MÓDULO 0485 - PROGRAMACIÓN

UT 4-

ESTRUCTURAS DE DATOS ESTÁTICAS

Técnico Superior En Desarrollo De Aplicaciones Web

1º DAW (Grupo 1WV) (Vespertino)

Curso 2025-26

Profesor: Javier Rojo

ÍNDICE

ESTRUCTURAS DE DATOS ESTÁTICAS

ESTRUCTURAS DE DATOS ESTÁTICAS

ÍNDICE

INTRODUCCIÓN

- Supongamos que quiero pedir 5 notas al usuario
- Posteriormente, quiero volver a mostrar las 5 notas introducidas y decirle su media
- ¿Cómo podríamos hacerlo con lo que sabemos hasta ahora?

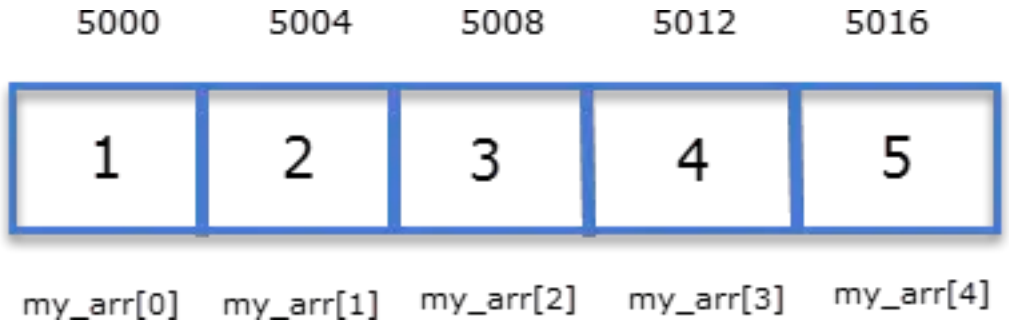
```
Dime tu nota 1: 4
Dime tu nota 2: 6
Dime tu nota 3: 3
Dime tu nota 4: 9
Dime tu nota 5: 7

Las notas introducidas son: [4, 6, 3, 9, 7]
La media es: 5
```

ARRAYS

Estructuras de datos

- Hasta ahora, para hacer referencia a un dato utilizábamos una variable
- **Problema**: cuando tenemos una gran cantidad de datos que guardan relación entre sí no podemos utilizar una variable para cada dato
- **Solución**: utilización de arrays
    - Un **array** es una **estructura de datos** con elementos del **mismo tipo** (numérico o alfanumérico), reconocidos por un nombre en común
    - El tamaño del array se establece en el momento de la creación
    - Para referirnos a cada elemento del array utilizamos un **índice** (la **primera posición** es la **0**)

- Los arrays **unidimensionales** se conocen con el nombre de **vectores**.
- Los arrays **bidimensionales** se conocen con el nombre de **matrices**.
- Los arrays de **tres o mfis dimensiones** se conocen con el nombre de **multidimensionales**.



ARRAYS UNIDIMENSIONALES

ARRAYS

Declaración

- Para declarar un array necesitamos dos instrucciones:

  1. Declaración: permite declarar el tipo de datos de la variable

  tipo identificador[]; o *tipo[] identificador;*

  Ejemplo:

```
int notas[];
```

```
double[] cuentas;
```

  - Esta declaración indica para qué servirá el array
  - No reserva espacio en la memoria RAM al no saberse todavía el tamaño del mismo.
  - Todavía no puede utilizarse el array, falta instanciarlo.

ARRAYS

Declaración

- Para declarar un array necesitamos dos instrucciones:

  2. Instanciación: se define el tamaño y queda listo para ser utilizado

```
int notas[];
notas = new int[8];
```

=

```
int notas[] = new int[8];
```

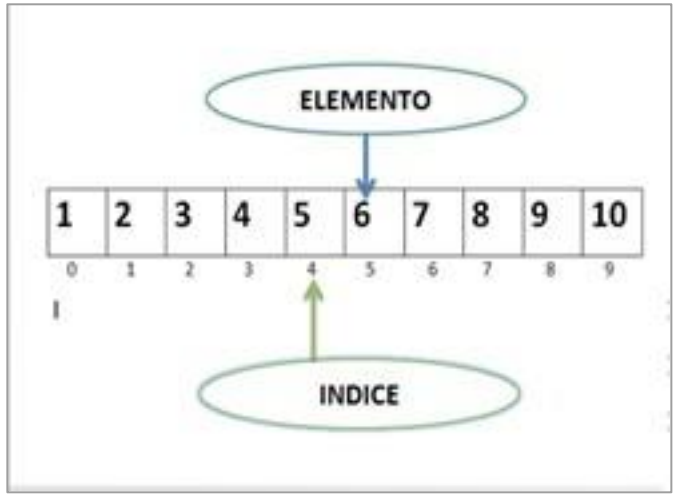
  - Define un vector de 8 posiciones
  - Cada posición se inicializa al valor por defecto del tipo de dato

Posición:	0	1	2	3
	0 0	0 0	0 0	0 0

¿Cuál es el valor por defecto de float? ¿Y de boolean? ¿Y de String?

ARRAYS

Estructuras de datos: arrays unidimensionales



- Los arrays que solo tienen una dimensión se conocen como **vectores**
- **Ejemplo**: declaración de un vector de 10 elementos enteros

```
int[] vector = new int[10];
```

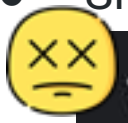
- Para acceder a cada uno de los elementos utilizamos un índice

```
vector[0] = 10;
System.out.println(vector[0]);
```

Asignamos el valor 10 a la posición 0

Escribimos por pantalla el valor de la posición 0. ¿Qué mostrará?

- Si accedemos a una posición que no existe se producirá un error



```
vector[10] = 3;
```

ARRAYS

Estructuras de datos: arrays unidimensionales

- Se pueden asignar los valores en el momento de la inicialización
- El tamaño del vector será el número de elementos que se añaden en la inicialización

```
String[] alumnos = {"Pepe", "Juan", "María", "Marta"};
```

- Las instrucciones equivalentes serían:

```
String[] alumnos = new String[4];
alumnos[0] = "Pepe";
alumnos[1] = "Juan";
alumnos[2] = "María";
alumnos[3] = "Marta";
```

ARRAYS

Estructuras de datos: arrays unidimensionales

Recorrido de un vector

- Podemos recorrer cada elemento de un vector con los bucles estudiados hasta ahora

- Ejemplos:

```
for (int i = 0; i < 10; i++) {
    vector[i] = i;
}
```

Recorrido para inicializar el vector

```
for (int i = 0; i < 10; i++) {
    System.out.println(vector[i]);
}
```

Recorrido para escribir los elementos del vector

Longitud del vector

¿Y si mañana el vector es de 30 posiciones y tengo 10 bucles que lo usan?

ARRAYS

Estructuras de datos: arrays unidimensionales

Recorrido de un vector

- ¿Cómo puedo conocer la longitud de un vector?
  - Con la propiedad length

```
int[] vector = new int[10];
System.out.println(vector.length);
```

// Muestra 10

-
```
for (int i = 0; i < vector.length; i++) {
    System.out.println(vector[i]);
}
```

ARRAYS

Estructuras de datos: arrays unidimensionales

Recorrido de un vector

- También podemos utilizar un nuevo tipo de bucle: **for each**

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

=

```
for (int valor : vector) {
    System.out.println(valor);
}
```

- Ten en cuenta:
  - Solo se utiliza en recorridos para trabajar con los valores (nunca para insertar)
  - No podemos saber qué posición estamos mostrando (no tenemos una variable iteradora)

ARRAYS

Estructuras de datos: arrays unidimensionales

Recorrido de un vector

- **Ejercicio**. Realiza un programa que defina un vector de 5 cadenas. Rellena el array con datos pedidos por teclado y, posteriormente, muestra las cadenas almacenadas en mayúsculas.

- **Ejercicio**: Realizar un programa que defina un vector llamado vector_numeros de 10 enteros y lo inicialice con valores aleatorios del 1 al 10. Posteriormente, debe mostrar por pantalla el valor de cada elemento del vector.

- **Ejercicio**: Realizar un programa que pida al usuario las 5 notas del módulo de Programación y las almacene en una estructura de datos. A continuación, calcula la nota media del alumno.

ARRAYS

Estructuras de datos: arrays u

Recorrido de un vector

- **Ejercicio**. Realiza un programa
  vector de 5 cadenas. Rellena e
  con datos pedidos por teclado
  muestra las cadenas almacen
  mayúsculas.

```java
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    String valor;
    String[] vector = new String[5];

    for (int i = 0; i < vector.length; i++) { // Bucle de relleno
        System.out.println("Introduce el valor de la posición " + i);
        valor = sc.nextLine();
        vector[i] = valor;
        // vector[i] = sc.nextLine(); Instrucción equivalente
    }

    for (int i = 0; i < vector.length; i++) { // Bucle para mostrar
        System.out.println(vector[i].toUpperCase());
    }
}
```

ARRAYS

Estructuras de datos: arrays unidimensionales

Recorrido de un vector

- **Ejercicio**. Realizar un programa
  vector llamado vector_numeros
  inicialice con valores aleatorios
  Posteriormente, debe mostrar p
  de cada elemento del vector.

```
public static void main(String[] args) {
    final int MIN_RANDOM = 1;
    final int MAX_RANDOM = 10;

    int[] valores = new int[10];

    for (int i = 0; i < valores.length; i++) { // Bucle de relleno
        valores[i] = (int)(MIN_RANDOM + Math.random()
                * (MAX_RANDOM - MIN_RANDOM + 1));
    }

    for (int i = 0; i < valores.length; i++) { // Bucle para mostrar
        System.out.println(valores[i]);
    }
}
```

ARRAYS

Estructuras de datos: arrays unidimensionales

Recorrido de un vector

- **Ejercicio**. Realizar un programa que
  5 notas
  del módulo de Programación y las
  almacene en una estructura de da
  calcula la nota media del alumno.

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    float[] notas = new float[5];

    for (int i = 0; i < notas.length; i++) { // Bucle de relleno
        System.out.print("Dime la nota " + i + ": ");
        notas[i] = sc.nextFloat();
    }

    float suma = 0;
    for (int i = 0; i < notas.length; i++) { // Bucle para mostrar
        suma += notas[i];
    }

    System.out.println("La nota media es " + suma / notas.length);
}
```

ARRAYS

Estructuras de datos: arrays unidimensionales

Copia de vectores

- Analiza el siguiente código:

```
public static void main(String[] args) {
    int[] v1 = {3, 4, 6};
    int[] v2 = {10, 20, 3, 5};

    v2 = v1;     // ¿Qué valores almacena v2 después de esta línea?
    v2[0] = 35; // ¿Ahora qué valores tiene v2? ¿Y v1?
    v1[2] = 9;  // ¿Y ahora qué valores tienen v1 y v2?

}
```

ARRAYS

Estructuras de datos: arrays unidimensionales

Copia de vectores

```
Valores de v1: 3, 4, 6,
Valores de v2: 3, 4, 6,
```

Salida

```java
public static void main(String[] args) {
    int[] v1 = {3, 4, 6};
    int[] v2 = {10, 20, 3, 5};

    v2 = v1;
    System.out.print("Valores de v1: ");
    for (int i = 0; i < v1.length; i++) {
        System.out.print(v1[i] + ", ");
    }
    System.out.print("\nValores de v2: ");
    for (int i = 0; i < v2.length; i++) {
        System.out.print(v2[i] + ", ");
    }

    v2[0] = 35;
    v1[2] = 9;
}
```

ARRAYS

Estructuras de datos: arrays unidimensionales

Copia de vectores

```
Valores de v1: 35, 4, 6,
Valores de v2: 35, 4, 6,
```

Salida

```java
public static void main(String[] args) {
    int[] v1 = {3, 4, 6};
    int[] v2 = {10, 20, 3, 5};

    v2 = v1;
    v2[0] = 35;
    System.out.print("Valores de v1: ");
    for (int i = 0; i < v1.length; i++) {
        System.out.print(v1[i] + ", ");
    }
    System.out.print("\nValores de v2: ");
    for (int i = 0; i < v2.length; i++) {
        System.out.print(v2[i] + ", ");
    }
    v1[2] = 9;
}
```

ARRAYS

Estructuras de datos: arrays unidimensionales

Copia de vectores

```
Valores de v1: 35, 4, 9,
Valores de v2: 35, 4, 9,
```

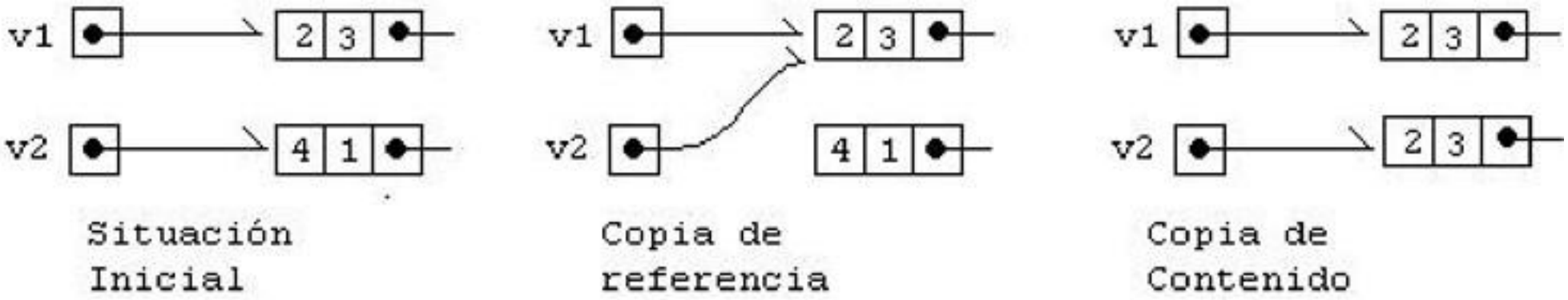
Salida

```java
public static void main(String[] args) {
    int[] v1 = {3, 4, 6};
    int[] v2 = {10, 20, 3, 5};

    v2 = v1;
    v2[0] = 35;
    v1[2] = 9;
    System.out.print("Valores de v1: ");
    for (int i = 0; i < v1.length; i++) {
        System.out.print(v1[i] + ", ");
    }
    System.out.print("\nValores de v2: ");
    for (int i = 0; i < v2.length; i++) {
        System.out.print(v2[i] + ", ");
    }
}
```

Estructuras de datos: arrays unidimensionales

Copia de vectores

- Al hacer v2=v1, lo que ocurre es que v2 apunta a la posición de memoria de v1
- Eso es lo que se denomina un **copia de referencia**



- Si queremos mantener ambos vectores independientes, necesitamos hacer una **copia de contenido**

ARRAYS

Estructuras de datos: arrays unidimensionales

Copia de vectores - Copia de contenido

- **Opción 1**: copiar los elementos uno a uno

- Salida del programa:

```
Valores de v1: 10, 20, 3,
Valores de v2: 10, 20, 3, 5,
```

- ¿Por qué v2 mantiene el 5?

```java
public static void main(String[] args) {
    int[] v1 = {3, 4, 6};
    int[] v2 = {10, 20, 3, 5};

    for (int i = 0; i < v1.length; i++) {
        v1[i] = v2[i];
    }
    // Comprobamos los valores de ambos
    System.out.print("Valores de v1: ");
    for (int i = 0; i < v1.length; i++) {
        System.out.print(v1[i] + ", ");
    }
    System.out.print("\nValores de v2: ");
    for (int i = 0; i < v2.length; i++) {
        System.out.print(v2[i] + ", ");
    }
}
```

ARRAYS

Estructuras de datos: arrays unidimensionales

Copia de vectores - Copia de contenido

- **Opción 2**: utilizar la función arraycopy

System.arraycopy(v_origen, i_origen, v_destino, i_destino, length);

v_origen: Vector orígen

i_origen: Posición inicial de la copia

v_destino: Vector destino

i_destino: Posición final de la copia

length: Cantidad de elementos a copiar

```java
public static void main(String[] args) {
    int[] v1 = {3, 4, 6};
    int[] v2 = {10, 20, 3, 5};

    System.arraycopy(v1, srcPos: 0, v2, destPos: 0, v1.length);
    // Comprobamos los valores de ambos
    System.out.print("Valores de v1: ");
    for (int i = 0; i < v1.length; i++) {
        System.out.print(v1[i] + ", ");
    }
    System.out.print("\nValores de v2: ");
    for (int i = 0; i < v2.length; i++) {
        System.out.print(v2[i] + ", ");
    }
}
```

```
Valores de v1: 3, 4, 6,
Valores de v2: 3, 4, 6, 5,
```

ARRAYS

Estructuras de datos: arrays unidimensionales

Copia de vectores - Copia de contenido

```
int[] v1 = {3, 4, 6};
int[] v2 = {10, 20, 3, 5};
```

- Analiza cuál será el contenido de v1 y v2 después de ejecutar cada una las siguientes sentencias:

```
System.arraycopy(v1, srcPos: 0, v2, destPos: 1, v1.length);
```

```
System.arraycopy(v1, srcPos: 0, v2, destPos: 1, length: 2);
```

```
System.arraycopy(v1, srcPos: 1, v2, destPos: 1, length: 3);
```

```
System.arraycopy(v2, srcPos: 0, v1, destPos: 0, v2.length);
```

```
System.arraycopy(v2, srcPos: 0, v1, destPos: 0, v1.length);
```

ARRAYS

Estructuras de datos: arrays unidimensionales

Copia de vectores - Copia de contenido

```
int[] v1 = {3, 4, 6};
int[] v2 = {10, 20, 3, 5};
```

- Analiza cuál será el contenido de v1 y v2 después de ejecutar cada una las siguientes sentencias:

```
System.arraycopy(v1, srcPos: 0, v2, destPos: 1, v1.length);
```

```
Valores de v1: 3, 4, 6,
Valores de v2: 10, 3, 4, 6,
```

```
System.arraycopy(v1, srcPos: 0, v2, destPos: 1, length: 2);
```

```
Valores de v1: 3, 4, 6,
Valores de v2: 10, 3, 4, 5,
```

```
System.arraycopy(v1, srcPos: 1, v2, destPos: 1, length: 3);
```

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException
    at java.base/java.lang.System.arraycopy(Native Method)
    at Main.main(Main.java:8)
```

```
System.arraycopy(v2, srcPos: 0, v1, destPos: 0, v2.length);
```

```
System.arraycopy(v2, srcPos: 0, v1, destPos: 0, v1.length);
```

```
Valores de v1: 10, 20, 3,
Valores de v2: 10, 20, 3, 5,
```

ARRAYS

Estructuras de datos: arrays unidimensionales

Búsqueda en un array desordenado

- Recorrer todos los elementos del array hasta:
  - Encontrar el elemento buscado
  - O hasta llegar al final del array → El elemento no está en el vector

- **Ejercicio**. Implementa un programa que defina el siguiente vector: [1, 10, 4, 3, 6]. Pídele un número al usuario e indícale si ese número está en el vector o no.
- **Ejercicio**. Implementa un programa que defina un vector de 10 posiciones con valores aleatorios del 1 al 10. Posteriormente, pide al usuario un número del 1 al 10 e indica si está en el vector o no. Debe cumplir las siguientes condiciones:
  - Obliga a que el usuario introduzca un número del 1 al 10
  - Indica la posición en la que has encontrado el elemento

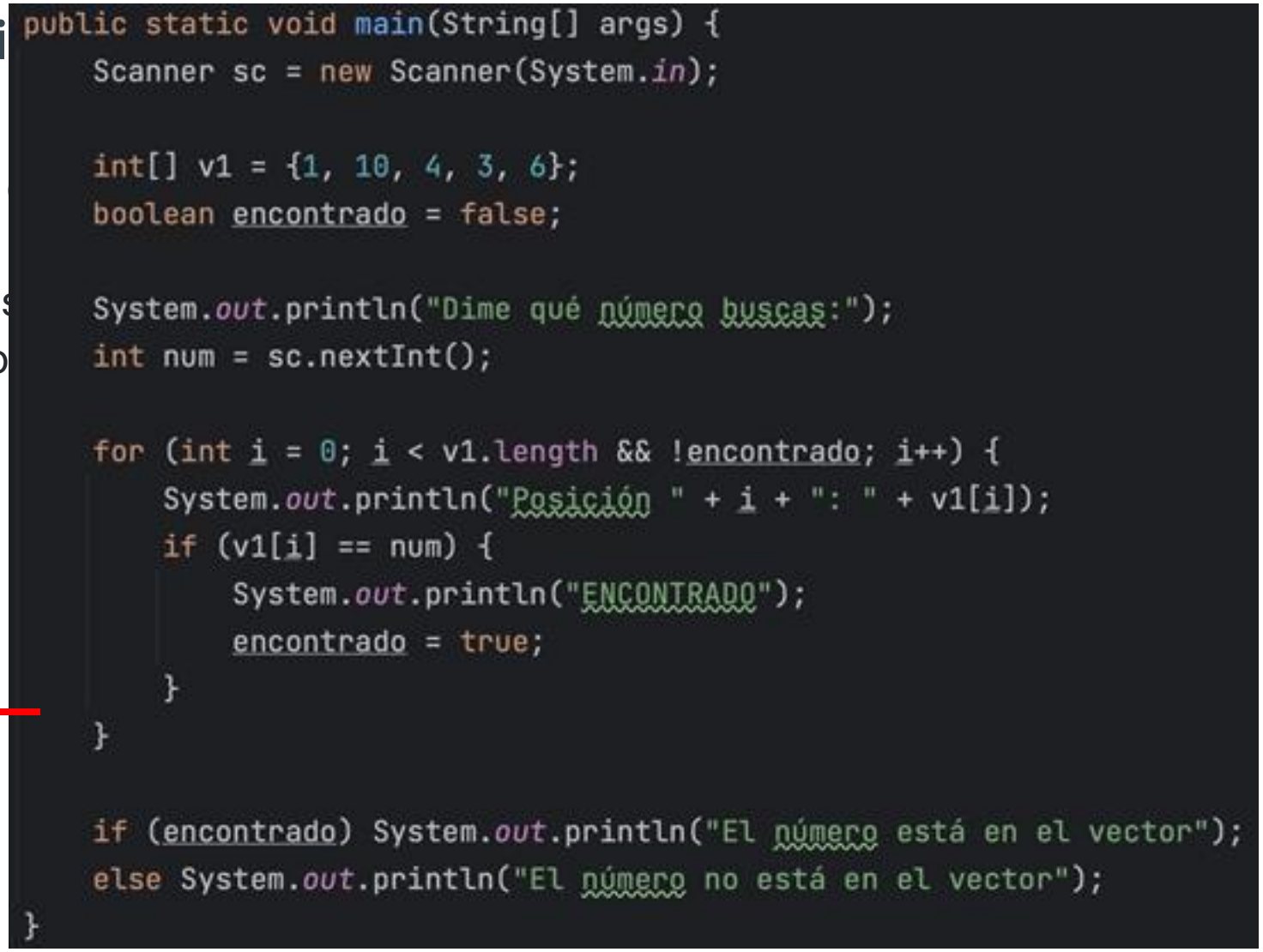
ARRAYS

Estructuras de datos: arrays unidi

Búsqueda en un array desordenado

- **Ejercicio**. Implementa un programa
  siguiente vector:
  [1, 10, 4, 3, 6]. Pídele un número al us
  si ese número está en el vector o no

```
Dime qué número buscas:
4
Posición 0: 1
Posición 1: 10
Posición 2: 4
ENCONTRADO
El número está en el vector
```



```java
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);

    int[] v1 = {1, 10, 4, 3, 6};
    boolean encontrado = false;

    System.out.println("Dime qué número buscas:");
    int num = sc.nextInt();

    for (int i = 0; i < v1.length && !encontrado; i++) {
        System.out.println("Posición " + i + ": " + v1[i]);
        if (v1[i] == num) {
            System.out.println("ENCONTRADO");
            encontrado = true;
        }
    }

    if (encontrado) System.out.println("El número está en el vector");
    else System.out.println("El número no está en el vector");
}
```

ARRAYS

Estructuras de datos: arrays unidimensio

Búsqueda en un array desordenado

- **Ejercicio 2**.

```
Valores v1: 1, 4, 2, 6, 7, 7, 6, 10, 3, 1,
Dime qué número buscas:
7
Posición 0: 1
Posición 1: 4
Posición 2: 2
Posición 3: 6
Posición 4: 7
ENCONTRADO
El número está en el vector en la posición 4
```

- **Problema de esta solución:**
  - Solo muestra la primera posición que lo encuentra
- Cambia el código para que muestre todas las posiciones en las que se encuentra el número y el nº de veces que aparece el número en el vector

```
Scanner sc = new Scanner(System.in);
final int MIN_RANDOM = 1;
final int MAX_RANDOM = 10;

int[] v1 = new int[10];
for (int i = 0; i < v1.length; i++) {
    v1[i] = (int) (MIN_RANDOM + Math.random() * (MAX_RANDOM - MIN_RANDOM + 1));
}

int num;
do { // Pedimos un número válido
    System.out.println("Dime qué número buscas:");
    num = sc.nextInt();
} while (num < MIN_RANDOM || num > MAX_RANDOM);

int posicionEncontrado = 0;
boolean encontrado = false;
for (int i = 0; i < v1.length && !encontrado; i++) {
    System.out.println("Posición " + i + ": " + v1[i]);
    if (v1[i] == num) {
        System.out.println("ENCONTRADO");
        encontrado = true;
        posicionEncontrado = i;
    }
}

if (encontrado) {
    System.out.println("El número está en el vector en la posición "
            + posicionEncontrado);
} else {
    System.out.println("El número no está en el vector");
}
```

ARRAYS

Estructuras de datos: arrays unidimensionales

Búsqueda en un array desordenado

- **Cambiamos el código de búsqueda**

```
Valores v1: 6, 8, 2, 8, 8, 7, 1, 3, 5, 10,
Dime qué número buscas:
8
ENCONTRADO en la posición: 1
ENCONTRADO en la posición: 3
ENCONTRADO en la posición: 4
El número aparece en el vector 3 veces
```

```
int contador = 0;
for (int i = 0; i < v1.length; i++) {
    if (v1[i] == num) {
        System.out.println("ENCONTRADO en la posición: " + i);
        contador++;
    }
}

if (contador != 0) {
    System.out.println("El número aparece en el vector "
            + contador + " veces");
} else {
    System.out.println("El número no está en el vector");
}
```

ARRAYS

Estructuras de datos: arrays unidimensionales

Búsqueda en un array ordenado (ascendente o descendente)

- Hay varias formas de hacerlo.
- La más sencilla de implementar consiste en recorrer todos los elementos del array hasta:
  - Encontrar el elemento buscado
  - Encontrar un elemento mayor que el buscado → el elemento no está en el array
  - Llegar al final del array → el elemento no está en el array
- Es de especial interés el método de búsqueda conocido como **búsqueda binaria** o dicotómica.

- **Ejercicio**. Implementa un programa que defina el siguiente vector ordenado: [1, 2, 4, 10, 25, 36]. Pídele un número al usuario e indícale si ese número está en el vector o no. Impleméntalo con for y while.

- **Ejercicio**. Implementa un programa que defina el siguiente vector ordenado: [30, 26, 14, 5, 2]. Pídele un número al usuario e indícale si ese número está en el vector o no.

ARRAYS

Estructuras de datos: arrays unid

Búsqueda en un array desordenado

- **Ejercicio**. Implementa un programa
  siguiente vector ordenado: [1, 2, 4, 1
  número al usuario e indícale si ese
  vector o no.

```
Dime qué número buscas: 4
Posición 0: 1
Posición 1: 2
Posición 2: 4
ENCONTRADO
El elemento está en el vector
```

```
Dime qué número buscas: 8
Posición 0: 1
Posición 1: 2
Posición 2: 4
Posición 3: 10
SALIMOS
El elemento no está en el vector
```

```java
Scanner sc = new Scanner(System.in);
int[] vector = {1, 2, 4, 10, 25, 36};

System.out.print("Dime qué número buscas: ");
int numero = sc.nextInt();

boolean salir = false, encontrado = false;
for (int i = 0; i < vector.length && !salir && !encontrado; i++) {
    System.out.println("Posición " + i + ": " + vector[i]);
    if (vector[i] == numero) {
        encontrado = true;
        System.out.println("ENCONTRADO");
    } else if (vector[i] > numero) {
        salir = true;
        System.out.println("SALIMOS");
    }
}

if (encontrado) {
    System.out.println("El elemento está en el vector");
} else {
    System.out.println("El elemento no está en el vector");
}
```

ARRAYS

Estructuras de datos: arrays unidimensio

Búsqueda en un array desordenado

- **Ejercicio**. Implementa un programa que defin
  siguiente vector ordenado: [1, 2, 4, 10, 25, 36]
  número al usuario e indícale si ese número e
  vector o no.

```
Dime qué número buscas: 4
Posición 0: 1
Posición 1: 2
Posición 2: 4
ENCONTRADO
El elemento está en el vector
```

```
Dime qué número buscas: 8
Posición 0: 1
Posición 1: 2
Posición 2: 4
Posición 3: 10
SALIMOS
El elemento no está en el vector
```

```java
Scanner sc = new Scanner(System.in);
int[] vector = {1, 2, 4, 10, 25, 36};

System.out.print("Dime qué número buscas: ");
int numero = sc.nextInt();

boolean salir = false, encontrado = false;
int i = 0;
while (i < vector.length && !salir && !encontrado) {
    System.out.println("Posición " + i + ": " + vector[i]);
    if (vector[i] == numero) {
        encontrado = true;
        System.out.println("ENCONTRADO");
    } else if (vector[i] > numero) {
        salir = true;
        System.out.println("SALIMOS");
    }
    i++;
}

if (encontrado) {
    System.out.println("El elemento está en el vector");
} else {
    System.out.println("El elemento no está en el vector");
}
```

ARRAYS

Estructuras de datos: arrays unidi

Búsqueda en un array desordenado

- **Ejercicio**. Implementa un programa
  siguiente vector ordenado: [30, 26, 1
  número al usuario e indícale si ese n
  vector o no.

```
Dime qué número buscas: 26
Posición 0: 30
Posición 1: 26
ENCONTRADO
El elemento está en el vector
```

```
Dime qué número buscas: 23
Posición 0: 30
Posición 1: 26
Posición 2: 14
SALIMOS
El elemento no está en el vector
```

```java
Scanner sc = new Scanner(System.in);
int[] vector = {30, 26, 14, 5, 2};

System.out.print("Dime qué número buscas: ");
int numero = sc.nextInt();

boolean salir = false, encontrado = false;
for (int i = 0; i < vector.length && !salir && !encontrado; i++) {
    System.out.println("Posición " + i + ": " + vector[i]);
    if (vector[i] == numero) {
        encontrado = true;
        System.out.println("ENCONTRADO");
    } else if (vector[i] < numero) {
        salir = true;
        System.out.println("SALIMOS");
    }
}

if (encontrado) {
    System.out.println("El elemento está en el vector");
} else {
    System.out.println("El elemento no está en el vector");
}
```

ARRAYS

Estructuras de datos: arrays unidimensionales

Ordenación de un array

- En primer lugar debemos determinar si queremos una ordenación ascendente o descendente.
- Seguidamente usamos uno de los muchos métodos de ordenación que hay.

- Puedes ver diversos métodos de ordenación en:
  http://artemisa.unicauca.edu.co/~nediaz/EDDI/cap03.htm

- **La clase Arrays de Java realiza el trabajo por nosotros:**

```
int vector[]={4,5,2,3,7,8,2,3,9,5};
Arrays.sort(vector); // Ordena de menor a mayor
```

```
[2, 2, 3, 3, 4, 5, 5, 7, 8, 9]
```

ARRAYS BIDIMENSIONALES

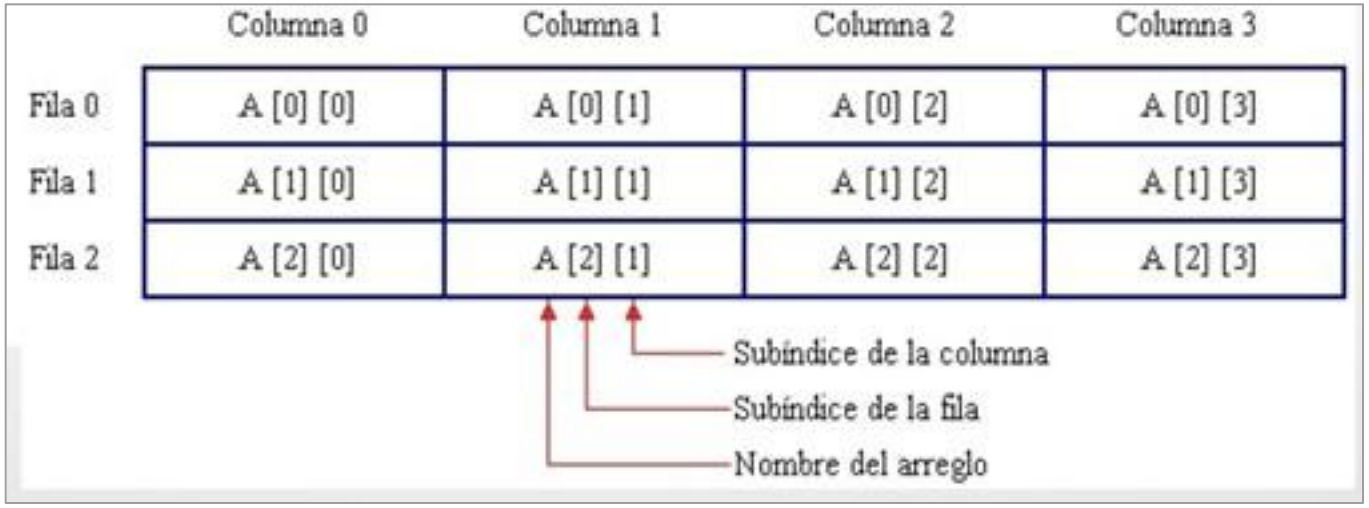
ARRAYS

Estructuras de datos: arrays bidimensionales

- Los arrays que tienen dos dimensiones se conocen como **tablas o matrices**
  - La **primera dimensión** indica el número de **filas**
  - La **segunda dimensión** el número de **columnas**
- **Ejemplo**: declaración de una matriz de enteros con 3 filas y 4 columnas

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

¿Cuántos elementos tendremos en total? 4x3=12 elementos



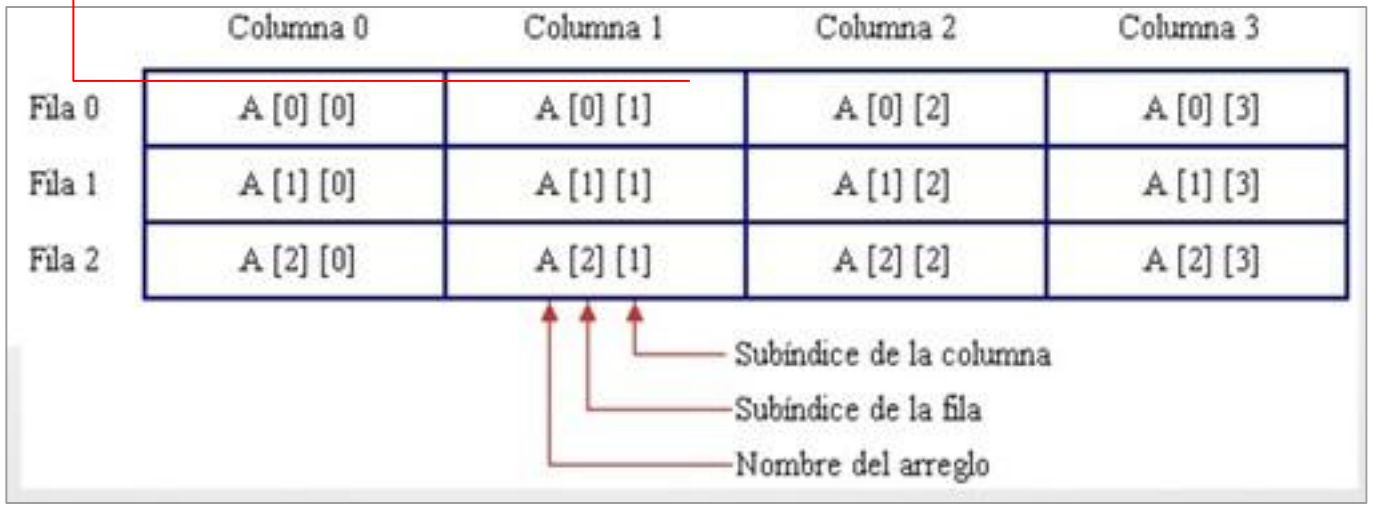
ARRAYS

Estructuras de datos: arrays bidimensionales

- Para acceder a cada uno de los elementos tenemos que indicar la fila y la columna en la que se encuentra (recordad que siempre empieza por el 0)
- **Ejemplo**: Inicialización del elemento de la primera fila y segunda columna

```
matriz[0][1] = 10;
```

- Si accedemos a una posición que no existe se producirá un error



ARRAYS

Estructuras de datos: arrays bidimensionales

Recorrido de una matriz

- Para recorrer todos los elementos de una matriz necesitamos dos bucles anidados
  - El bucle exterior permite recorrer las filas
  - El bucle interior permite recorrer las columnas

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

Recorrido para inicializar la matriz

```
for (int i = 0; i < matriz.length; i++) {
    for (int j = 0; j < matriz[0].length; j++) {
        System.out.print(matriz[i][j] + " ");
    }
    System.out.println("");
}
```

Recorrido para escribir los elementos de la matriz

Estructuras de datos: arrays bidimensionales

Recorrido de una matriz

- **Ejercicio.** Inicializa una **matriz de enteros** con los números del 1 al 5, sus cuadrados y sus cubos
  - ¿Cuántas filas necesitamos? ¿Cuántas columnas?

- **Ejercicio**. Realizar un programa que defina una matriz de 10 filas y 3 columnas de enteros. A continuación, inicializa la primera columna con valores aleatorios (del 1 al 10). Posteriormente, guarda el cuadrado de cada número en la 2º columna de la fila correspondiente y el cubo en la 3º. Finalmente, muestra los resultados.

ARRAYS

Estructuras de datos: arrays bidimensionales

Recorrido de una matriz

- **Ejercicio**. Inicializa una matriz con los números del 1 al 5, sus cuadrados y sus cubos

```
1 1 1
2 4 8
3 9 27
4 16 64
5 25 125
```

```java
int[][] matriz = new int[5][3];

for (int i = 0; i < matriz.length; i++) {
    for (int j = 0; j < matriz[0].length; j++) {
        matriz[i][j] = (int) Math.pow((i + 1), (j+1));
    }
}

// Mostramos la matriz
for (int i = 0; i < matriz.length; i++) {
    for (int j = 0; j < matriz[0].length; j++) {
        System.out.print(matriz[i][j] + " ");
    }
    System.out.println("");
}
```

ARRAYS

Estructuras de datos: arrays bidimensionales

Recorrido de una matriz

- **Ejercicio 2.**

```
3 9 27
10 100 1000
3 9 27
4 16 64
7 49 343
3 9 27
1 1 1
7 49 343
10 100 1000
5 25 125
```

```java
final int MIN_RANDOM = 1;
final int MAX_RANDOM = 10;
int[][] matriz = new int[10][3];

for (int i = 0; i < matriz.length; i++) {
    matriz[i][0] = (int) (MIN_RANDOM + Math.random() * (MAX_RANDOM - MIN_RANDOM + 1));
}

for (int i = 0; i < matriz.length; i++) {
    for (int j = 1; j < matriz[0].length; j++) {
        matriz[i][j] = (int) Math.pow(matriz[i][0], (j+1));
    }
}

// Mostramos la matriz
for (int i = 0; i < matriz.length; i++) {
    for (int j = 0; j < matriz[0].length; j++) {
        System.out.print(matriz[i][j] + " ");
    }
    System.out.println("");
}
```

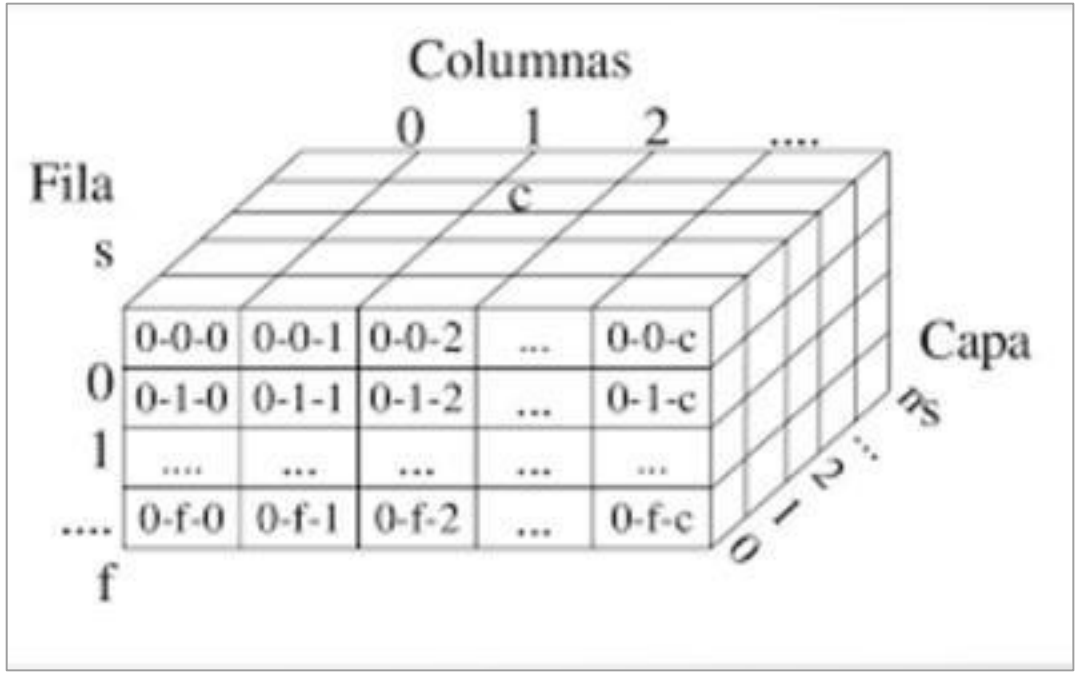
ARRAYS MULTIMENSIONALES

ARRAYS

Estructuras de datos: arrays multidimensionales

- Los arrays pueden tener las dimensiones que queramos
- Ejemplo: array de tres dimensiones de valores enteros

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



- Para inicializar el primer elemento accedemos a la posiciones 0:

```
tabla[0][0][0] = 10;
```

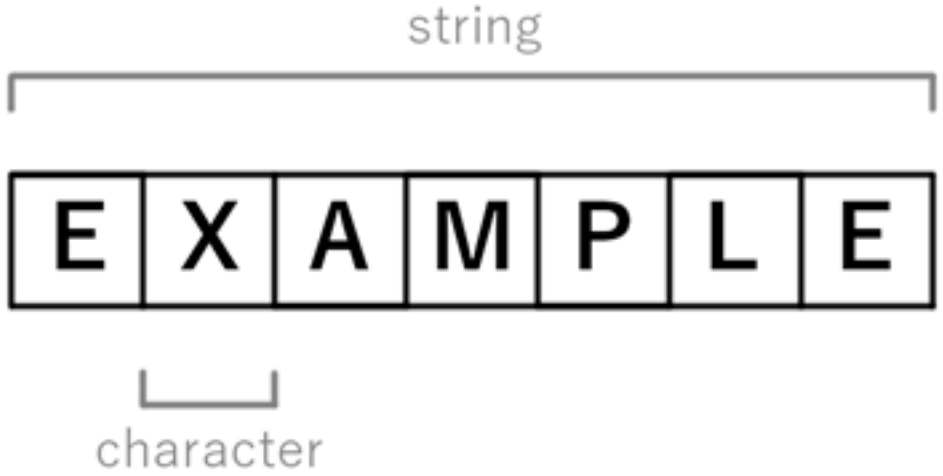
ARRAYS

Estructuras de datos: arrays multidimensionales

Recorrido de un array multidimensional

- Por cada dimensión necesitamos un bucle
- Ejemplo de recorrido de un array de tres dimensiones

```
for (int i = 0; i < tabla.length; i++) {
    for (int j = 0; j < tabla[0].length; j++) {
        for (int k = 0; k < tabla[0][0].length; k++) {
            System.out.println(tabla[i][j][k]);
        }
    }
}
```



CLASE STRING

CLASE STRING

Comparaciones

- No podemos utilizar el operador ==
- Se utiliza uno de estos métodos:
  - **cadena1.equals(cadena2)** → distingue mayúsculas de minúsculas
  - **cadena1.equalsIgnoreCase(cadena2)** → no se tienen en cuenta mayúsculas y minúsculas

- ¿Qué salida produce este código?
- ¿Qué es “null”?

```
String cad1 = null;
String cad2 = "Hola";

System.out.println(cad2.equals(cad1));
System.out.println(cad1.equals(cad2));
```

CLASE STRING

Comparaciones

- Para **comparar** teniendo en cuenta el **orden alfabético**
  - **cadena1.compareTo(cadena2)** → distingue mayúsculas de minúsculas
  - **cadena1.compareToIgnoreCase(cadena2)** → no se tienen en cuenta mayúsculas y minúsculas
- Esos métodos devuelven:
  - valor positivo → si 1º cadena > 2º cadena
  - 0 → si son iguales
  - valor negativo → si 1º cadena < 2º cadena
- **OJO**: el orden no es el del alfabeto español, sino el de la tabla ASCII (la ñ es mucho mayor que la 0)

CLASE STRING

Métodos más utilizados

- Se usan de la forma ***miCadena.nombreMetodo(argumentos)***

BIBLIOGRAFÍA

BIBLIOGRAFÍA

Apuntes actualizados y adaptados a partir de la siguiente documentación:

1. [1] Apuntes Programación de José Antonio Díaz-Alejo. IES Camp de Morvedre.

2. [2] Apuntes Programación de Javier Valero Lionel Tarazón. Ceedcv.

¿DUDAS?

FIN

Javier Rojo
fjrojom001@educarex.es