Grado en Ingeniería Informática
Introducción a la Programación
2021-2022

Tema 2. Introducción a los Arrays

- Arrays unidimensionales
- Arrays multidimensionales

Tema 2. Introducción a los Arrays

Arrays unidimensionales

- Introducción
- Declaración y creación de un array
- Acceso a los elementos de un array
- Procesamiento en un array

Tema 2. Introducción a los Arrays

Arrays unidimensionales

Introducción

Supongamos que necesitamos leer 100 números, calcular su media y encontrar cuantos de ellos tienen un valor superior a la media. Nuestro programa debería en primer lugar leer los números y calcular a continuación la media para después comparar ese valor con cada uno de ellos y determinar si es superior o no. Los números se deben almacenar en variables, ello supondría la declaración de 100 variables y el programa resultaría poco práctico. ¿Cómo solucionar el problema?.

Java y los lenguajes de alto nivel proporcionan una estructura de datos, *arrays*, que permiten dar una solución más eficiente al problema. En lugar de declarar 100 variables individuales tales como numero0, numero1, … y numero99 declaramos una única variable array, numeros, y utilizamos numeros[0], numeros[1] … numeros[99] para representar elementos individuales.

Tema 2. Introducción a los Arrays

Arrays unidimensionales

Es la estructura de datos representativa del **acceso directo**. Es una colección de elementos del mismo tipo, es decir, es una **estructura de datos homogénea**, a la que podemos **acceder** a sus elementos individuales **a partir de un índice** (**posición**). Es **una estructura de datos estática** y en cuanto al almacenamiento interno, los elementos del array **se almacenan en posiciones contiguas de memoria.**

Propiedades de los arrays

✓ Los arrays se utilizan como **contenedores** para almacenar datos relacionados (en vez de declarar variables por separado para cada uno de los elementos del array).

✓ La forma de utilizar un elemento concreto de un array es a través del nombre de la variable array junto al número (**índice**) que distingue ese elemento entre corchetes (`[ ]`). Por ejemplo, `array[3]`

Tema 2. Introducción a los Arrays

Arrays unidimensionales

✓ Todos los **datos** incluidos en el array son **del mismo tipo**. Se pueden crear arrays de enteros de tipo `int` o de reales de tipo `float`, pero en un mismo array no se pueden mezclar datos de tipo `int` y datos de tipo `float`.

✓ El tamaño del array se establece cuando se crea el array (con el operador **new**, igual que cualquier otro objeto).

✓ A los elementos del array se accederá a través de la posición (**índice**) que ocupan dentro del conjunto de elementos del array.

```
double [] array;
array = new double[7];
array[0] = 1.0;
```

Tema 2. Introducción a los Arrays

Arrays unidimensionales

Clasificación de los arrays

✓ **Según el número de elementos con datos**. **Array completo** (es aquél que utiliza siempre todos los elementos para almacenar valores) y **Array incompleto** (aquél en el que el número de elementos almacenados es menor que el tamaño del array). En este último caso, se recomienda tener los datos agrupados al principio o al final, y una variable que controle el número de elementos utilizados (`numElem`).

✓ **Según el orden de los datos**. **Array ordenado** (en este caso los datos estarán ordenados según algún criterio, independientemente de que el array esté completo o incompleto, y se debe de gestionar las inserciones y eliminaciones correctamente para que el array permanezca ordenado) y **Array desordenado** (el orden no es un criterio para el almacenamiento de los datos, y al insertar o eliminar un elemento no nos preocupamos del orden)

Tema 2. Introducción a los Arrays

Arrays unidimensionales

❑ Declaración

Para declarar un array, se utilizan corchetes **[ ]**

```
tipo identificador [];
```

o bien

```
tipo [] identificador;
```

donde

tipo es el tipo de dato de los elementos del array

identificador es el nombre de la variable array

Tema 2. Introducción a los Arrays

Arrays unidimensionales

❑ Creación

Los arrays se crean con el operador **new**

```
identificador = new tipo[numeroElementos];
```

Entre corchetes [ ] se indica el tamaño del array, es decir, el número de elementos.

❑ Declaración y creación

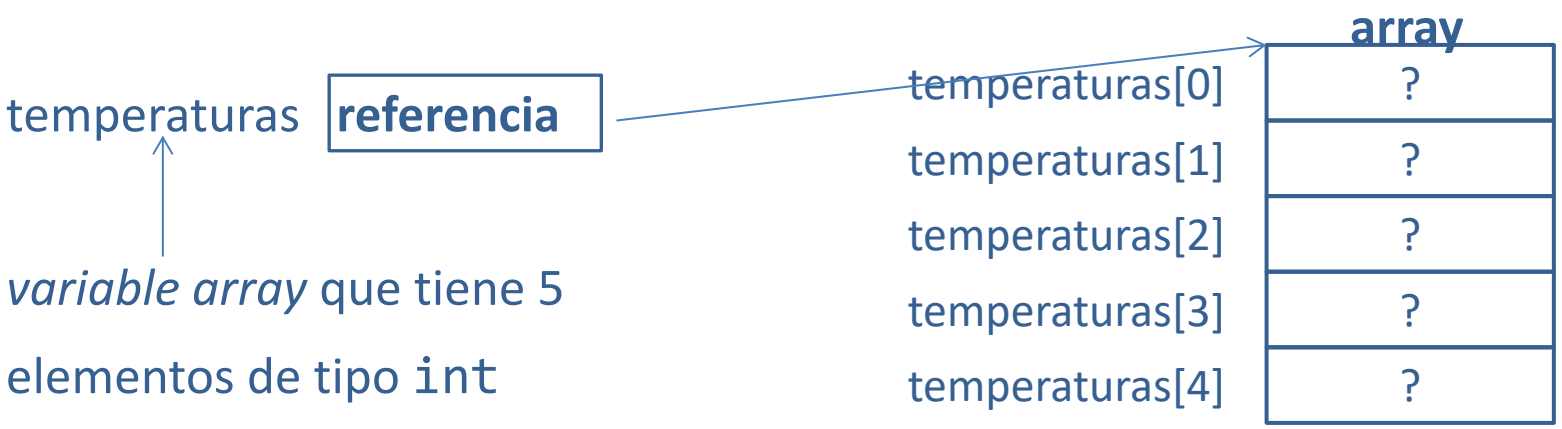
```
tipo [] identificador  = new tipo[numeroElementos];
```

Tema 2. Introducción a los Arrays

Arrays unidimensionales

- **Ejemplos**

```
final int NUMERO_DE_ELEMENTOS = 50;
float [] notas = new float[NUMERO_DE_ELEMENTOS];
int [] temperaturas = new int[5];
```

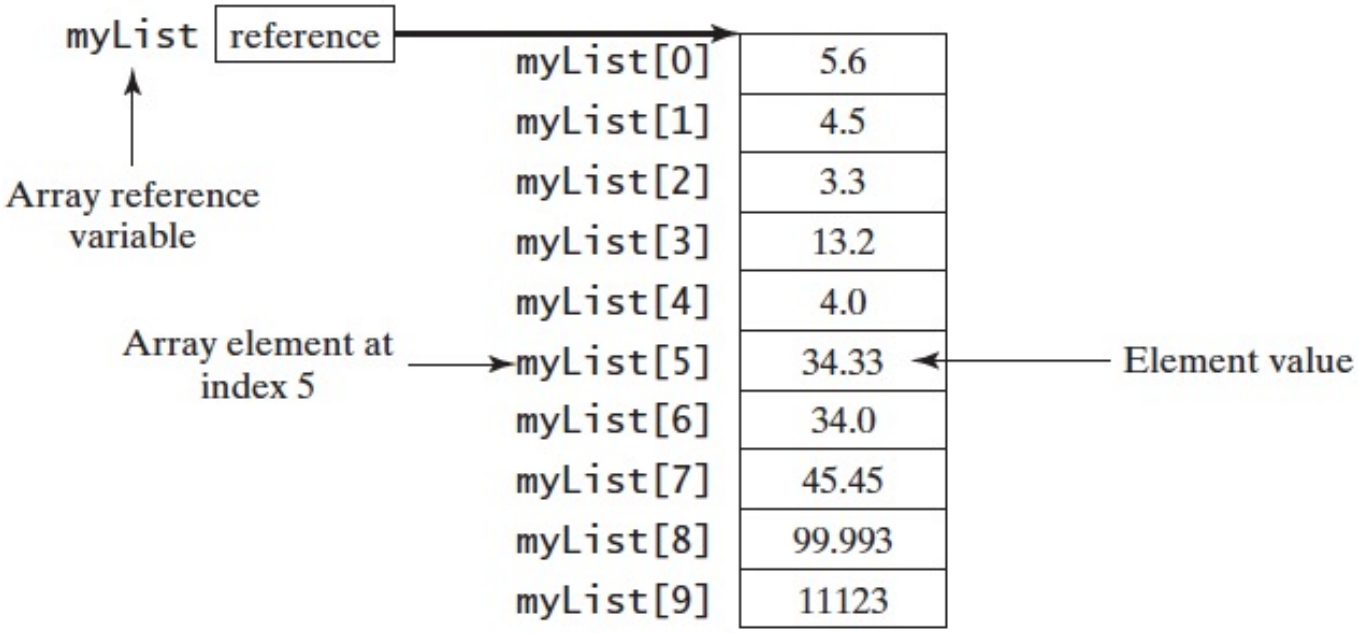


Tema 2. Introducción a los Arrays

Arrays unidimensionales

- **Ejemplo** ⇒ El array `myList` tiene 10 elementos de tipo `double`, e índices `int` de 0 a 9

```
double[] myList = new double[10];
```



Tema 2. Introducción a los Arrays

Arrays unidimensionales

❑ Referencias y variable array

En Java a cada dirección de memoria se le denomina **referencia**. Cada posición de la memoria de un ordenador tiene una dirección única que lo identifica.

Las variables de array lo que almacenan realmente es una **referencia**. Las referencias se modifican en cada ejecución dependiendo de la ocupación de la memoria.

Cuando una **variable de array** se le asigna a un array, se dice que la variable *referencia* al array. Las variables array pueden verse como medios para acceder a los elementos del array a los que referencian.

Es posible acceder a un mismo array mediante *más de una variable* o, dicho de otra forma, el mismo array puede estar referenciado por más de una variable de array.

La única condición para que una variable pueda referenciar a un array es que el tipo de la variable y el del array **coincidan**.

Tema 2. Introducción a los Arrays

Arrays unidimensionales

❑ Declaración, creación e inicialización

tipo [] identificador = { valor0, valor1, …, valor*k*}

Ejemplo

```
int [] temperaturas = {20, 35, 15, 10, -5};
```

temperaturas[0]	20
temperaturas[1]	35
temperaturas[2]	15
temperaturas[3]	10
temperaturas[4]	-5

Tema 2. Introducción a los Arrays

Arrays unidimensionales

❑ Acceso a los elementos de un array

Para acceder a los elementos de un array utilizamos **índices** que indican la posición del elemento en el array

array[indice]

- ✓ En Java, el **indice** del *primer* elemento de un array es siempre **0**.
- ✓ El tamaño del array puede obtenerse utilizando la propiedad **array.length**
- ✓ Por tanto, el **indice** del *último* elemento es **array.length-1**

Tema 2. Introducción a los Arrays

Arrays unidimensionales

❑ Procesamiento en un array

Las operaciones se realizan elemento a elemento. Cuando realizamos una operación que afecta a todos o parte de los elementos de un array se utiliza un bucle **for** por dos razones:

- Todos los elementos del array son del mismo tipo, por tanto, se procesarán del mismo modo usando un bucle **for**.
- El tamaño del array es conocido, resulta natural utilizar un bucle **for.**

Ejemplo 1. Almacenar en el array temperaturas las temperaturas leídas de teclado.

```
Scanner entrada = new Scanner (System.in);
System.out.println("Introduce los valores de las temperaturas");
for (int i = 0; i < temperaturas.length; i++) {
        System.out.println("Introduce temperatura " + (i + 1));
        temperaturas[i] = entrada.nextInt();
}
```

Tema 2. Introducción a los Arrays

Arrays unidimensionales

Ejemplo 2. Inicializar el array de temperaturas con valores aleatorios.

```
int N = 50;
for (int i = 0; i < temperaturas.length; i++) {
        temperaturas[i] = (int) (Math.random() * N);
}
```

Ejemplo 3. Mostrar el array de temperaturas.

```
for (int i = 0; i < temperaturas.length; i++) {
        System.out.print(temperaturas[i] + "\t");
}
```

Tema 2. Introducción a los Arrays

Arrays unidimensionales

Ejemplo 4. Calcular la media de todos los elementos del array de temperaturas.

```
double suma = 0.0, media;
for (int i = 0; i < temperaturas.length; i++) {
        suma += temperaturas[i] ;
}
media = suma / temperaturas.length;
```

Ejemplo 5. Encontrar el elemento mayor del array de temperaturas.

```
int max = temperaturas[0];
for (int i = 1; i < temperaturas.length; i++) {
        if (temperaturas[i] > max)
                max = temperaturas[i];
}
```

Tema 2. Introducción a los Arrays

Arrays unidimensionales

Ejemplo 6. Calcular la media de todos los elementos de un array usando un método de clase (**static**).

```
public static double media(int [] array) {
        double suma = 0.0;
        for (int i = 0; i < array.length; i++) {
                suma += array[i] ;
        }
        return suma / array.length;
}
```

Tema 2. Introducción a los Arrays

Arrays unidimensionales

Ejemplo 7. Encontrar el menor índice del elemento con mayor valor del array.

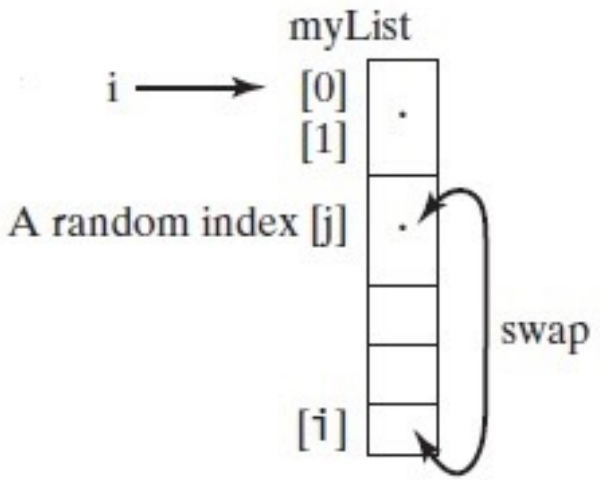
```
double max = array[0];
int indiceMaxValor = 0;
for (int i = 1; i < array.length; i++) {
        if (array[i] > max) {
                max = array[i] ;
                indiceMaxValor = i;
        }
}
```

Tema 2. Introducción a los Arrays

Arrays unidimensionales

Ejemplo 8. Aleatorizar el contenido del array (**random suffling**).

```
for (int i = array.length - 1; i > 0; i--) {
        int j = (int)(Math.random() * i);
        double temp = array[i] ;
        array[i] = array[j];
        array[j] = temp;
}
```

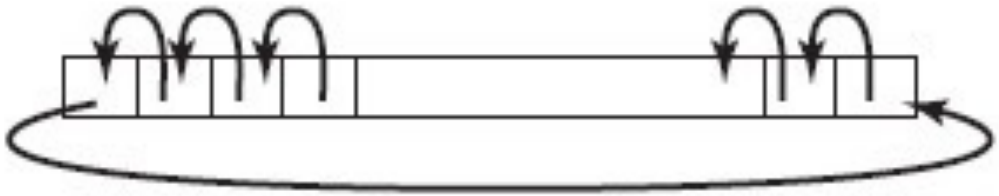


Tema 2. Introducción a los Arrays

Arrays unidimensionales

Ejemplo 9. Desplazamiento de elementos en el array (**shifting elements**).

```
double temp = array[0] ;
for (int i = 1; i < array.length; i++) {
        array[i - 1] = array[i];
}
array[array.length - 1] = temp;
```



Tema 2. Introducción a los Arrays

Arrays unidimensionales

Más sobre array

null: literal que se encarga de hacer que una variable array que referencia a un array, no referencie a nada (función contraria a **new**)

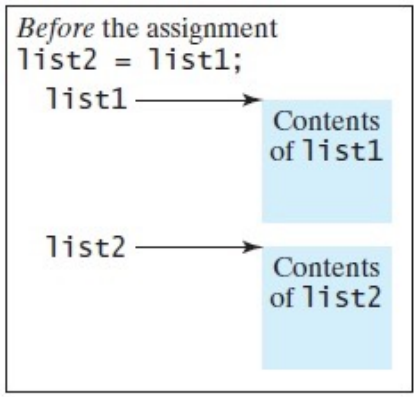
Arrays como parámetros de métodos. En Java, el mecanismo de paso de parámetros al invocar a un método es que el valor de la variable que se utiliza se copia al parámetro del método. Cuando utilizamos arrays como parámetros el mecanismo es el mismo, es decir, el valor de la variable array utilizado es la llamada se **copia** al parámetro del método. Pero en este caso, lo que se **copia es la referencia a un array** y con ello se consigue que el array esté referenciado tanto por la variable array como por el parámetro del método. Por lo que, la modificación de un elemento del array dentro del método también es visible desde la variable array externa al método (**MUY IMPORTANTE**)

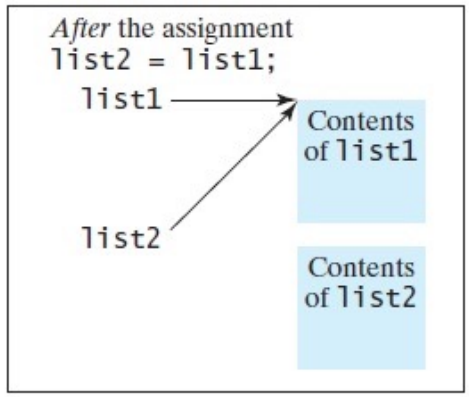
Tema 2. Introducción a los Arrays

Arrays unidimensionales

Ejemplo 10. Copia de arrays ⇒ Copiar el contenido de un array en otro, para ello tenemos que **copiar elemento a elemento** de un array a otro.

```
int [] arrayFuente = {2, 3, 1, 5, 10};
int [] arrayDestino = new int[arrayFuente.length];
for (int i = 0; i < arrayFuente.length; i++) {
        arrayDestino[i] = arrayFuente[i];
}
```





Tema 2. Introducción a los Arrays

Arrays unidimensionales

❑ Paso de arrays a métodos

- Cuando se pasa un array a un método, se pasa la **referencia** del array al método

```
1  package org.ip.tema02;
2  public class PasandoArraysAMetodos {
3      public static void main(String[] args) {
4          int x = 1;
5          int[] y = new int[10];
6          y[0] = 1;
7          System.out.println("x = " + x);
8          System.out.println("y[0] = " + y[0]);
9          metodo(x, y);
10         System.out.println("x = " + x);
11         System.out.println("y[0] = " + y[0]);
12     }
13     public static void metodo(int numero, int[] array) {
14         numero = 1001;
15         array[0] = 7777;
16     }
17 }
```

SALIDA

```
x = 1
y[0] = 1
x = 1
y[0] = 7777
```

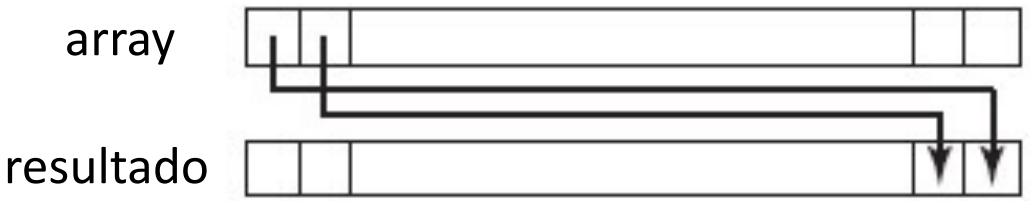
Tema 2. Introducción a los Arrays

Arrays unidimensionales

❑ Devolver un array desde un método

- Cuando un método devuelve un array, se devuelve su **referencia**

```
public static int[] invertir(int[] array) {
        int[] resultado = new int[array.length];
        for (int i = 0, j = array.length - 1; i < array.length; i++, j--) {
                resultado[j] = array[i];
        }
        return resultado;
}
```



Tema 2. Introducción a los Arrays

Arrays unidimensionales

Error común

Accedemos a una posición que no existe

```
1  package org.ip.tema02;
2
3  public class Ejemplo1Array {
4      public static double media(int [] array) {
5          double suma = 0.0;
6          // Se dispara una excepcion
7          for (int i = 0; i <= array.length; i++) {
8              suma += (double)array[i];
9          }
10         return suma / array.length;
11     }
12
13     public static void main(String[] args) {
14         int [] temperaturas = {5, 10, 25, -4, 3};
15         double valorMedio = media(temperaturas);
16         System.out.println("La media es: " + valorMedio);
17     }
18 }
```

Se produce la excepción

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
        at org.ip.tema04.Ejemplo1Arrays.media(Ejemplo1Arrays.java:8)
        at org.ip.tema04.Ejemplo1Arrays.main(Ejemplo1Arrays.java:15)
```

Tema 2. Introducción a los Arrays

Arrays unidimensionales

Recorrido ⇒ Visitar sus elementos para procesarlos. Por procesar se entiende cualquier operación que realicemos con un elemento, como por ejemplo, asignarle un valor, realizar algún tipo de cálculo o mostrarlo. El recorrido puede ser total (se recorren todos sus elementos) o parcial (se recorre un subconjunto de sus elementos). El código tipo para recorrer un array es el siguiente (`desde` = primer elemento a visitar y `hasta` = último elemento visitado):

```
for (int i = desde; i <= hasta; i++) {
        // Procesado de array[i]
}
```

Tema 2. Introducción a los Arrays

Arrays unidimensionales

Recorrido ⇒ Ejemplo donde se quiere incrementar un 10% todos los elementos de un array

```
for (int i = 0; i <= array.length - 1; i++) {
        array[i] = array[i] + 0.1 * array[i];
}

for (int i = 0; i < array.length; i++) {
        array[i] = array[i] + 0.1 * array[i];
}
```

Tema 2. Introducción a los Arrays

Arrays unidimensionales

Recorrido ⇒ La instrucción **for** tiene una sintaxis alternativa, conocida como *for-each*, que permite recorrer los elementos de un array

```
for (tipoArray elemento : array) {
        // Procesado de elemento
}
```

Donde se declara una variable (`elemento`), que tiene que ser del mismo tipo que el array, a la que se asignará, en cada iteración, el valor de un elemento. Es **muy importante** tener en cuenta que la variable es una copia de cada elemento del array, y que en el caso de que se modifique, estamos modificando una copia y no el elemento de la tabla

Tema 2. Introducción a los Arrays

Arrays unidimensionales

Recorrido ⇒ Ejemplo de uso de la sintaxis *for-each* para una instrucción **for**, que permite mostrar los elementos de un array y calcular la suma total de todos los elementos de un array de `double`

```
double sumaTotal = 0.0;
for (double elemento : array) {
        System.out.println(elemento);
        sumaTotal += elemento;
}
System.out.println("Suma tota = " + sumaTotal);
```

Tema 2. Introducción a los Arrays

Arrays unidimensionales

Inserción ⇒ Insertar un nuevo elemento al final del array (primer elemento vacío disponible), con `numElem` elementos utilizados. El valor `numElem` coincide con el índice donde vamos a insertar el nuevo elemento. Por tanto, con el array vacío, `numElem` se inicializará a 0 y se incrementará después de cada inserción

```
if (numElem == array.length)
        // No insertamos en el array, pues no hay huecos al final
else {
        array[numElem] = nuevo;
        numElem++;      // ahora tenemos un elemento más
}
```

Tema 2. Introducción a los Arrays

Arrays unidimensionales

Eliminación o **Borrado** ⇒ Para eliminar un elemento del array, primero tenemos que buscarlo, recorriendo uno por uno todos los elementos, si lo encontramos, lo sustituimos por el último elemento del array, con lo que conseguimos que no queden huecos y que todos todos los datos estén contiguos. Para finalizar, decrementamos el número de elementos útiles

```
if (indice == array.length)          // Elemento no encontrado
        // Elemento no encontrado, pues no hacemos nada
else {
        array[indice] = array[numElem - 1];
        numElem--;      // ahora tenemos un elemento menos
}
```

Tema 2. Introducción a los Arrays

Arrays multidimensionales

- Introducción
- Declaración y creación de un array multidimensional
- Acceso a los elementos de un array multidimensional
- Procesamiento en un array multidimensional

Tema 2. Introducción a los Arrays

Arrays bidimensionales

❑ Introducción

En el apartado anterior hemos estudiado cómo usar un **array unidimensional** para almacenar una **colección lineal** de elementos. Podemos usar un **array bidimensional** para guardar una **matriz** o una tabla. Por ejemplo, la tabla siguiente describe las distancias entre ciudades, podemos usar un array bidimensional para almacenarlas.

	Chicago	Boston	New York	Atlanta	Miami	Dallas	Houston
Chicago	0	983	787	714	1375	967	1087
Boston	983	0	214	1102	1763	1723	1842
New York	787	214	0	888	1549	1548	1627
Atlanta	714	1102	888	0	661	781	810
Miami	1375	1763	1549	661	0	1426	1187
Dallas	967	1723	1548	781	1426	0	239
Houston	1087	1842	1627	810	1187	239	0

```
double[][] distances = {
  {0, 983, 787, 714, 1375, 967, 1087},
  {983, 0, 214, 1102, 1763, 1723, 1842},
  {787, 214, 0, 888, 1549, 1548, 1627},
  {714, 1102, 888, 0, 661, 781, 810},
  {1375, 1763, 1549, 661, 0, 1426, 1187},
  {967, 1723, 1548, 781, 1426, 0, 239},
  {1087, 1842, 1627, 810, 1187, 239, 0},
};
```

Tema 2. Introducción a los Arrays

Arrays bidimensionales

❑ Declaración y creación

```
tipo [][] identificador  = new tipo[elementos1][elementos2];
```

Ejemplos

```
int [][] matriz = new int[5][5];
int [][] matrizA = {{4,5,9},{10,14,25}, {7,8,15}}
Equivalente a: int [][] matrizA = new int[3][3];
matrizA[0][0]= 4; matrizA[0][1]= 5; matrizA[0][2]= 9;
matrizA[1][0]= 10; matrizA[1][1]= 14; matrizA[1][2]= 25;
matrizA[2][0]= 7; matrizA[2][1]= 8; matrizA[2][2]= 15;
```

matrizA

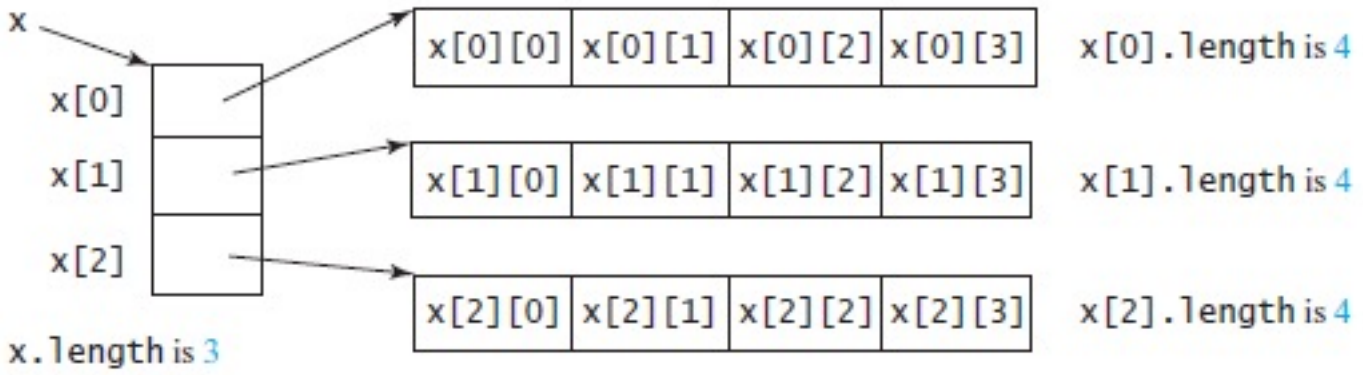
4	5	9
10	14	25
7	8	15

Tema 2. Introducción a los Arrays

Arrays bidimensionales

Un array bidimensional lo podemos ver como un array unidimensional en el cual cada elemento es otro array unidimensional.

Por ejemplo: int [][] x = new int[3][4]; podemos representarlo como sigue,

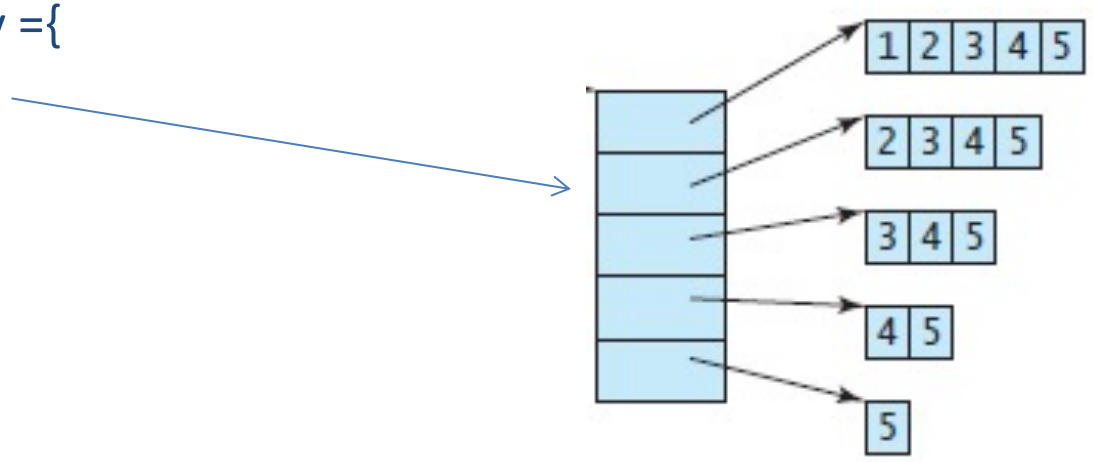


Tema 2. Introducción a los Arrays

Arrays bidimensionales

Cada fila en el array bidimensional es a su vez un array unidimensional, por tanto, las filas pueden tener diferentes longitudes. Un array de este tipo se conoce como *array desigua*. Por ejemplo:

```
int [][] trianguloArray ={
            {1,2,3,4,5},
            {2,3,4,5},
            {3,4,5},
            {4,5},
            {5}
    };
```



Equivalente a: int [][] trianguloArray = new int[5][]; trianguloArray[0] = new int[5]; trianguloArray[1] = new int[4]; trianguloArray[2] = new int[3]; etc.

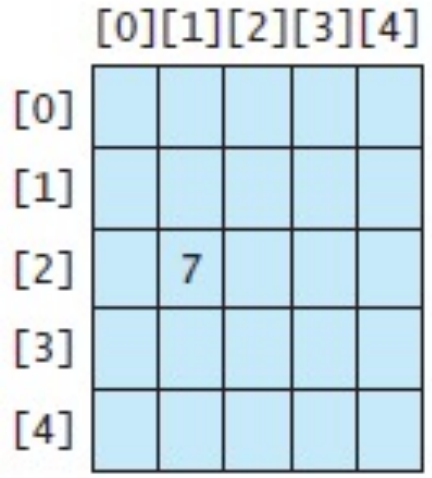
Tema 2. Introducción a los Arrays

Arrays bidimensionales

❑ Acceso a los elementos de un array bidimensional

Para acceder a los elementos de un array utilizamos índices, en este caso necesitaremos dos que indican la *fila* y la *columna*.

```
int [][] matriz = new int[5][5];
matriz[2][1] = 7;
```



Tema 2. Introducción a los Arrays

Arrays bidimensionales

❑ Procesamiento en un array bidimensional

Si se van a tratar todos los elementos del array o parte de ellos que ocupan posiciones consecutivas utilizaremos **dos bucles for** (anidados), uno para indicar las *filas* y el otro para las *columnas*. Vamos a ver algunos ejemplos:

Ejemplo 1. Inicializar el array bidimensional con valores que introducimos por teclado.

```
int [][] matriz = new int [3][3];
Scanner entrada = new Scanner(System.in);
System.out.println("Vamos a crear una matriz de " + matriz.length + " filas y " +
matriz[0].length + " columnas");
for (int fila = 0; fila < matriz.length; fila++) {
        for (int columna = 0; columna < matriz[fila].length ; columna++) {
                matriz[fila][columna] = entrada.nextInt();}}
```

Tema 2. Introducción a los Arrays

Haciendo uso de un método:

```
public static int [][] leerEnteros2D() {      // Lectura matriz cuadrada
   Scanner entrada = new Scanner(System.in);
   System.out.println("Introduce el número de filas y columnas de la matriz");
   int dimension = entrada.nextInt();
   int [][]matriz = new int [dimension][dimension];   // numero filas = numero columnas
   System.out.println("Introduce valores enteros en la matriz ");
   for (int i = 0; i < matriz.length; i++) {
     for (int j = 0; j < matriz[i].length; j++) {
         System.out.print("Introduce valor matriz[" + i + "," + j +"] => ");
         matriz[i][j] = entrada.nextInt();
     }}
      return matriz;
}
```

Tema 2. Introducción a los Arrays

Haciendo uso de un método:

```
public static int [][] leerEnteros2D() {      // Lectura matriz no cuadrada
   Scanner entrada = new Scanner(System.in);
   System.out.println("Introduce el número de filas");
   int dimension1 = entrada.nextInt();
   System.out.println("Introduce el número de columnas");
   int dimension2 = entrada.nextInt();
   int [][]matriz = new int [dimension1][dimension2];
   System.out.println("Introduce valores enteros en la matriz ");
   for (int i = 0; i < matriz.length; i++) {
     for (int j = 0; j < matriz[i].length; j++) {
         System.out.print("Introduce valor matriz[" + i + "," + j +"] => ");
         matriz[i][j] = entrada.nextInt();
    }}
   return matriz;}
```

Tema 2. Introducción a los Arrays

Arrays bidimensionales

- **Procesamiento en un array bidimensional (matriz)**

Ejemplo 2. Inicializar la matriz con valores aleatorios entre 0 y 99.

```
int [][] matriz = new int[3][3];
for (int fila = 0; fila < matriz.length; fila++){
        for (int columna = 0; columna < matriz[fila].length ; columna++) {
                matriz[fila][columna] = (int)(Math.random() * 100);
        }
}
```

Tema 2. Introducción a los Arrays

Arrays bidimensionales

- **Procesamiento en un array bidimensional (matriz)**

Ejemplo 3. Mostrar la matriz por pantalla.

```
System.out.println("La matriz creada es ");
for (int fila = 0; fila < matriz.length; fila++) {
        for (int columna = 0; columna < matriz[fila].length; columna++) {
                System.out.print(matriz[fila][columna] + "\t");
        }
        System.out.println();
}
```

Tema 2. Introducción a los Arrays

Arrays bidimensionales

- **Procesamiento en un array bidimensional (matriz)**

Ejemplo 4. Sumar todos los elementos de la matriz.

```
int sumaTotal = 0;
for (int fila = 0; fila < matriz.length; fila++) {
        for (int columna = 0; columna < matriz[fila].length; columna++) {
                sumaTotal += matriz[fila][columna];
        }
}
```

Tema 2. Introducción a los Arrays

Arrays bidimensionales

- **Procesamiento en un array bidimensional (matriz)**

Ejemplo 4. Sumar los elementos de la matriz por columnas.

```
for (int columna = 0; columna < matriz[0].length; columna++) {
        int sumaColumna = 0;
        for (int fila = 0; fila < matriz.length; fila++) {
                sumaColumna += matriz[fila][columna];
        }
        System.out.println("Suma columna " + columna + " = " + sumaColumna);
}
```

Tema 2. Introducción a los Arrays

Arrays bidimensionales

- **Paso de matrices a métodos** ⇒ Cuando se pasa una matriz a un método, se pasa la referencia de la matriz al método
- **Devolver un array desde un método** ⇒ Cuando un método devuelve una matriz, se devuelve su referencia

```
23    public static int suma(int[][] matriz) {
24        int sumaTotal = 0;
25        for (int fila = 0; fila < matriz.length; fila++) {
26            for (int columna = 0; columna < matriz[fila].length; columna++) {
27                sumaTotal += matriz[fila][columna];
28            }
29        }
30        return sumaTotal;
31    }
32 }
```

Tema 2. Introducción a los Arrays

```java
package org.ip.tema02;

import java.util.Scanner;

public class PasandoDevolviendoMatricesMetodos {

    public static void main(String[] args) {
        int[][] matriz = getMatriz();
        System.out.println("\nLa suma de todos los elementos es " + suma(matriz));
    }

    public static int[][] getMatriz() {
        Scanner entrada = new Scanner(System.in);
        int[][] matriz = new int[3][4];
        System.out.println("Introduce " + matriz.length + " filas and "
            + matriz[0].length + " columnas: ");
        for (int i = 0; i < matriz.length; i++)
            for (int j = 0; j < matriz[i].length; j++)
                matriz[i][j] = entrada.nextInt();
        entrada.close();
        return matriz;
    }
```

Tema 2. Introducción a los Arrays

Ejemplo 5. Ejemplo de un método que muestra los valores de una matriz 2D

```
public static void mostrar2D(int [][] matriz) {
        System.out.println("Los valores guardados en la matriz son ");
        for (int i = 0; i < matriz.length; i++) {
          for (int j = 0; j < matriz[i].length; j++) {
                System.out.print(matriz[i][j] + "\t");
           }
           System.out.println();
        }
}
```

Tema 2. Introducción a los Arrays

Arrays bidimensionales

Ejemplo 6. Obtener qué fila tiene el mayor valor de suma.

```
int maxSumaFila = 0;
int indiceMaxSumaFila = 0;
for (int columna = 0; columna < matriz[0].length; columna++) {
        maxSumaFila += matriz[0][columna];
}
for (int fila = 1; fila < matriz.length; fila++) {
        int sumaFilaActual = 0;
        for (int columna = 0; columna < matriz[fila].length; columna++) {
                sumaFilaActual += matriz[fila][columna];
        }
        if (sumaFilaActual > maxSumaFila) {
                maxSumaFila = sumaFilaActual;       indiceMaxSumaFila = fila;
        }
}
System.out.println("Fila " + indiceMaxSumaFila + " tiene la maxima suma de " + maxSumaFila);
```

¡MUCHAS GRACIAS!

UNIVERSIDAD DE ALMERÍA