IES DR LLUÍS SIMARRO

ENSENYAMENT PÚBLIC

XÀTIVA

CFGS Desarrollo de aplicaciones web

Módulo profesional: Programación

GENERALITAT VALENCIANA

Conselleria d'Educació, Investigació, Cultura i Esport

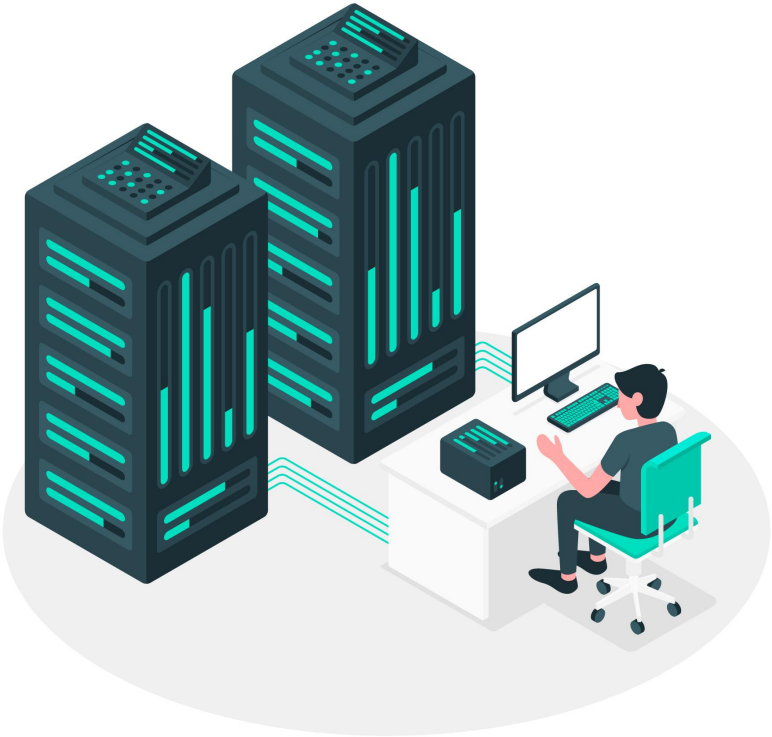
GOBIERNO DE ESPAÑA

MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL

Unió Europea

Fons Social Europeu

L'FSE inverteix en el teu futur

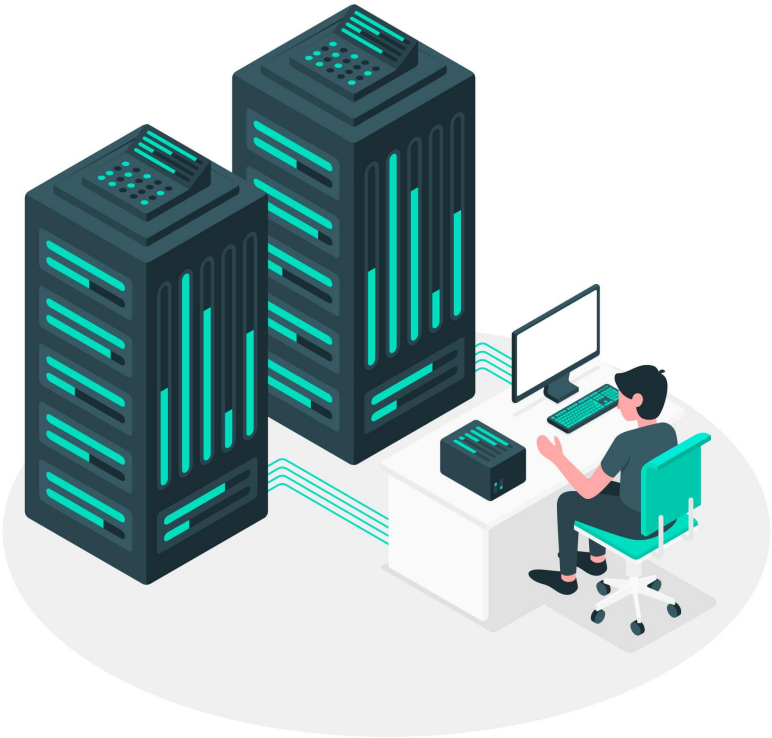


Material elaborado por:

Anna Sanchis

Revisado y editado por:

Edu Torregrosa Llácer



Datos profesor

Edu Torregrosa Llácer

eduardotorregrosa@ieslluissimarro.org

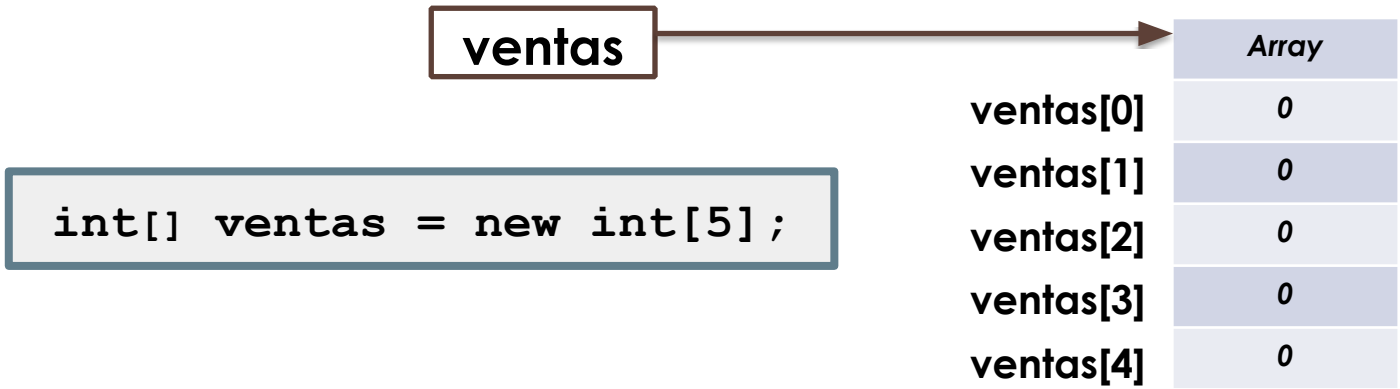
Web del módulo: https://aules.edu.gva.es/

Clases predefinidas JAVA

1. Arrays multimensionales
    a. Definición
    b. Declaración
    c. Uso
    d. Ejemplos
2. Métodos de arrays
    a. Copia
    b. Clonado
    c. Ordenación

Definición

Ejemplo de array de una dimensión para almacenar las ventas de cinco comerciales:




```
int[] ventas = new int[5];
```

Definición

- Si por ejemplo, necesitáramos almacenar las ventas por ciudad, podríamos necesitar una dimensión adicional:

```
int[][] ventas2 = new int[10][5];
```

- Para manejarnos mejor, se puede interpretar que la primera dimensión serían las **filas** (10) y que la segunda serían las **columnas**(5). Esto coincidiría con el concepto de matriz matemática.
- Se podrían seguir añadiendo dimensiones, por ejemplo para incluir los años:

```
int[][][] ventas3 = new int[20][10][5];
```

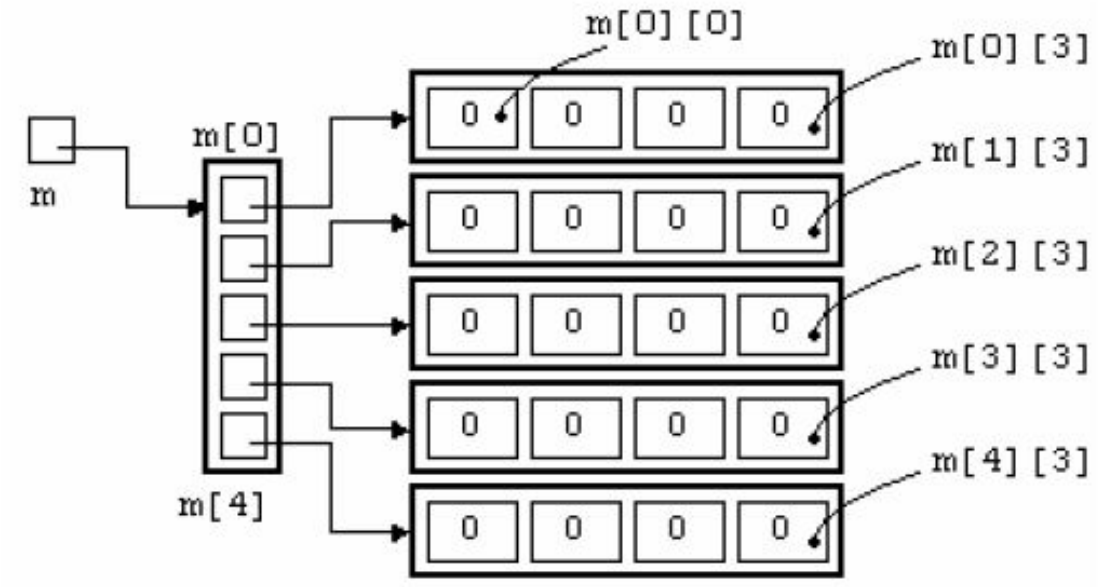
Definición

- Un array multidimensional es aquel que para acceder a una posición concreta, en vez de utilizar un único valor como índice, se utiliza una secuencia de diversos valores. Cada índice sirve como coordenada para una dimensión diferente.
- Un array de dimensión 2 es una tabla.
- La sintaxis es como la de los arrays unidimensionales pero se le añade un [] más.

```
Tipo[] identificadorVariable = new Tipo[numeroFilas][numeroColumnas];
```

Declaración de arrays bidimensional

```
int [ ][ ] m = new int [5][4];
```



Declaración de arrays bidimensional

```
public static void main(String[] args)
{
        //Declaración Arrays de 1, 2 y 3 dimensiones
        int[] ventas = new int[5];
        int[][] ventas2 = new int[10][5];
        int[][][] ventas3 = new int[20][10][5];
        System.out.println(Arrays.toString(ventas));

        //Otras formas de declarar un array multidimensional
        int[][] noMatriz = new int[5][];
        noMatriz[0] = new int[1];
        noMatriz[1] = new int[3];

 }
```

Declaración de arrays bidimensionales

La inicialización con valores concretos:

```
int[][] arrayBidi = {
  {1 ,2 ,3 ,4 ,5 },
  {6 ,7 ,8 ,9 ,10},
  {11,12,13,14,15}
        };
```

Cuyo resultado sería:

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

Uso de arrays bidimensionales

El acceso a las posiciones de un array bidimensional sería:

```
identificadorVariable[índexFila][índexColumna]
```

Se muestra a continuación las posiciones de cada elemento en una matriz de 5 x 4

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]
a[2][0]	a[2][1]	a[2][2]	a[2][3]	a[2][4]
a[3][0]	a[3][1]	a[3][2]	a[3][3]	a[3][4]

Uso de arrays bidimensionales

Comportamiento del método **length**:

Tiene un comportamiento especial:

a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]
a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]
a[2][0]	a[2][1]	a[2][2]	a[2][3]	a[2][4]
a[3][0]	a[3][1]	a[3][2]	a[3][3]	a[3][4]

Por ejemplo, si dicha matriz la identificamos como **arrayBidi**:

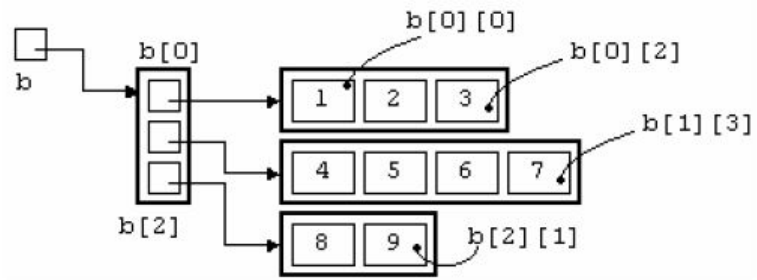
- **arrayBidi.length** vale 4 (hay 4 filas)
- **arrayBidi[0].length**, **arrayBidi[1].length**, etc. vale 5 (hay 5 columnas).

Uso de arrays bidimensionales

- El número de elementos de los arrays referenciados puede especificarse para cada uno de ellos y ser diferente entre sí.
- Por ejemplo, la ejecución de la sentencia:

```
int [ ][ ] b = { {1, 2, 3}, {4, 5, 6, 7}, {8, 9} };
```

Genera esta estructura de datos, donde no existen, por ejemplo, los elementos b[0][3] ni b[2][2].



Uso de arrays bidimensionales

- Para recorrer un array de varias dimensiones la mejor opción es utilizar bucles for anidados.
- Un bucle anidado es un for dentro de otro for, en el que cada índice se utiliza para una dimensión diferente.
- Sea por ejemplo, la siguiente matriz de 3x4:

```
int [][] matrizDeEnteros = {{1,3,5,7},{5,4,1,16},{7,9,61,13}};
```

1	3	5	7
5	4	1	16
7	9	61	13

Uso de arrays bidimensionales

```
public class Ejemplo_Recorrer {

    public static void main(String[] args) {

        //Declaración y inicialización

        int [][] matrizDeEnteros = {{1,3,5,7},{5,4,1,16},{7,9,61,13}};

        //Recorrido

        for (int fila = 0; fila < matrizDeEnteros.length; fila++) {

            for (int columna = 0; columna < matrizDeEnteros[fila].length; columna++)

                System.out.print(matrizDeEnteros[fila][columna]+” ”);

            System.out.println("");

        } } }
```

Ejercicio

Crear y cargar una matriz de 4 filas por 4 columnas. Imprimir la diagonal principal:

x - - -

- x - -

- - x -

- - - x

Debes utilizar los siguientes métodos:

- public void cargar()
- public void imprimirDiagonalPrincipal()

Métodos de arrays

Copiar arrays

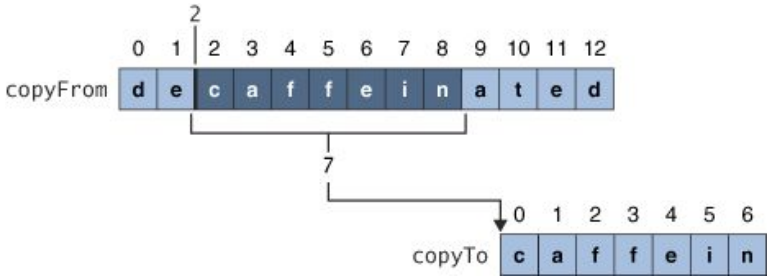
Copiar un array significa pasar los datos de un array a otro.

Existe un método en System que permite hacerlo:

```
public static native void arraycopy(Object from, int fromStart,
                                    Object to, int dst_position, int count)
```

Parámetros

- **from**: Array origen
- **fromStart**: Posición de inicio



en origen

- **to**: Array destino
- **toStart**: Posición de inicio en destino
- **count**: cantidad de elementos a copiar

Es importante que tanto el array origen como el array destino estén dimensionados

Copiar arrays

```
public class Ejemplo_Copia_Array {

    public static void main(String[] args) {

        //Declaración e inicialización

        int[] primos = {1, 2, 3, 5, 7, 11, 13, 17, 19, 23};

        int[] copia = new int[primos.length];

        //Copia

        System.arraycopy(primos, 1, copia, 3, 6);

        //Salida

        System.out.println(Arrays.toString(primos));

        System.out.println(Arrays.toString(copia));        }}
```

Clonado

- Todos los objetos en java tienen el método clone que realiza una copia exacta del contenido en otro espacio de memoria.

- Si se utiliza en los arrays, nos permitirá hacer una copia del mismo y de esta forma crear otro array independiente del anterior.

Ejemplo de clonado

```
public class Ejemplo_Clone_Array {
    public static void main(String[] args) {
        //Declaración e inicialización
        int[] primos = {1, 2, 3, 5, 7, 11, 13, 17, 19, 23};
        int[] copiaClonado;
        int[] copiaReferencia;
        //Clonación
        copiaClonado = primos.clone();
        //Cambio el clon, sin afectar al original
        copiaClonado[0] = 0;
        //Copia de referencia
        copiaReferencia = primos;
        //Cambio la copia y afecta al original
        copiaReferencia[1] = 0;

        //Salida
        System.out.println(Arrays.toString(primos));
        System.out.println(Arrays.toString(copiaClonado));
        System.out.println(Arrays.toString(copiaReferencia));
    }}
```

Ordenación

- **¿Para qué sirven?**
  - Para reorganizar el orden de los elementos de una estructura (p.e. vector)
- **¿Por qué ordenar los datos?**
  - Es mucho más eficiente trabajar con datos cuando la información está ordenada.
- **¿Qué es un algoritmo de ordenación?**
  - Forma estructurada que indica cómo ordenar los elementos que hay dentro.
- **¿Qué podemos ordenar?**
  - Cualquier estructura con elementos ordenables: números, meses, etc.
- **¿Son todos los algoritmos iguales?**
  - No, algunos son más eficientes que otros.
- **¿Qué algoritmos hay?**
  - Selección, inserción, burbuja, etc.

Ordenación - Método burbuja

Intercambia cada pareja consecutiva que no esté ordenada

```
Para i=0 hasta n
    Para j=0 hasta n-i-1
        Si A[i] > A[j]
        Intercambiar ( A[i], A[j])
```

El método burbuja

```
public class Ejemplo_Burbuja {
    public static int[] ordenBurbuja(int[] vector) {
        int aux;
        for (int i = 0; i < vector.length; i++) {
            for (int j = 0; j < vector.length - i - 1; j++) {
             //Hacemos que el valor máximo esté lo más a la derecha posible
                if (vector[j] > vector[j + 1]) {
                    aux = vector[j];
                    vector[j] = vector[j + 1];
                    vector[j + 1] = aux;
                }
             }
         }
       return vector;    }
```

El método burbuja

```
    public static void main(String[] args) {
        //Declaración e inicialización
        int[] arrayEjemplo = {3, 6, 9, 0, 1, 2, 3};
        int[] arrayOrdenado;
        //Salida inicial
        System.out.println("Sin ordenar: ");
        System.out.println(Arrays.toString(arrayEjemplo));
        //Ordenación
        arrayOrdenado = ordenBurbuja(arrayEjemplo);
        //Salida final
        System.out.println("Ordenado: ");
        System.out.println(Arrays.toString(arrayOrdenado));
    }
}
```

Preguntas