UT5: ARRAYS Y MATRICES

Contenido

1. ¿Qué es un array?

Un array es una **colección finita de datos del mismo tipo**, que se almacenan en **posiciones consecutivas de memoria** y reciben un **nombre común**.

Ej: Se quieren guardar las notas de los 20 alumnos de una clase.
Gráficamente el array se puede representar de la siguiente forma:

Array (unidimensional) notas:

8.50	6.35	5.75	8.50	· · ·	3.75	6.00	7.40
notas[0]	notas[1]	notas[2]	notas[3]	· · ·	notas[17]	notas[18]	notas[19]

Para acceder a cada elemento del array se utiliza el nombre del array y un índice que indica la posición que ocupa el elemento dentro del array. El índice se escribe entre corchetes.

El primer elemento del array ocupa la posición 0, el segundo la posición 1, etc. En un array de N elementos el último ocupará la posición N-1.
En el ejemplo, notas[0] contiene la nota del primer alumno y notas[19] contiene la del último.

El atributo **length** de un array contiene el tamaño del array independientemente de que tenga el valor por defecto u otro valor.

Los índices deben ser enteros no negativos.

2 Arrays unidimensionales

Para crear un array se deben realizar dos operaciones:

1. Declaración
2. Instanciación

2.1 Declaración de un array

En la declaración se crea la referencia al array. La referencia es el nombre del array en el programa. Se debe indicar el nombre del array y el tipo de datos que contendrá.

De forma general un array unidimensional se puede declarar de cualquiera de estas dos formas:

tipo [] nombreArray; o tipo nombreArray[];

- **tipo:** indica el tipo de datos que contendrá. Un array puede contener elementos de tipo básico o referencias a objetos.
- **nombreArray:** es la referencia al array.

Ej:
int [] ventas; // referencia un array de tipo int llamado ventas
double [] temperaturas; // referencia un array de tipo double llamado temperaturas
String [] nombres; // referencia un array de tipo String llamado nombres

2.2 Instanciar un array

Mediante la instanciación se reserva un bloque de memoria para almacenar todos los elementos del array en posiciones consecutivas de memoria.

La dirección donde comienza el bloque de memoria, donde se almacenará el array, ***se asigna al nombre***. De forma general:

nombreArray = new tipo[tamaño];

- **nombreArray**: es el nombre creado en la declaración.
- **tipo**: indica el tipo de datos que contiene.
- **tamaño**: es el número de elementos del array. Debe ser una expresión entera positiva. El tamaño no se puede modificar durante la ejecución del programa.

- **new**: operador para crear objetos. Mediante new se asigna la memoria necesaria para ubicar el objeto. ***Java implementa los arrays como objetos***.

Ej: ventas = new int[5]; // se reserva memoria para 5 enteros

Lo normal es que la declaración y la instanciación se hagan en una sola instrucción:

tipo [] nombreArray = new tipo[tamaño];

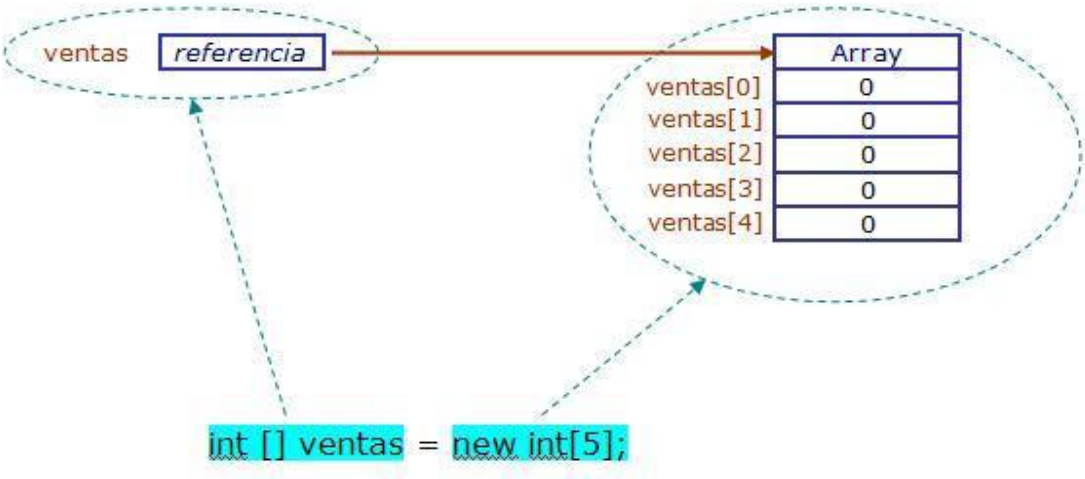
Ej: int [] ventas = new int[5];

El tamaño del array también se puede indicar durante la ejecución del programa, es decir, en tiempo de ejecución se puede pedir por teclado el tamaño del array y crearlo.

Ej:

```
……..
Scanner teclado = new Scanner(System.in);

System.out.print("Número de elementos del array: ");
int numeroElementos = teclado.nextInt();
int [] ventas = new int[numeroElementos];
```

Si no hay memoria suficiente para crear el array, new lanza una excepción java.lang.OutOfMemoryError.



Diferencia entre la referencia y el contenido del array

Debe quedar clara la diferencia entre la referencia (manejador del array o nombre del array) y el contenido del array.

El nombre del array contiene la dirección de memoria del contenido del array.

2.3 Inicializar arrays unidimensionales

Un array es un objeto, por lo tanto, cuando se crea, a sus elementos se les asigna automáticamente un valor inicial. Los valores iniciales por defecto para un array en Java son:

0 para arrays numéricos
'\u0000' (carácter nulo) para arrays de caracteres
false para arrays booleanos
null para arrays de String y de referencias a objetos.

También se pueden dar otros valores iniciales al array cuando se crea. Los valores iniciales se escriben entre llaves separados por comas y deben aparecer en el orden en que serán asignados a los elementos del array. El número de valores determina el tamaño del array.

Ej:

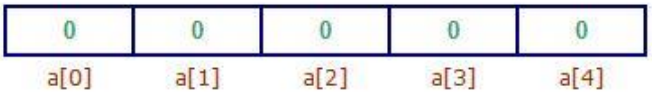
```
double [] notas = {6.7, 7.5, 5.3, 8.75, 3.6, 6.5};

boolean [] resultados = {true, false, true, false};

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

2.4 Acceder a los elementos de un array

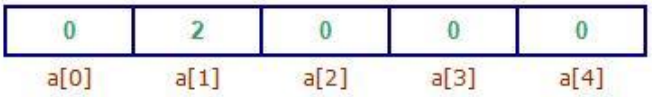
Para acceder a cada elemento del array se utiliza el nombre del array y el índice que indica la posición que ocupa el elemento dentro del array. El índice se escribe entre corchetes. Se puede utilizar como índice un valor entero, una variable de tipo entero o una expresión de tipo entero.

Un elemento de un array se puede utilizar igual que cualquier otra variable. Con los elementos de un array se pueden hacer las mismas operaciones que se pueden hacer con el resto de variables (incremento, decremento, operaciones aritméticas, comparaciones, etc.).

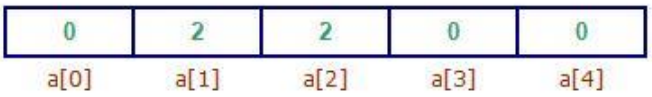
```
int m = 5;
int [] a = new int[5];
```



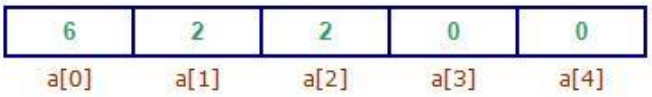

```
a[1] = 2;
```



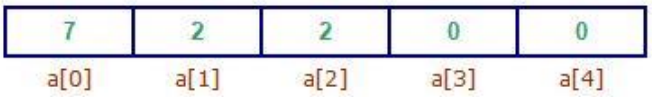

```
a[2] = a[1];
```



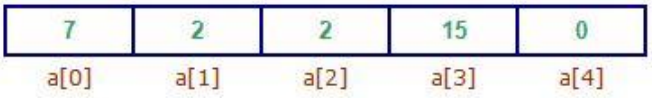

```
a[0] = a[1] + a[2] + 2;
```




```
a[0]++;
```




```
int m = 5;
a[3] = m + 10;
```



Si se intenta acceder a un elemento que está fuera de los límites del array (índice negativo o con un índice mayor que el último elemento del array) el compilador no avisa del error. El error se producirá durante la ejecución. En ese caso se lanza una excepción: **Java.lang.ArrayIndexOutOfBoundsException**.

Se puede saber el número de elementos del array mediante el **atributo length**. Se puede utilizar length (observa que no es un método) para comprobar el rango del array y evitar errores de acceso.

Ej: Para asignar un valor (leído por teclado) a un elemento del array:

```
Scanner teclado = new Scanner(System.in);

int i, valor;
int [] a = new int[10];

System.out.print("Posición: ");
i = teclado.nextInt();

System.out.print("Valor: ");
valor = teclado.nextInt();

if (i>=0 && i < a.length)
        a[i] = valor;
```

2.5 Recorrer un array unidimensional

Para recorrer un array se utiliza una instrucción iterativa (normalmente una instrucción for, aunque también puede hacerse con while o do..while) usando una variable entera como índice, que tomará valores desde el primer elemento al último o desde el último al primero.

Ej:

```
double[] notas = {2.3, 8.5, 3.2, 9.5, 4, 5.5, 7.0};
for (int i = 0; i < notas.length; i++)
        System.out.print(notas[i] + " ");
```

Ej: Programa que lee por teclado la nota de los alumnos de una clase y calcula la nota media del grupo. También muestra los alumnos con notas superiores a la media. El número de alumnos se lee por teclado.

```
import java.util.*;
public class Ejemplo {
    public static void main(String[] args) {
        Scanner teclado= new Scanner(System.in);
        int numAlum, i;
        double suma = 0, media;

        do {
            System.out.print("Número de alumnos de la clase: ");
            numAlum = Integer.parseInt(teclado.nextLine());
        } while (numAlum <= 0);

        double[] notas = new double[numAlum];

        System.out.print("Empezamos a pedir datos de los alumnos: ");

        for (i = 0; i<notas.length; i++) {
            System.out.print("Alumno " + (i + 1) + " Nota final: ");
            notas[i] = teclado.nextDouble();
        }

        for (i = 0; i < notas.length; i++)
            suma = suma + notas[i];

        media = suma / notas.length;
        System.out.printf("Nota media del curso: %2f %n", media);
        System.out.println("Listado de notas superiores a la media: ");
```

```
            for (i = 0; i < notas.length; i++)
                if (notas[i] > media)
                    System.out.println("Alumno numero " + (i + 1)+ "
                    Nota final: " + notas[i]);

            teclado.close();

        }
}
```

2.6 Arrays unidimensionales de caracteres en Java

Un array de caracteres en Java se crea de forma similar a un array de cualquier otro tipo de datos.

Ej: Array de 8 caracteres llamado cadena. Por defecto se inicializa con el carácter nulo.

```
char [] cadena = new char[8];
```

\u0000	\u0000	\u0000	\u0000	\u0000	\u0000	\u0000	\u0000
cadena[0]	cadena [1]	cadena [2]	cadena [3]	cadena [4]	cadena [5]	cadena [6]	cadena [7]

Ej: Array de 5 caracteres llamado vocales. Se asignan valores iniciales: a, e, i, o, u

```
char [] vocales = {'a', 'e', 'i', 'o', 'u'};
```

a	e	i	o	u
vocales[0]	vocales [1]	vocales [2]	vocales [3]	vocales [4]

A diferencia de los demás arrays, **se puede mostrar el contenido completo de un array de caracteres mediante una sola instrucción print o printf**.

Ej:

```
System.out.println(cadena); // Muestra 8 caracteres nulos (en blanco)
System.out.println(vocales); // Muestra aeiou
```

El **atributo length** de un array de caracteres contiene el tamaño del array independientemente de que sean caracteres nulos u otros caracteres.

Ej: char [] cadena = new char[8];

\u0000	\u0000	\u0000	\u0000	\u0000	\u0000	\u0000	\u0000
cadena[0]	cadena [1]	cadena [2]	cadena [3]	cadena [4]	cadena [5]	cadena [6]	cadena [7]

System.out.println(cadena.length); // Muestra: 8
cadena[0] ='m';
cadena[1] ='n';

m	n	\u0000	\u0000	\u0000	\u0000	\u0000	\u0000
cadena[0]	cadena [1]	cadena [2]	cadena [3]	cadena [4]	cadena [5]	cadena [6]	cadena [7]

System.out.print(cadena);
System.out.print(cadena);
System.out.println(".");

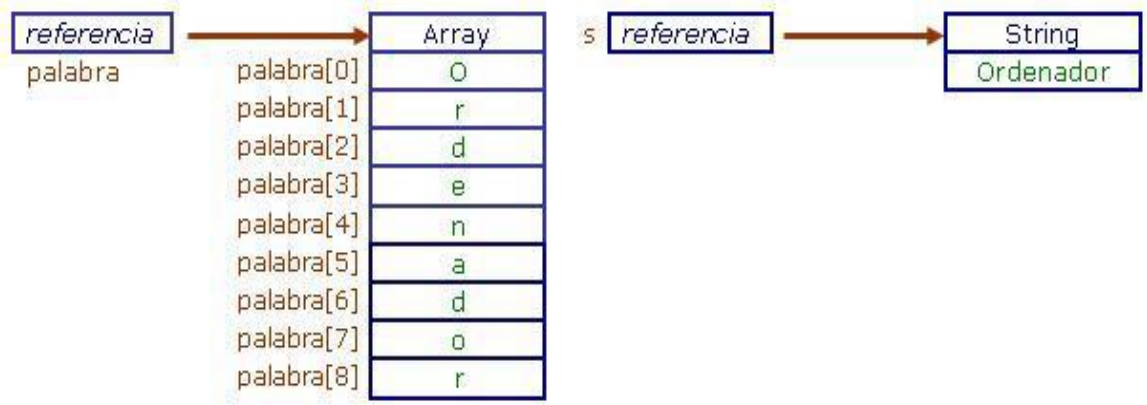
Muestra: mnbbbbbbmnbbbbbb. //b representa los espacios en blanco

Se puede **asignar un String a un array de caracteres** mediante el método **toCharArray()** de la clase String.

Ej: String s = "Ordenador";

char [] palabra = s.toCharArray();

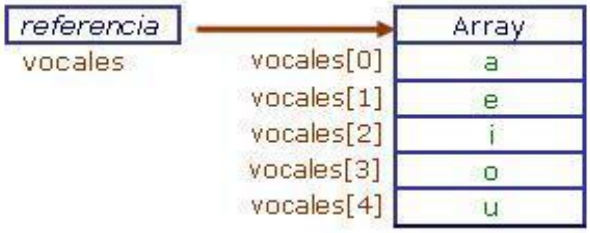
Se crea un nuevo array de caracteres con el contenido del String s y se asigna la dirección de memoria a la variable palabra.



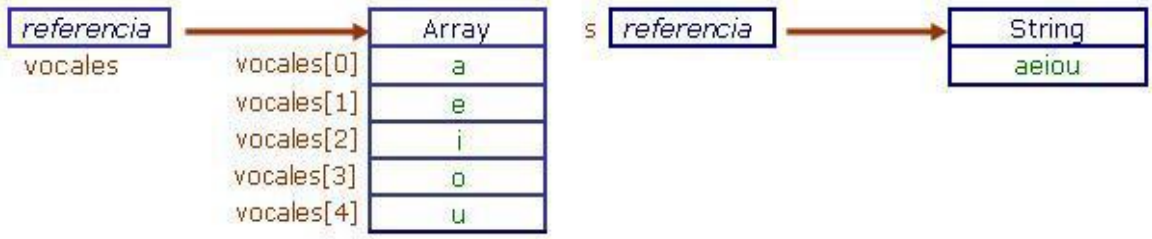
También se puede crear un String a partir de un array de caracteres.

Ej: char [] vocales = {'a', 'e', 'i', 'o', 'u'};

String s = new String(vocales);



Se crea un nuevo String con el contenido del array vocales y se asigna la dirección de memoria a s.



Recorrer un array de caracteres unidimensional:

Se puede recorrer de forma completa utilizando una instrucción iterativa, normalmente un for.

Ej: char [] s = new char[10];
s[0]='a';
s[1]='b';
s[2]='c';

```
for(int i = 0; i<s.length; i++)
        System.out.print(s[i]+ " "); //Muestra todos los caracteres del array,
                                     // incluidos los nulos.
```

Si los caracteres no nulos se encuentran todos al principio del array se puede recorrer hasta que nos encontremos un carácter nulo:

Ej:
```
char [] s = new char[10];
s[0]='a';
s[1]='b';
s[2]='c';
int i = 0;
while(s[i] != '\0'){
        System.out.print(s[i]); // Muestra los caracteres del array hasta que
        i++; // encuentra el primer nulo.
} // Ojo, que si está lleno del todo da error
```

3. Arrays bimensionales (Matrices)

Un array puede tener más de una dimensión. El caso más general son los arrays bidimensionales, también llamados matrices o tablas. **La dimensión de un array la determina el número de índices necesarios para acceder a sus elementos.**

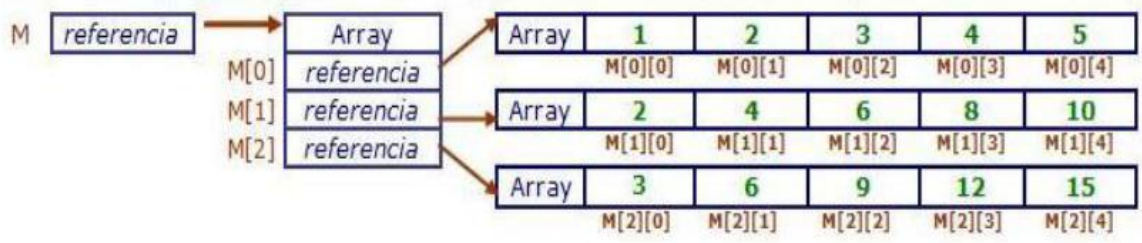
Una matriz necesita ***dos índices*** para acceder a sus elementos. Gráficamente se puede representar una matriz como una tabla de n filas y m columnas cuyos elementos son todos del mismo tipo.

La siguiente figura representa un array M de 3 filas y 5 columnas:

	0	1	2	3	4
0	1	2	3	4	5
1	2	4	6	8	10
2	3	6	9	12	15

Pero en realidad **una matriz en Java es un array de arrays**.

Gráficamente, podemos representar la disposición real en memoria del array anterior así:



La longitud del array M (M.length) es 3. Se corresponde con el número de filas del array bidimensional.

La longitud de cada fila del array (M[i].length) es 5. Se corresponde con el número de columnas de cada fila del array bidimensional.

Para acceder a cada elemento de la matriz se utilizan **dos índices**. El primero indica la fila y el segundo la columna.

Se crean de forma similar a los arrays unidimensionales, añadiendo un índice.

Ej:
Matriz de datos de tipo int llamado ventas de 4 filas y 6 columnas.

```
int [][] ventas = new int[4][6];
```

Matriz de datos double llamado temperaturas de 3 filas y 4 columnas.

```
double [][] temperaturas = new double[3][4];
```

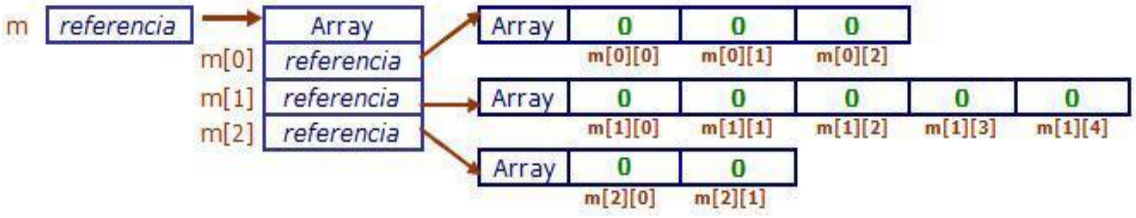
En Java se pueden crear arrays irregulares en los que el número de elementos de cada fila es variable. Solo es obligatorio indicar el número de filas.

Ej: int [][] m = new int[3][]; //crea una matriz m de 3 filas.

A cada fila se le puede asignar un número distinto de columnas:

```
m[0] = new int[3];
m[1] = new int[5];
m[2] = new int[2];
```

Gráficamente podemos representar la disposición real en memoria del array anterior así:



3.1 Inicializar matrices

Una matriz es un objeto, por tanto, cuando se crea, a sus elementos se les da automáticamente un valor inicial, al igual que a los arrays unidimensionales.

- 0 para arrays numéricos
- '\u0000' (carácter nulo) para arrays de caracteres
- false para arrays booleanos
- null para arrays de String y de referencias a objetos.

También es posible dar otros valores iniciales a la matriz cuando se crea. Los valores iniciales se escriben entre llaves separados por comas. Los valores que se le asignen a cada fila aparecerán a su vez entre llaves separados por comas.
El número de valores determina el tamaño de la matriz.

Ej: int [][] numeros = {{6,7,5}, {3, 8, 4}, {1,0,2}, {9,5,2}};
Se crea la matriz numeros de tipo int, de 4 filas y 3 columnas, y se le da esos valores iniciales.

3.2 Recorrer matrices

Para recorrer una matriz **se anidan dos bucles for** (también se puede con otros).

En general, para recorrer un array multidimensional se anidan tantas instrucciones for como dimensiones tenga el array.

Se usa siempre length para obtener el número de columnas que tiene cada fila

```
for (i = 0; i < a.length; i++) { //número de filas
        for (j = 0; j < a[i].length; j++) //número de columnas de cada fila
        System.out.print(a[i][j] + " ");
        System.out.println();
}
```

Ej: Programa que lee por teclado números enteros y los guarda en una matriz de 5 filas y 4 columnas. A continuación, muestra los valores leídos, el mayor y el menor y las posiciones que ocupan.

```
import java.util.*;
public class Bidimensional1 {
        public static void main(String[] args) {
                final int FILAS = 5, COLUMNAS = 4;
                Scanner teclado = new Scanner(System.in);
                int i, j, mayor, menor;
                int filaMayor, filaMenor, colMayor, colMenor;
                int[][] A = new int[FILAS][COLUMNAS];

                System.out.println("Leemos e insertamos elementos en la matriz: ");
                for (i = 0; i < FILAS; i++)
                        for (j = 0; j < COLUMNAS; j++) {
                                System.out.print("A[" + i + "][" + j + "]= ");
                                A[i][j] = Integer.parseInt(teclado.nextLine());
                        }
                System.out.println("Leemos los valores introducidos en la matriz:");
                for (i = 0; i < A.length; i++) {
                        for (j = 0; j < A[i].length; j++)
                                System.out.print(A[i][j] + " ");
                                System.out.println();
                }
```

```
                mayor = A[0][0]; //se toma el primero como mayor y menor
                menor = A[0][0];
                filaMayor = filaMenor = colMayor = colMenor = 0;
                for (i = 0; i < A.length; i++) {
                        for (j = 0; j < A[i].length; j++) {
                                if (A[i][j] > mayor) {
                                        mayor = A[i][j];
                                        filaMayor = i;
                                        colMayor = j;
                                } else if (A[i][j] < menor) {
                                        menor = A[i][j];
                                        filaMenor = i;
                                        colMenor = j;
                                }
                        }
                }

                teclado.close();
                System.out.print("Elemento mayor: " + mayor);
                System.out.println(" Fila: "+ filaMayor + " Columna: " +  colMayor);
                System.out.print("Elemento menor: " + menor);
                System.out.println(" Fila: "+ filaMenor + " Columna: " +  colMenor);
        }
}
```