que lea 7 números enteros y los introduzca en un array unidimensional. Luego muestras el array, intercambias los números que se encuentren en la 2ª y 4ª posición, y muestras el nuevo array por pantalla.
- Utiliza al menos 3 métodos: uno para introducir los datos, otro para mostrar los datos y otro para intercambiar los datos.

EJERCICIOS

- **Ejercicio 03.- (OBLIGATORIO)** Escribir un programa en JAVA que contenga un método que rellena un array unidimensional (vector) de enteros aleatorios entre el 1 y el 6, y luego, otro método, lo muestre por pantalla.
- El programa también tendrá un método donde el usuario elegirá la longitud del array entre 1 y 10. Este método también controla que el usuario nos introduzca un número entero y no una letra (mediante excepciones). En caso de que el usuario introduzca el dato incorrecto se lo volveremos a pedir las veces que hagan falta.

JAVA

ARRAYS



3. Matrices

3.- Matrices

- Una matriz es un array bidimensional.
- Se definen igual que los vectores excepto que se requiere un índice por cada dimensión.
- Su declaración + instanciación es la siguiente:

tipo nombre [] [] = new **tipo** [filas][columnas];

- Una matriz bidimensional se podría representar gráficamente como una tabla con filas y columnas, donde el primer índice controlaría las filas y el segundo las columnas.

3.- Matrices

- Ejemplo de un array bidimensional [5][12]:

ARRAY CON LAS VENTAS DE COCHES EN PLASENCIA

	E N E R O	F E B R E R O	M A R Z O	A B R I L	M A Y O	J U N I O	J U L I O	A G O S T O	S E P T I E M B R E	O C T U B R E	N O V I E M B R E	D I C I E M B R E
2011	60 (0,0)	30 (0,1)	50 (0,2)	60 (0,3)	30 (0,4)	50 (0,5)	40 (0,6)	50 (0,7)	80 (0,8)	30 (0,9)	90 (0,10)	70 (0,11)
2012	10 (1,0)	90 (1,1)	60 (1,2)	50 (1,3)	30 (1,4)	40 (1,5)	40 (1,6)	70 (1,7)	10 (1,8)	90 (1,9)	60 (1,10)	30 (1,11)
2013	60 (2,0)	40 (2,1)	80 (2,2)	90 (2,3)	10 (2,4)	40 (2,5)	20 (2,6)	30 (2,7)	80 (2,8)	20 (2,9)	40 (2,10)	50 (2,11)
2014	70 (3,0)	70 (3,1)	20 (3,2)	10 (3,3)	50 (3,4)	60 (3,5)	70 (3,6)	10 (3,7)	90 (3,8)	60 (3,9)	50 (3,10)	40 (3,11)
2015	20 (4,0)	50 (4,1)	10 (4,2)	70 (4,3)	90 (4,4)	80 (4,5)	20 (4,6)	90 (4,7)	60 (4,8)	50 (4,9)	20 (4,10)	60 (4,11)

PROGRAMACIÓN

3.- Matrices

- Ejemplo de como rellenar y mostrar un array de 4 filas y 3 columnas:

```
package array1;
import java.util.Scanner;
/**
 * @author Oscar Laguna García
 */
public class Array1 {

    public static void rellenaArray(int array[][]){
        Scanner entrada = new Scanner(System.in);
        int i,j;
        System.out.println("Vamos a rellenar un array de 4x3");
        for (i = 0; i < 4; i++) {
            for (j = 0; j < 3; j++) {
                System.out.print("Introduce un valor para la posicion ["+i+j+"] : ");
                array[i][j]=entrada.nextInt();
            }
        }
    }

    public static void muestraArray(int array[][]){
        int i,j;
        for (i = 0; i < 4; i++) {
            for (j = 0; j < 3; j++) {
                System.out.print(" ["+i+j+"]-->" +array[i][j]);
            }
            System.out.print("\n");
        }
    }

    public static void main(String[] args) {
        int[][] array = new int [4][3]; //Array de 4 filas y 3 columnas
        rellenaArray(array);
        muestraArray(array);
    }
}
```

3.- Matrices

- Ejemplo de como rellenar y mostrar un array de 4 filas y 3 columnas utilizando el método length:

```
package arraysbidimensionales;
import java.util.Scanner;
/**
 * @author OLG
 */
public class ArraysBidimensionales {

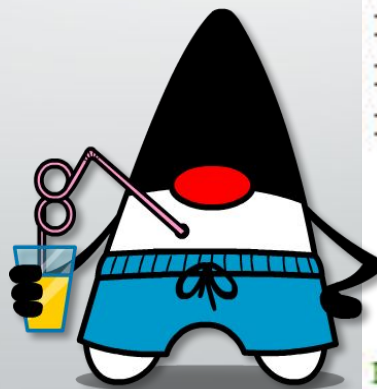
    public static void rellenarArray(int array[][]) {
        Scanner entrada = new Scanner(System.in);
        System.out.println("Vamos a rellenar un array de 4x3");
        for (int i = 0; i < array.length; i++) {
            for (int j = 0; j < array[i].length; j++) {
                System.out.print("Introduce un valor para la posición [" + i + j + "] :");
                array[i][j] = entrada.nextInt();
            }
        }
    }

    public static void mostrarArray(int array[][]) {
        for (int i = 0; i < array.length; i++) {
            for (int j = 0; j < array[i].length; j++) {
                System.out.print(" [" + i + j + "]-->" + array[i][j]);
            }
            System.out.print("\n");
        }
    }

    public static void main(String[] args) {
        int[][] array = new int[4][3];
        rellenarArray(array);
        mostrarArray(array);
    }
}
```

3.- Matrices

- Resultado de la ejecución:



```
Vamos a rellenar un array de 4x3
Introduce un valor para la posicion [00] : 2
Introduce un valor para la posicion [01] : 5
Introduce un valor para la posicion [02] : 7
Introduce un valor para la posicion [10] : 9
Introduce un valor para la posicion [11] : 3
Introduce un valor para la posicion [12] : 1
Introduce un valor para la posicion [20] : 6
Introduce un valor para la posicion [21] : 3
Introduce un valor para la posicion [22] : 7
Introduce un valor para la posicion [30] : 8
Introduce un valor para la posicion [31] : 5
Introduce un valor para la posicion [32] : 2

[00]-->2   [01]-->5   [02]-->7
[10]-->9   [11]-->3   [12]-->1
[20]-->6   [21]-->3   [22]-->7
[30]-->8   [31]-->5   [32]-->2

BUILD SUCCESSFUL (total time: 17 seconds)
```


3.- Matrices

- Ejemplo de como **rellenar** un array de 4 filas y 3 columnas y luego **mostrarlo al revés**. Bastará con cambiar el orden de los bucles for:

```
package probandofor;
import java.util.Scanner;
/**
 * @author OLG
 */
public class ProbandoFor {

    public static void rellenaArray(int array[][]) {
        Scanner entrada = new Scanner(System.in);
        System.out.println("Vamos a rellenar un array de 4x3");
        for (int i = 0; i < array.length; i++) { //i<4
            for (int j = 0; j < array[i].length; j++) { //j<3
                System.out.print("Introduce un valor para la posicion [" + i + j + "] : ");
                array[i][j] = entrada.nextInt();
            }
        }
    }

    public static void muestraArrayAlreves(int array[][]) {
        for (int j = 0; j < 3; j++) {
            for (int i = 0; i < 4; i++) {
                System.out.print(" [" + i + j + "]-->" + array[i][j]);
            }
            System.out.println("\n");
        }
    }

    public static void main(String[] args) {
        int[][] array = new int[4][3];
        rellenaArray(array);
        muestraArrayAlreves(array);
    }
}
```

3.- Matrices

- Resultado de la ejecución:



```
Vamos a rellenar un array de 4x3
Introduce un valor para la posicion [00] : 2
Introduce un valor para la posicion [01] : 5
Introduce un valor para la posicion [02] : 7
Introduce un valor para la posicion [10] : 9
Introduce un valor para la posicion [11] : 3
Introduce un valor para la posicion [12] : 1
Introduce un valor para la posicion [20] : 6
Introduce un valor para la posicion [21] : 3
Introduce un valor para la posicion [22] : 7
Introduce un valor para la posicion [30] : 8
Introduce un valor para la posicion [31] : 5
Introduce un valor para la posicion [32] : 2
[00]-->2 [10]-->9 [20]-->6 [30]-->8
[01]-->5 [11]-->3 [21]-->3 [31]-->5
[02]-->7 [12]-->1 [22]-->7 [32]-->2
BUILD SUCCESSFUL (total time: 21 seconds)
```


3.- Matrices

- Otra forma de hacer el método **mostrar al revés**, utilizando el método *length*, sería así:

```
public static void muestraArrayAlreves(int array[][]) {
    //for (int j = 0; j < 3; j++) {
    //for (int i = 0; i < 4; i++) {
    for (int j = 0; j < array[0].length; j++) {
        for (int i = 0; i < array.length; i++) {
            System.out.print(" [" + i + j + "]-->" + array[i][j]);
        }
        System.out.print("\n");
    }
}
```

3.- Matrices

- Una matriz también podemos inicializarla cuando la declaramos. Habrá que tener en cuenta el orden en el que los valores son asignados a los elementos de la matriz.
- Ejemplo: `int [] [] numeros = {{1,2,3},{4,5,6},{7,8,9},{10,11,12}};`

Quedarían asignados de la siguiente manera:

`numeros[0][0]=1 numeros[0][1]=2 numeros[0][2]=3`

`numeros[1][0]=4 numeros[1][1]=5 numeros[1][2]=6`

`numeros[2][0]=7 numeros[2][1]=8 numeros[2][2]=9`

`numeros[3][0]=10 numeros[3][1]=11 numeros[3][2]=12`

EJERCICIOS

- **Ejercicio 04.- (OPTATIVO)** Escribir un programa que contenga un método que guarde en un array bidimensional de 4×3 12 números aleatorios. Los números aleatorios estarán comprendidos entre el número 100 y el número 200. Luego, otro método, mostrará solo los números pares que contiene el array.

EJERCICIOS

- **Ejercicio 05.- (OBLIGATORIO)** Realiza un programa en JAVA que calcule el mayor, el menor y la suma de todos los elementos de un array bidimensional de 4x2 números enteros.
- El programa ejecutará un método en el que el usuario introduzca los valores, luego ejecutará otro método que visualizará los elementos del array de forma atractiva y, por último, se mostrará el mayor, el menor y la suma de todos los elementos (otros 3 métodos independientes)

JAVA

ARRAYS

4. Arrays multidimensionales



4.-Arrays multidimensionales

- Son aquellos arrays que tienen 3 o más dimensiones.
- Declaración + Instanciación:

tipo nombre[][]...[] = new tipo [t_dim1]...[t_dimN];

- Ejemplo: `int [][][] array = new int [3][5][4];`

Diagram illustrating a 3D array structure for student exam results.

alumno

alumno[1]	Fidel
alumno[2]	Ely
alumno[3]	Juan
alumno[4]	Daniel
alumno[5]	Paty

examen

	Examen1	Examen2	Examen3	Promedio
Materia3	89	88	85	87.33
Materia2	100	90	98	96
Material	100	58	89	82.33
	90	70	98	86
	100	95	98	97.67

Arrows indicate the mapping from the 3D array structure to the 2D table representation.

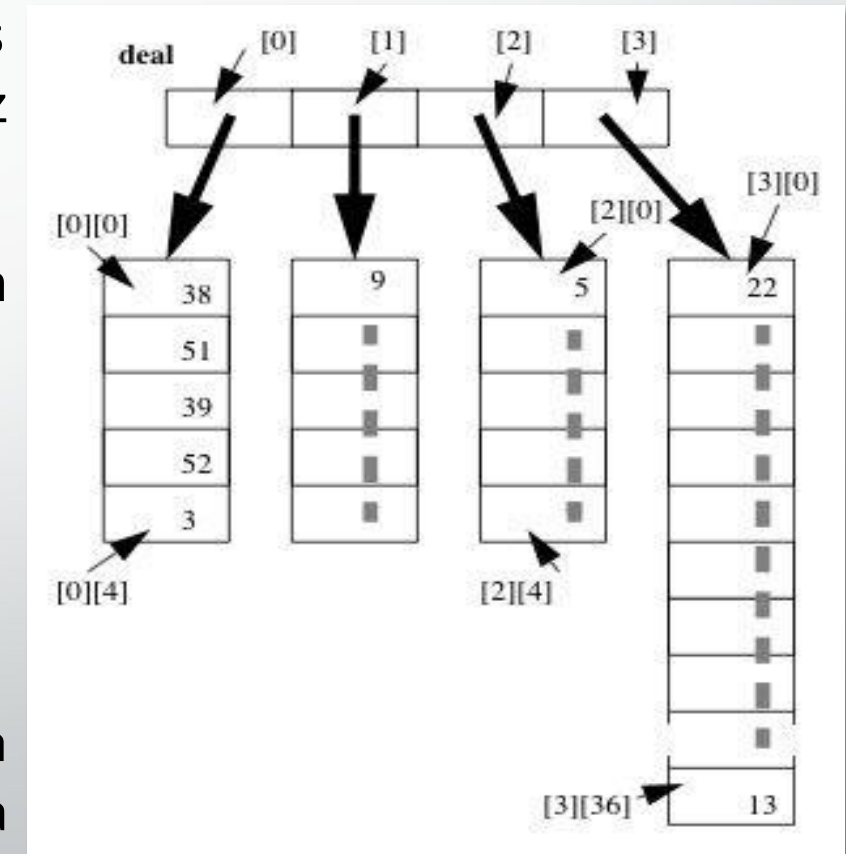
4.-Arrays multidimensionales

- También se consideran arrays multidimensionales aquellos arrays que a su vez contienen otro array.
- Un ejemplo de Declaración + Instanciación de un array bidimensional de arrays unidimensionales:

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

```
tabla [0] = new int [5];
```

```
tabla [1] = new int [5];...
```
- Fíjate que solo es obligatoria declarar la primera dimensión. Esto me van a permitir que cada fila tenga un número distinto de columnas.



4.-Arrays multidimensionales

- Para recorrer los Arrays multidimensionales lo podremos hacer de la siguiente forma:

```
for (int i=0;i<array.length; i++){
    for (int j=0; j<array[i].length; j++){
        .... operaciones ....
    }
}
```

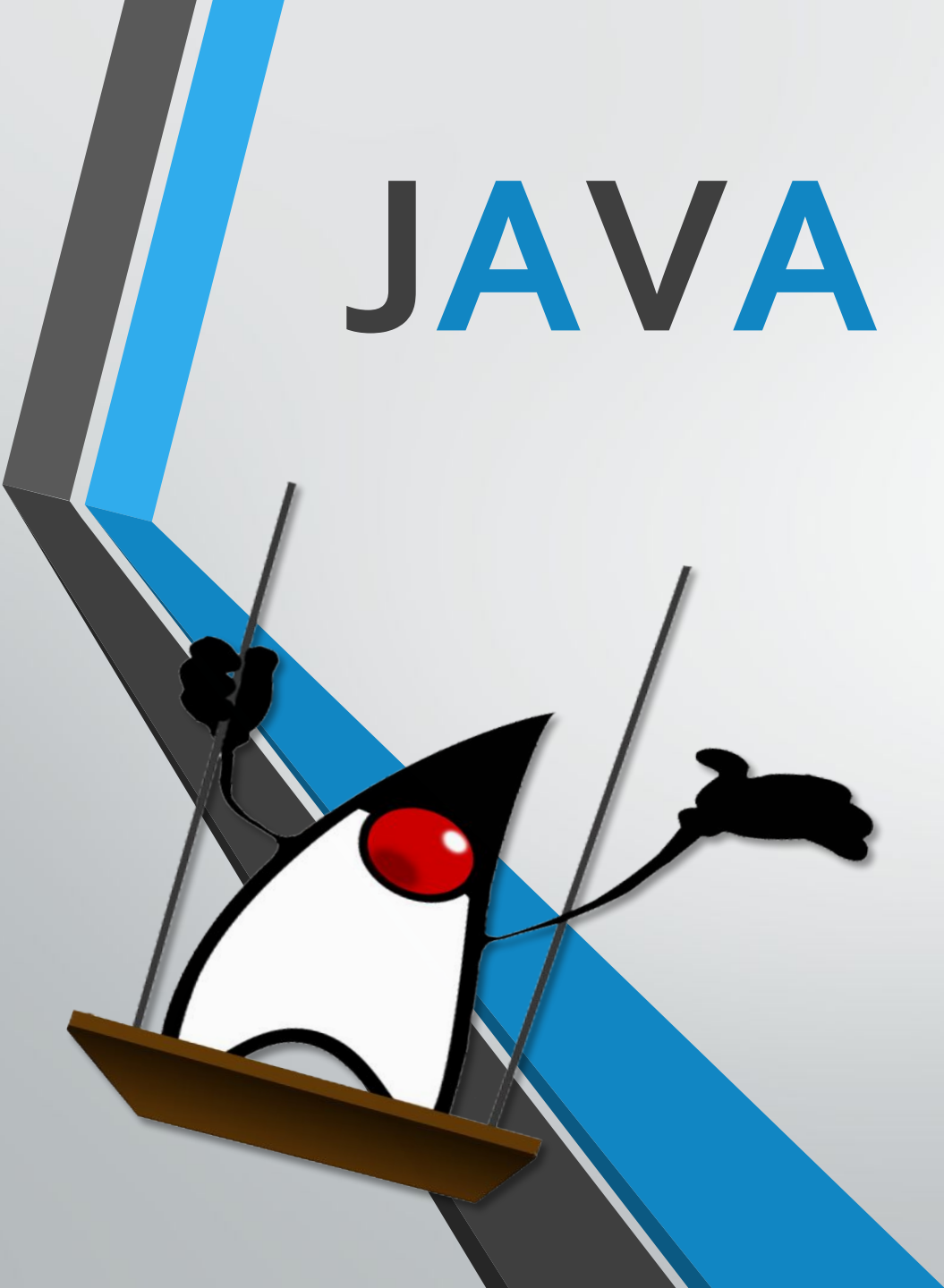
[0,0]	[0,1]	[0,2]	[0,3]	[0,4]	[0,5]						
[1,0]	[1,1]	[1,2]									
[2,0]											

- Esta forma de recorrer también nos servirá para los Arrays bidimensionales.

JAVA

ARRAYS

5. Arrays y el método main



5.- Arrays y el método main

- Fíjate que el método main, en todos los ejercicios que hemos hecho, siempre a recibido un array de Strings como parámetro, llamado "args":

public static void main (**String [] args**)

- Este array "args" nos sirve por si queremos pasarle parámetros a nuestro programa en el momento de ejecutarlo.
- Veamos un ejemplo:

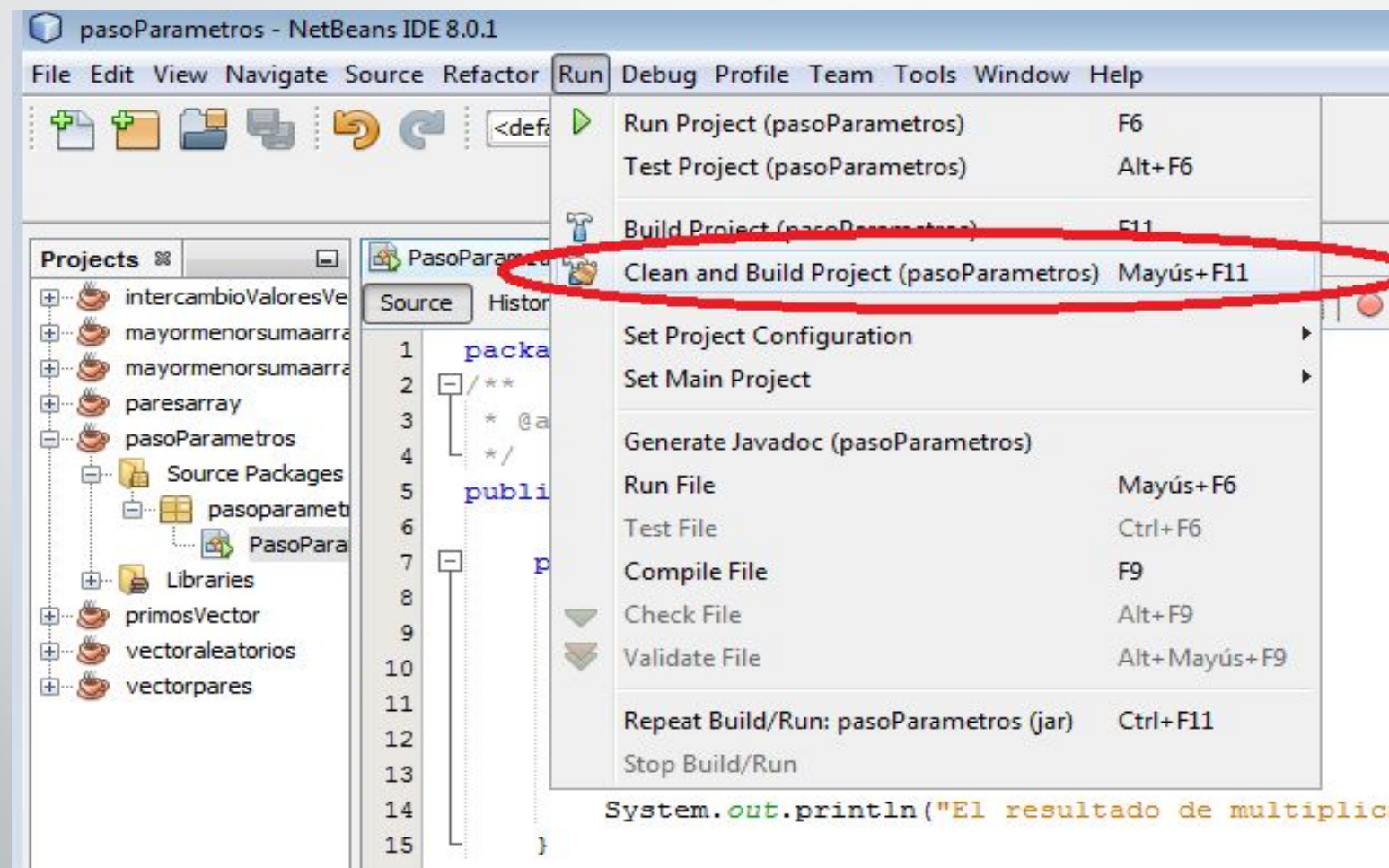
5.- Arrays y el método main

```
package pasoparametros;

/**
 * @author Oscar Laguna García
 */
public class PasoParametros {

    public static void main(String[] args) {
        int numero1;
        int numero2;
        int resultado;
        numero1=Integer.parseInt(args[0]);
        numero2=Integer.parseInt(args[1]);
        resultado=numero1*numero2;
        System.out.println("El resultado de multiplicar "+numero1+" y "+numero2+" es de "+resultado);
    }
}
```


5.- Arrays y el método main



5.- Arrays y el método main



```
C:\>java -jar pasoParametros.jar 5 6
El resultado de multiplicar 5 y 6 es de 30
C:\>
```

JAVA

ARRAYS

6. Ejercicios de consolidación



EJERCICIOS

- **Ejercicio 06.- (OPTATIVO)** Escribir un programa que rellene un array unidimensional con los 80 primeros números primos y luego los visualice.
- Ejemplo de ejecución:

Contenido de un array unidimensional de 80 posiciones con números primos:

[0] □ 1; [1] □ 2; [2] □ 3; [3] □ 5; [4] □ 7; [5] □ 11; [6] □ 13; [7] □ 17; [8] □ 19...

EJERCICIOS

- **Ejercicio 07.- (OPTATIVO)** Realiza un programa que cree 15 números aleatorios (entre 0 y 50) y muestre los 10 mayores.

EJERCICIOS

- **Ejercicio 08.- (OBLIGATORIO)** Realizar un programa en JAVA que le pida al usuario un número entero de 5 cifras y que le devuelva el mismo número escrito al revés.
- Para ello necesitarás un método que le pida el número al usuario, otro método dividirá el número en cifras y que las vaya metiendo en un array. Por último, otro método visualizará el array.
- Ejemplo de ejecución:

*Introduzca un número entero de 5 cifras: **85921***

*El numero introducido escrito al revés es el: **12958***

EJERCICIOS

- **Ejercicio 09.- (OPTATIVO)** Realizar un programa en JAVA que le pida al usuario un número entero y que le devuelva el mismo número escrito al revés.
- Para ello necesitarás un método que le pida el número al usuario, otro método calculará cuantas cifras tiene el número, otro método dividirá el número en cifras y las irá metiendo en un array. Por último, otro método visualizará el array al revés.
- Ejemplo de ejecución:

*Introduzca un número entero: **821***

*El numero introducido escrito al revés es el: **128***

EJERCICIOS

- **Ejercicio 10.- (OBLIGATORIO)** Escribe un programa que contenga un método que rellene un vector de 10 números enteros con números aleatorios entre 1 y 8, pudiendo contener elementos duplicados.
- Otro método visualizará el array creado. Luego, otro método debe sustituir cada valor repetido por 0. Para terminar, vuelve a visualizar el array ya modificado.
- Ejemplo de ejecución:

Se han generado los siguientes números: 8 1 5 7 2 1 5 4 3 6

Sustituimos los elementos repetidos por un 0: 8 0 0 7 2 0 0 4 3 6

EJERCICIOS

- **Ejercicio 11.- (OPTATIVO)** Escribe un programa que contenga un método que rellene un vector de 10 números enteros con números aleatorios entre 0 y 9, sin repetir.
- Otro método visualizará el array creado.
- Ejemplo de ejecución:

Se ha generado el siguiente array:

8 1 5 7 2 0 9 4 3 6

EJERCICIOS

- **Ejercicio 14.- (OBLIGATORIO)** Diseña programa que almacene las temperaturas medias de un mes que introduzca un usuario. Para hacerlo más sencillo vamos a suponer que el mes tiene 28 días y está formado por 4 semanas de 7 días. Hasta que el usuario pulse 5, mostrar un menú que nos permita:
 1. Rellenar las temperaturas.
 2. Mostrar las temperaturas.
 3. Visualizar la temperatura media del mes.
 4. Día o días más calurosos del mes. Ejemplo: *El día o días más calurosos fueron:*
 - El Jueves de la Semana 3 con 40 grados.
 - El Sábado de la Semana 4 con 40 grados.
 5. Salir del programa.
- *Fíjate que necesitarás otro array con el nombre de los días de la semana.*

EJERCICIOS

- **Ejercicio 15.- (OPTATIVO)** Realiza un programa que muestre un menú en el que se le ofrezcan al usuario las siguientes opciones:
 1. Rellenar un array unidimensional de 12 posiciones con las ventas de coches mensuales. Estas ventas serán números aleatorios entre 10 y 100.
 2. Mostrar las ventas introducidas en el punto anterior.
 3. Mostrar las ventas introducidas al revés.
 4. Que muestre la suma total de ventas del año.
 5. Que muestre las ventas totales de los meses pares.
 6. Que muestre el nombre del mes con más ventas. (Necesitarás otro array con el nombre de los meses)
 7. Salir del programa.
- Hasta que el usuario no pulse 7 no saldremos del programa y se volverá a mostrar el menú.

EJERCICIOS

- **Ejercicio 16.- (OBLIGATORIO)** Sabiendo que tenemos una clase de 6 alumnos (Pepe, Juan, Ana, Marta, Pedro y María), con 4 asignaturas cada uno (Lengua, Mates, Historia y Física), realiza un programa que le dé al usuario las siguientes opciones:
 1. Rellenar las notas de los alumnos.
 2. Mostrar las notas introducidas en el punto anterior.
 3. Que nos diga que alumno es el mejor de la clase. (nota media más alta) . (Necesitarás utilizar otro array unidimensional con los nombres de los alumnos)
 4. Que nos diga el alumno con más suspensos.
 5. Que nos diga cual es la asignatura más difícil. (nota media más baja) . (Necesitarás utilizar otro array unidimensional con los nombres de las asignaturas)
 6. Salir del programa.
- Hasta que el usuario no pulse 6 no saldremos del programa y se volverá a mostrar el menú.

EJERCICIOS

- **Ejercicio 17.- (OPTATIVO)** Diseña un método que cree un array unidimensional de 10 números enteros aleatorios entre el 0 y el 9. Luego, otro método, lo visualizará por pantalla, otro método ordenará sus elementos de mayor a menor, y por último volverás a visualizar el array ya ordenado.

EJERCICIOS

- **Ejercicio 18.- (OPTATIVO)** Diseña un método que tome como parámetros de entrada dos arrays de 5 enteros y devuelva como salida un único array de 10 enteros con los elementos de los anteriores arrays ordenados de menor a mayor.

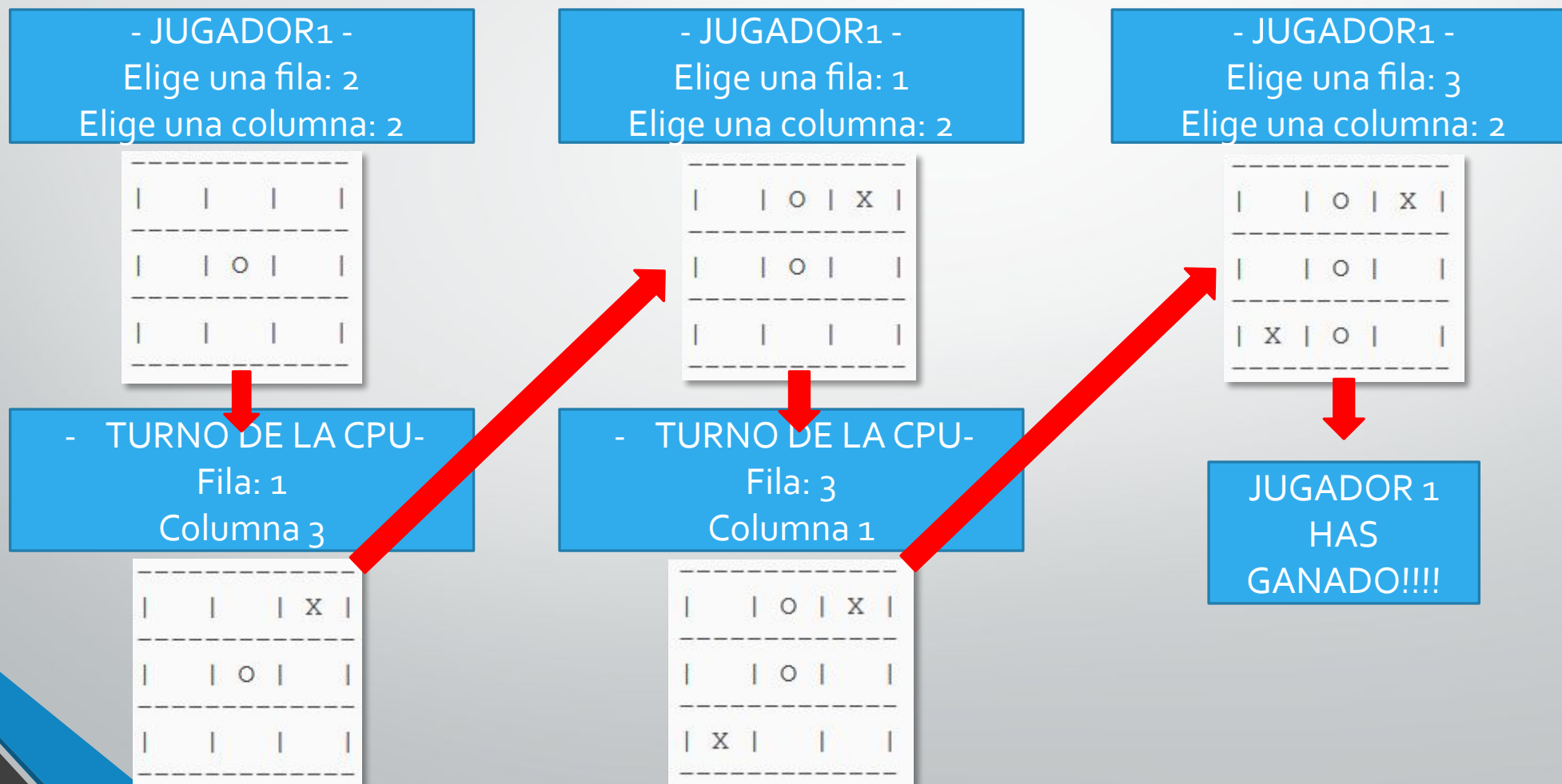
EJERCICIOS

- **Ejercicio 19.- (OPTATIVO)** Realiza un programa que cree un vector de 100 posiciones con números aleatorios entre 10 y 80. Una vez creado este vector, el programa deberá mostrar el mayor, el menor, el valor que más se repite y la media.

EJERCICIOS

- **Ejercicio 20.- (OBLIGATORIO)** Implementa en JAVA el juego del 3 en Raya mediante métodos estáticos. Para ello tendremos:
 - Un método que se encargará de dibujar el tablero con las fichas que ya hayan sido colocadas.
 - Otro método para pedirle al jugador la posición donde colocar la ficha.
 - Un método permitirá que el ordenador coloque una ficha al azar.
 - Otro método comprobará si el jugador que acaba de mover ha ganado la partida.
 - Por último, un método comprobará si la partida a terminado en empate:
 - Ejemplo de ejecución:


EJERCICIOS



EJERCICIOS

- **Ejercicio 21.- (OPTATIVO)** El Sudoku es un juego que consiste en completar un array de 9x9 con números entre el 1 y el 9, de tal forma que estos no se repitan en la misma fila, ni en la misma columna, ni en un sub-array de 3x3. Por ejemplo, esto es un Sudoku y su solución:

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



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

EJERCICIOS

- Crea un programa en JAVA que te pida los 81 números que forman un panel de Sudoku y calculará si el Sudoku pasado está formado correctamente o, por el contrario, es incorrecto:

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



¡CORRECTO!

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



¡INCORRECTO!