

UD3.- Arrays

1.- Introducció

ARRAY = VECTOR = ARREGLO

Un **array** és un tipus de dada capaç d'emmagatzemar múltiples valors. S'utilitza per a agrupar dades molt paregudes, per exemple, si tinguérem la necessitat d'emmagatzemar la temperatura mitjana diària a La Vall d'Uixó durant l'últim any es poden utilitzar les variables temp0, temp1, temp2, temp3, ... i així fins a temp364, (365 variables diferents) però seria poc pràctic.

Sols per a introduir les dades de cada dia ja necessitaríem 365 línies de codi, a més a més, si volem traure la temperatura mitjana de tot l'any, sols per calcular-la tindríem una tremenda línia de codi.

En lloc de crear 365 variables seria molt millor crear un **array** de grandària 365 (és com si tinguérem una sola variable que pot emmagatzemar diversos valors).



Gràcies als **array** es pot crear un conjunt de variables amb el mateix nom. La diferència serà que un número (índex de l'**array**) distingirà a cada variable.

Un **array** o **vector** és una col·lecció de valors d'un mateix tipus dins d'una mateixa variable. De manera que es pot accedir a cada valor independentment.

2.- Propietats dels Arrays

Algunes propietats dels **arrays** són:

- Els **arrays** s'utilitzen com a contenidors per a emmagatzemar dades relacionades (en lloc de declarar variables per separat per a cadascun dels elements del **arrays**).
- Totes les dades incloses a l'**arrays** són del mateix tipus. Es poden crear **arrays** d'enters de tipus **int** o de reals de tipus **double**, però en un mateix **array** no es poden mesclar tipus de dades, per ex. **int** i **double**.
- La grandària del **array** s'estableix quan es crea el **array** (amb l'operador **new**, igual que qualsevol altre objecte).
- Als elements del **array** s'accedirà a través de la **posició (índex)** que ocupen dins del conjunt d'elements de l'**array**.

- **L'índex del primer element sempre es 0. El vector sempre comença en la posició 0.** (Si té un tamany 5, les posicions o índex són **0,1,2,3,4**)  

- Es poden declarar arrays a qualsevol tipus de dades (enters, booleans, doubles, ... i fins i tot objectes).
- Els **arrays** unidimensionals es coneixen amb el nom de **vectors**.
- Els **arrays** bidimensionals es coneixen amb el nom de **matrius**.

3.- Arrays Unidimensionals (Vectors)

Un array és un objecte en Java i com a tal. S'ha de **declarar** i s'ha d'**instanciar**.

Instanciar en Java significa crear un objecte d'una classe, o el que es el mateix, generar un exemplar d'eixa classe.

Per exemple, la **classe** *persona* definiria totes les característiques generals d'una persona, i a partir d'ella crearíem **objectes** o instàncies com *Jaume*, *Isabel*, *Pepe*, que són casos concrets de persones (de la classe *persona*).

- **DECLARACIÓ:** Un array es pot declarar de dos maneres.

```
tipus identificador[];                int arrayEnters[];
tipus[] identificador;                double[] arrayReals;
```

- **INSTANCIACIÓ:** Utilitzant l'operador **new** i indicant el tamany o grandària que tindrà el nostre vector

```
Identificador = new tipus[tamany];    arrayEnters = new int[5];
arrayReals = new double[10];
```

- El més habitual és **declarar i instanciar** el vector en una sola línia.

```
tipus identificador[] = new          int arrayEnters[] = new
tipus[] identificador = new tipus[tamany] double[] arrayReals = new
```

En els exemples anteriors es declara un **array** de tipus **int** i un altre de tipus **double**. Aquesta declaració indica per a què servirà l'**array**, però no reserva espai en la memòria RAM al no saber-se encara la grandària d'aquest. Per tant no podem utilitzar l'**array** fins que no l'instanciem. Quan s'usa **new**, és quan es reserva l'espai necessari en memòria. Un **array** no inicialitzat és un **array null** (sense valor).

- **ASSIGNACIÓ DE VALORS**

Els valors del l'**array** s'assignen (emmagatzemen) utilitzant l'**índex** del mateix entre **claudàtors**. Per exemple, per a emmagatzemar el valor 8 en la tercera posició d'un array escriuríem:

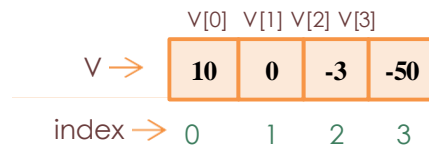
```
int[] v = new int[4]; // Declaració i creació d'un array de longitud 4
v[0] = 10; // Assignació de valors
v[1] = 0;
v[2] = -3;
v[3] = -50;
```

També es poden assignar valors a l'array en la pròpia declaració i instanciació:

```
int v[] = {10, 0, -3, -50,};
int v2[] = new int[]{10, 0, -3, -50,}; //Equivalent a l'anterior
```

En tots els casos anteriors, estem declarant un array de 4 elements:

```
v[0] = 10;
v[1] = 0;
v[2] = -3;
v[3] = -50;
```



A Java (com en altres llenguatges) **el primer element d'un array esta a la posició zero**. El primer element de l'array v, és v[0].

Cada element de l'array actua com una variable qualsevol, amb la diferència que està indexada dins l'array. Per exemple:

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

```
num[0] = 8;
```

```
num[1] = 33;
```

```
num[2] = 20;
```

```
num[3] = 15;
```

```
num[4] = 11;
```

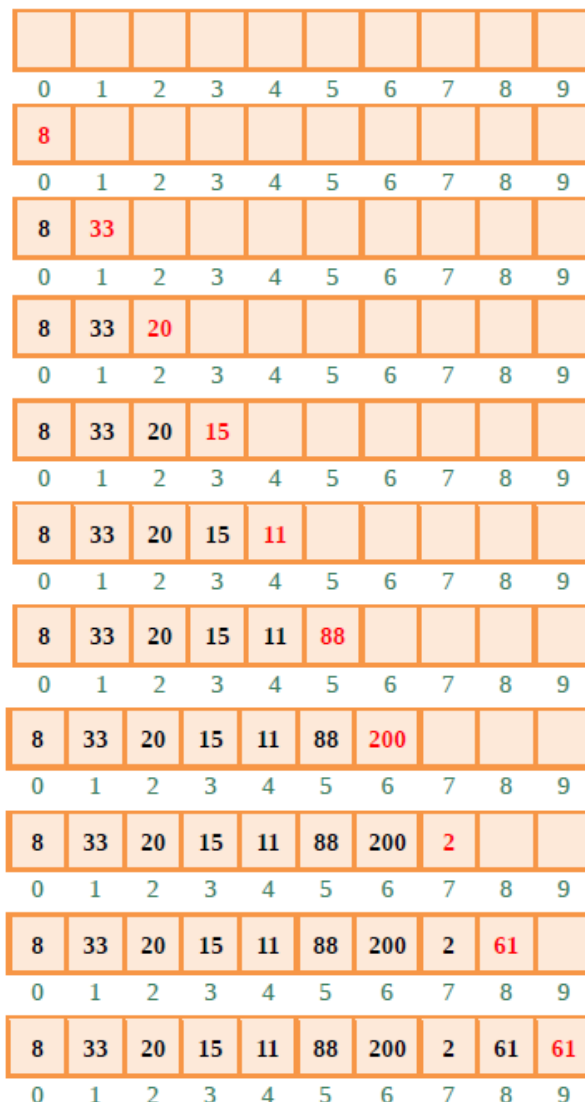
```
num[5] = 88;
```

```
num[6] = num[2] * 10;
```

```
num[7] = num[2] / 10;
```

```
num[8] = num[0] + num[1] + num[2];
```

```
num[9] = num[8];
```



Exemple:

```
public class Arrays1 {
    public static void main(String[] args) {

        int[] v; // definim v como un array de enters

        v = new int[4]; // reservem espai per a 4 enters
                       //(Creació de l'objecte array)

        v[0] = 10;    // Assignació de valors
        v[1] = 0;
        v[2] = -3;
        v[3] = -50;

        // int v[] = new int[]{10, 0, -3, -50,}; Delaració equivalent

        System.out.print("Els valors de l'Array són els següents: ");

        System.out.print(v[0] + ", " + v[1] + ", " + v[2] + ", " + v[3]);

        int suma = v[0] + v[3];

        System.out.println("\nEl primer element del array mes l'ultim sumen " + suma);
    }
}
```

Console

```
<terminated> Arrays1 [Java Application] C:\Program Files\Java\jdk-14.0.1
Els valors de l'Array són els següents: 10, 0, -3, -50
El primer element del array mes l'ultim sumen -40
```

En resum:

Tipo de dato de los elementos del vector

Nombre del vector

Número de elementos del vector

Asignación de valores

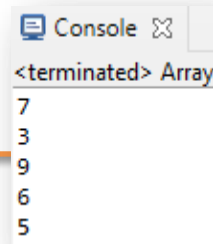
```
int[] notas = new int[7];
notas[0] = 14;
```

4.- Recórrer un Array. Longitud d'un vector

Per a recórrer un vector (accedir a tots els seus elements) sempre **serà necessari un bucle**.

En el següent exemple declarem i instanciem un vector tipus `int` amb les notes d'un alumne i després utilitzem un bucle `for` per a recórrer el vector i mostrar tots els elements.

```
public class Arrays2 {  
    public static void main(String[] args) {  
        // Declarem i instanciem un vector de tipus int  
        int notes[] = new int[] {7, 3, 9, 6, 5};  
        // Array de grandària 5. Els seus elements estaran en les posicions de 0 a 4  
        // Recorrem el vector des d'i=0 fins a <i 5 (és a dir, des de 0 fins a 4)  
        for (int i = 0; i < 5; i++) {  
            System.out.println(notes[i]);  
        }  
    }  
}
```



```
<terminated> Array  
7  
3  
9  
6  
5
```

Els arrays posseeixen una propietat anomenada `length` que indica la seva grandària. Esta propietat ens retorna un **int**.

Podem utilitzar-la amb la sintaxi: `nomVector.length`

Per exemple:

```
int notes[] = new int[5];  
System.out.println(notes.length);
```

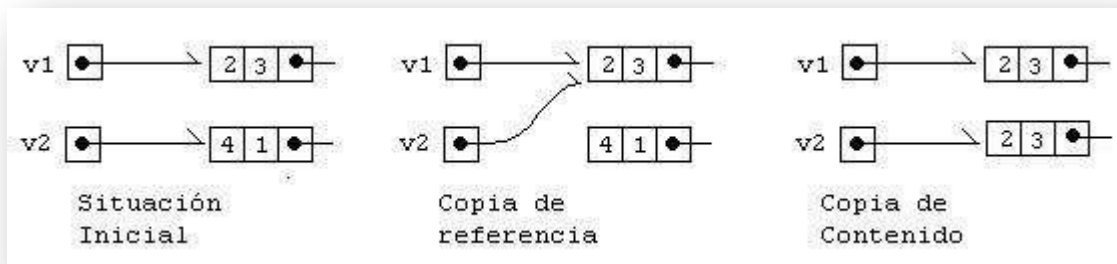
Si el vector té com en l'exemple 5 elements, la propietat `length` ens retornarà el valor enter 5, però el seu primer element es troba en `notes[0]` i l'últim en `notes[4]`, o el que és el mateix l'últim element és `notes[length - 1]`.

```
public class Arrays3 {  
    public static void main(String[] args) {  
        // Declarem i instanciem un vector de tipus int  
  
        int notes[] = new int[5];  
        notes[0] = 7;  
        notes[1] = 3;  
        notes[2] = 9;  
        notes[3] = 6;  
        notes[4] = 5;  
  
        for (int i = 0; i < notes.length; i++)  
            System.out.println(notes[i]);  
  
        // Declarem suma i mitja  
        int suma = 0;  
        int mitjana;  
  
        // Recorrem el vector des de 0 fins a 4, acumulant les notes en suma  
        for (int i = 0; i <= notes.length-1; i++)  
            suma += notes[i];    // Equival a: suma = suma + notes[i]  
  
        // Calculem la mitjana i la mostrem per pantalla  
        mitjana = suma / notes.length;  
  
        System.out.println("La nota mitjana és: " + mitjana);  
    }  
}
```

5.- Còpia de Vectors

Per a copiar vectors no n'hi ha prou amb igualar un vector a un altre com si fora una variable simple.

Si partim de dos vectors `v1`, `v2` i férem `v2=v1`, el que ocorreria seria que `v2` apuntaria a la posició de memòria de `v1`. Això és el que es denomina un copia de referència:



Si per exemple volem copiar tots els elements del vector `v2` en el vector `v1`, existeixen dues formes per a fer-ho:

- **Copiar els elements un a un**

```
for (int i = 0; i < v1.length; i++)
    v2[i] = v1[i];
```

- **Utilitzar la funció `arraycopy`**

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

`v_origen`: Vector origen

`i_origen`: Posició inicial de la còpia

`v_destí`: Vector de destí

`i_destin`: Posició final de la còpia

`length`: Quantitat d'elements a copiar

```
System.arraycopy(v1, 0, v2, 0, v1.length);
// Copiem tots els elements de v1 en v2
```


6.- La Classe Arrays

En el paquet java.util es troba una classe estàtica anomenada **Arrays**. Esta classe estàtica permet ser utilitzada com si fora un objecte (com ocorre amb Math). Esta classe posseeix mètodes molt interessants per a utilitzar sobre arrays.

```
import java.util.Arrays;
```

El seu ús és:

Arrays.mètode(arguments);

Alguns **mètodes** són:

- **fill**: permet omplir tot un array unidimensional amb un **determinat valor**. Els seus arguments són l'array a emplenar i el valor desitjat:
Per exemple , omplir un array de 23 elements sencers amb el valor -1

```
int valors[] = new int[23];
Arrays.fill(valors,-1); // Emmagatzema -1 en tot el *array 'valors'

// També permet decidir des que índex fins a quin índex emplenem:
Arrays.fill(valors,5,8,-1); // Emmagatzema -1 des del 5é a la 7é element
```

- **equals** : Compara dos arrays i retorna **true** si són iguals (**false** en cas contrari). Es consideren iguals si són del mateix tipus, grandària i contenen els mateixos valors.

```
int valorsA[] = new int[5];
int valorsB[] = new int[5];
boolean sonIguals;

sonIguals = Arrays.equals(valorsA, valorsB); // retorna true si els arrays són iguals
```

- **sort** : Permet ordenar un array en ordre ascendent. Es poden ordenar només una sèrie d'elements des d'un determinat punt fins a un determinat punt.

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

Arrays.sort(x); // Ordena x de menor a major
Arrays.sort(x,2,4); // Ordena x només des del 2n al 4t element
```

- **binarySearch** : Permet buscar un element de manera ultraràpida en un array **ordenat**. Retorna l'índex en el qual està col·locat l'element buscat. Exemple:

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

Arrays.sort(x);
System.out.println(Arrays.binarySearch(x,8)); //tornaria 7
```