



Unió Europea

Fons Social Europeu

L'FSE inverteix en el teu futur



GENERALITAT
VALENCIANA



UF06. - ARRAYS

- Teoria -

PROGRAMACIÓ
CFGS DAM

Autor:

José Manuel Martí Fenollosa

Revisat per:

Àngel Olmos Giner

a.olmosginer@edu.gva.es

2022/2023

ARRAYS

ÍNDEX DE CONTINGUT

1. INTRODUCCIÓ
2. PROPIETATS
3. VECTORS (Arrays Unidimensionals)
4. LA CLASE ARRAYS
5. LA CLASE STRING
6. MATRIUS (Arrays Multidimensionals)

1. INTRODUCCIÓ

INTRODUCCIÓ

Un array o vector



Col·lecció de valors d'un mateix tipus
dins d'una mateixa variable

Es pot accedir a cada valor independentment

Els utilitzarem, per exemple, per manejar moltes variables que es refereixen a dades similars

A més a Java, un array es un objecte que té propietats que es poden manipular

1. INTRODUCCIÓ

INTRODUCCIÓ

Per exemple: *cal emmagatzemar les notes d'una classe amb 18 alumnes i calcular la nota mitjana.*

OPCIÓ 1: *Caldria crear 18 variables, emmagatzemar les 18 notes, calcular la mitjana d'eixes 18 variables...*

OPCIÓ 2: ***En lloc de crear 18 variables seria molt millor crear un array de grandària 18 (és com si tinguérem una sola variable que pot emmagatzemar diversos valors).***

Gràcies als arrays es pot crear un conjunt de variables amb el mateix nom

La diferència serà que un número (índex del array) distingirà a cada variable

ARRAYS

ÍNDEX DE CONTINGUT

1. INTRODUCCIÓ
2. **PROPIETATS**
3. VECTORS (Arrays Unidimensionals)
4. LA CLASE ARRAYS
5. LA CLASE STRING
6. MATRIUS (Arrays Multidimensionals)

2. PROPIETATS

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

ARRAYS

ÍNDEX DE CONTINGUT

1. INTRODUCCIÓ
2. PROPIETATS
3. **VECTORS (Arrays Unidimensionals)**
4. LA CLASE ARRAYS
5. LA CLASE STRING
6. MATRIUS (Arrays Multidimensionals)

3. VECTORS (Arrays Unidimensionals)

3.1 Declaració

Un array es declara de manera similar a una variable simple però afegint claudàtors []

Un Vector es pot declarar de dues formes:

tipus identificador[];

tipus[] identificador;

Tipus de dada dels elements del vector

Nom de la variable

Exemples:

int notes[];

double comptes[]; //Hem declarat un array de tipus int i un altre de tipus double

Aquesta declaració indica per a què servirà el array, però no reserva espai en la memòria RAM al no saber-se encara la grandària d'aquest. **Encara no pot utilitzar-se el array, falta instanciarlo.**

3. VECTORS (Arrays Unidimensionals)

3.2 Instància

Un vegada declarat l'array, es té que ➡ Utilitzarem l'operador **new** (ací és quan es reserva l'espai en memòria)

Un array no inicialitzat és un array **null** (sense valor)

Exemple:

`int notes[]; // Declarem 'notes' com array de tipus int` \equiv `int notes[] = new int[5];`
`notes = new int[5]; // Instanciem 'notes' a grandària 5`

S'acaba de crea un array de cinc enters (es crea en memòria el array i s'inicialitzen els valors, els números s'inicialitzen a 0)

3. VECTORS (Arrays Unidimensionals)

3.3 Emmagatzematge

Els **valors** del array s'assignen (emmagatzemen) utilitzant l'**índex** del mateix **entre claudàtors**

El primer element del vector sempre estarà en la posició o índex 0

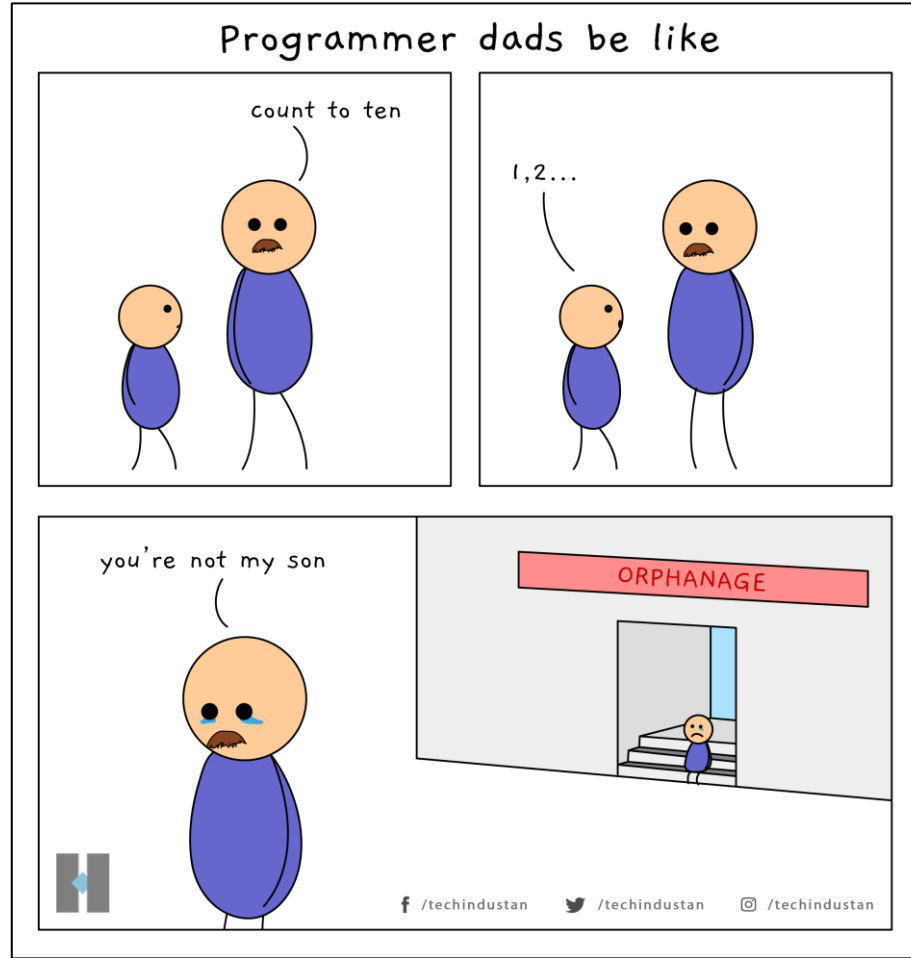
Índexs →	0	1	2	3	4
Valors →	8	10	2	3	5

Exemple: per a **emmagatzemar el valor 2 en la 3^a posició del array** escriuríem:

`notes[2] = 2;`

3. VECTORS (Arrays Unidimensionals)

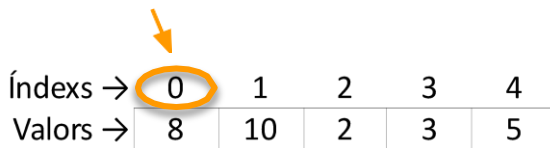
3.3 Emmagatzematge



3. VECTORS (Arrays Unidimensionals)

3.3 Emmagatzematge

El primer element del array notes, és notes[0].



Índexs →	0	1	2	3	4
Valors →	8	10	2	3	5

Exemple: Així es **declara, instancia i emmagatzema (declara i inicialitza)** al mateix temps un array de 5 elements: `int notes[] = new int[] {8, 10, 2, 3, 5};`

`int notes[] = {8, 10, 2, 3, 5};` *//Equivalent a l'anterior*

← **Fixa't!** D'aquesta manera podem **NO** utilitzar **new**.

L'exemple seria equivalent a:

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

```
notes[0] = 8;
```

```
notes[1] = 10;
```

```
notes[2] = 2;
```

```
notes[3] = 3;
```

```
notes[4] = 5;
```

3. VECTORS (Arrays Unidimensionals)

3.4 Longitud d'un vector

La propietat **length** indica la grandària d'un array

Exemple:

```
int notes[] = new int[5]; // Declara i instància vector tipus int de grandària 5
```

```
System.out.println(notes.length); // Mostrarà un 5
```

← El **primer element** es troba en **notes[0]** i l'**últim** en **notes[4]**.

3. VECTORS (Arrays Unidimensionals)

3.5 Recórrer d'un vector

Per a recórrer un vector = accedir a tots els seus elements ➡ **Serà necessari ... ?**

3. VECTORS (Arrays Unidimensionals)

3.5 Recórrer d'un vector

Per a recórrer un vector = accedir a tots els seus elements ➡ Serà necessari un bucle



3. VECTORS (Arrays Unidimensionals)

3.5 Recórrer d'un vector

Per a recórrer un vector = accedir a tots els seus elements ➡ Serà necessari un bucle

Exemple: declarem i instanciem un **vector de tipus int** amb les notes d'un alumne i després utilitzem un **bucle for per a recórrer el vector** i mostrar tots els elements.

```
int notas[] = new int[] {7, 3, 9, 6, 5};    // Declarem, instanciem i inicialitzem vector notas de tipus int  
  
for (int i = 0; i < notas.length; i++) {    // Com el vector és de grandària 5 els seus elements estaran en  
    System.out.println(notas[i]);           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)
```

DIY

3. VECTORS (Arrays Unidimensionals)

3.5 Recórrer d'un vector

Exemple: Ara calcularem la nota mitjana i la mostrarem per pantalla:

DIY

3. VECTORS (Arrays Unidimensionals)

3.5 Recórrer d'un vector

Exemple: Ara calcularem la nota mitjana i la mostrarem per pantalla:

```
int suma = 0; int mitjana;                                // Declarem suma i mitjana

for (int i = 0; i < notas.length; i++) {                  // Recorrem el vector des de 0 fins a 4, acumulant les
    summa += notas[i];                                     notes en suma
}                                                         // Equival a: suma = summa + notas[i]

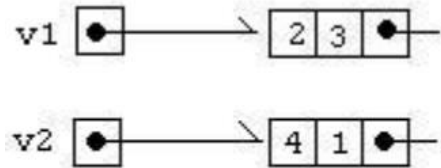
mitjana = summa / notas.length;                            // Calculem la mitjana i la mostrem per pantalla
System.out.println("La nota mitjana és: " + mitjana);
```

3. VECTORS (Arrays Unidimensionals)

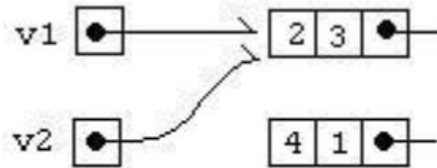
3.6 Còpia de vectors

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

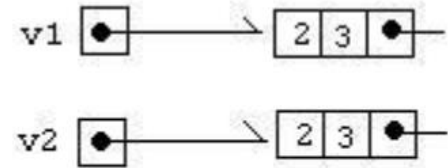
Si partirem de dos vectors $v1$ i $v2$, i férem $v2=v1$, el que ocurreria seria que $v2$ apuntaria a la posició de memòria de $v1$. Això és el que es denomina un **copia de referència**:



Situación
Inicial



Copia de
referencia



Copia de
Contenido

3. VECTORS (Arrays Unidimensionals)

3.6 Còpia de vectors

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 (i = 0; i < v1.length; i++){  
    v2[i] = v1[i];  
}
```

- Utilitzar la funció **arraycopy**

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

// Copiem tots els elements de v1 en v2

System.arraycopy(v_origen, i_origen, v_destí, i_destí, length);

v_origen: Vector origen

i_origen: Posició inicial de la còpia de v_origen

v_destí: Vector destino

i_destí: Posició inicial de la còpia en v_destí

length: Quantitat d'elements a copiar

EXERCICIS PROPOSATS
VECTORS fins a l'exercici 13



ARRAYS

ÍNDIX DE CONTINGUT

1. INTRODUCCIÓ
2. PROPIETATS
3. VECTORS (Arrays Unidimensionals)
4. **LA CLASE ARRAYS**
5. LA CLASE STRING
6. MATRIUS (Arrays Multidimensionals)

4. LA CLASSE ARRAYS

DEFINICIÓ

- En el paquet *java.util* es troba una **classe estàtica** anomenada ***Arrays***
- Permet ser utilitzada com si fora un objecte sense necessitat d'instanciar-lo (com ocorre amb *Math*)
- Aquesta classe posseeix **mètodes** molt interessants per a utilitzar sobre arrays

Arrays.mètode(arguments);

4. LA CLASSE ARRAYS

DEFINICIÓ

Alguns mètodes interessants són:

- **fill**: permet emplenar tot un array unidimensional amb un determinat valor
- **equals**: compara dos arrays i retorna *true* si són iguals (mateix tipus, grandària i mateixos valors).

Retorna *false* en cas contrari

- **sort**: ordena un array en ordre ascendent. Es poden ordenar només una sèrie d'elements des d'un determinat punt fins a un determinat punt
- **toString**: converteix l'array de valors en un String (útil per a mostrar per pantalla)

4. LA CLASSE ARRAYS

Exemples

fill:

Exemple 1: ompli un array de 10 elements sencers amb el valor -1

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

```
Arrays.fill(valors,-1); // Emmagatzema -1 en tot l'array 'valors'
```

Exemple 2: permet decidir des que índex fins a quin índex emplenem:

```
Arrays.fill(valors,5,8,-2); // Emmagatzema -2 des del 5é la 7é element (NO agafa el 8é)
```

```
public static void main(String[] args) {  
    int valors[] = new int[10];  
    Arrays.fill(valors,-1); // Emmagatzema -1 en tot el array 'valors'  
    System.out.println(Arrays.toString(valors));  
    Arrays.fill(valors,5,8,-2); // Modifica a -2 del 5é al 7é  
    System.out.println(Arrays.toString(valors));  
}
```

```
[-1, -1, -1, -1, -1, -1, -1, -1, -1, -1]  
[-1, -1, -1, -1, -1, -2, -2, -2, -1, -1]  
BUILD SUCCESSFUL (total time: 0 seconds)
```

4. LA CLASSE ARRAYS

Exemples

sort:

Exemple: ordenar arrays (ordre ascendent)

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

```
Arrays.sort(x);
```

// Ordena x de menor a major

```
Arrays.sort(x,2,7);
```

// Ordena x només des de 2^{on} al 6^e element (**NO agafa el 7é**)

```
public static void main(String[] args) {  
  
    int x[]={4,5,2,3,8,7,2,3,9,5};  
    System.out.println(Arrays.toString(x));  
    Arrays.sort(x);                // Ordena x de menor a major  
    System.out.println(Arrays.toString(x) + "\n");  
  
    int y[]={4,5,2,3,8,7,2,3,9,5};  
    System.out.println(Arrays.toString(y));  
    Arrays.sort(y,2,7);            // Ordena y només des de 2on al 4rt element  
    System.out.println(Arrays.toString(y));  
}
```



run:

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



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



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



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

BUILD SUCCESSFUL (total time: 0 s)

EXERCICIS PROPOSATS
Classe Array fins a l'exercici
17



ARRAYS

ÍNDIX DE CONTINGUT

1. INTRODUCCIÓ
2. PROPIETATS
3. VECTORS (Arrays Unidimensionals)
4. LA CLASE ARRAYS
5. **LA CLASE STRING**
6. MATRIUS (Arrays Multidimensionals)

5. LA CLASSE STRING

DEFINICIÓ

Les **cadenes de text** han de manejar-se creant objectes de tipus **String**

Exemple:

```
String text1 = "Prova de text!";
```

Les cadenes poden ocupar diverses línies utilitzant l'operador de **concatenació "+"**:

```
String text2 = "Aquest és un text que ocupa " +  
"diverses línies, no obstant això es pot " +  
"perfectament encadenar i almacenar" +  
" en un sol objecte de tipus String";
```

També es poden **crear objectes *String* sense utilitzar constants entrecomillades**:

```
char[] paraula = {'P', 'a', 'r', 'a', 'u', 'l', 'a'}; // Array de chars  
String cadena = new String(paraula);
```

5. LA CLASSE STRING

5.1 Comparació

Els objectes String NO poden comparar-se directament amb els operadors de comparació == com les variables simples.

S'han d'utilitzar aquests mètodes:

- ***s1.equals(s2)***. El resultat és true si la cadena1 és igual a la cadena2. **Recorda, Java és Case-Sensitive**
- ***s1.equalsIgnoreCase(s2)***. Com l'anterior, però no es tenen en compte majúscules i minúscules
- ***s1.compareTo(s2)***. Compara totes dues cadenes, considerant l'ordre alfabètic:
 - Si la primera cadena és major en ordre alfabètic que la segona, retorna la diferència positiva entre una cadena i una altra
 - Si són iguals retorna 0
 - Si és la segona la major, retorna la diferència negativa entre una cadena i una altra

L'ordre no és el de l'alfabet espanyol, sinó que usa la taula [ASCII](#) (p.e.: lletra ñ és molt major que l'o)
- ***s1.compareToIgnoreCase(s2)***. Igual que l'anterior, només que a més ignora les majúscules

5. LA CLASSE STRING

5.1 Comparació

```
public static void main(String[] args) {  
    String s1 = "prova de text!";  
    String s2 = "Prova de Text!";  
    System.out.print("Comprobemos si son iguales:");  
    System.out.println(s1.equals(s2));  
    System.out.print("Comprobemos si son iguales, ahora sin contar con las mayúsculas:");  
    System.out.println(s1.equalsIgnoreCase(s2));  
    System.out.print("Comparémoslas considerando el orden alfabético:");  
    System.out.println(s1.compareTo(s2)); /*Ordenant alfabèticament, i segons el codi ASCII, comparem la  
                                           'p' amb la 'P', seguint la diferència de 32 posicions*/  
  
    System.out.print("Comparémoslas considerando el orden alfabético, ahora sin contar con las mayúsculas:");  
    System.out.println(s1.compareToIgnoreCase(s2));  
}
```



run:



Comprobemos si son iguales:false

Comprobemos si son iguales, ahora sin contar con las mayúsculas:true

Comparémoslas considerando el orden alfabético:32



Comparémoslas considerando el orden alfabético, ahora sin contar con las mayúsculas:0

BUILD SUCCESSFUL (total time: 0 seconds)

5. LA CLASSE STRING

5.2 Mètodes més utilitzats

Són **funcions que posseeixen los propis objectes de tipus *String***. Per a utilitzar-los n'hi ha prou amb posar el nom del mètode i els seus paràmetres després del nom de l'objecte *String*.

objecteString.mètode(arguments);

Nom de
l'objecte *String*

Nom del
mètode

Paràmetres
del mètode

5. LA CLASSE STRING

5.2 Mètodes més utilitzats

**(a més a més dels mètodes de comparació ja vistos)*

Mètodes més utilitzats*:

valueOf : Convertix valors que no són de cadena a forma de cadena.

```
String numero = String.valueOf(1234); // Converteix el número int 1234 en l'String "1234"
```

length : Retorna la longitud d'una cadena (el nombre de caràcters de la cadena):

```
String text1="Prova";  
System.out.println(text1.length()); // Escriu un 5
```

Concatenar cadenes : Es pot fer de dues formes, utilitzant el mètode *concat* o amb l'operador *+*.

```
String s1= "Bon ", s2= " dia", s3, s4; s3 = s1 + s2;  
s4 = s1.concat(s2); //En tots dos casos el contingut de s3 i s4 seria el mateix: "Bon dia"
```

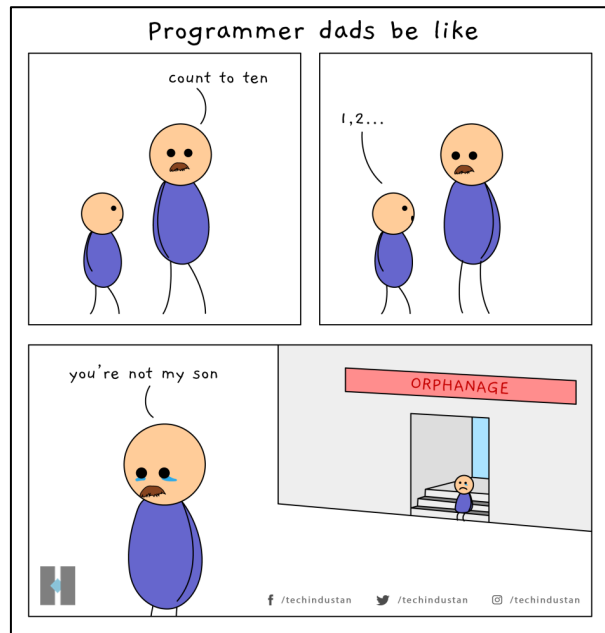
5. LA CLASSE STRING

5.2 Mètodes més utilitzats

charAt : Retorna un caràcter concret de la cadena segons la seua posició (el primer caràcter està en la posició 0)

```
String s1="Prova";
```

```
char c1 = s1.charAt(2); // c1 valdrà 'o'
```



Si la posició és negativa o sobrepassa la grandària de la cadena, ocorre un error d'execució, una excepció tipus ***IndexOutOfBoundsException*** (recorda aquest tipus d'error, es repetirà moltes vegades)

5. LA CLASSE STRING

5.2 Mètodes més utilitzats

substring : Dona com a resultat una porció del text de la cadena. La porció es pren des d'una posició inicial fins a una posició final (**sense incloure aqueixa posició final**) o des d'una posició fins al final de la cadena.

```
String s1="Bon dia";
```

```
String s2=s1.substring(0,3);    // s2 = "Bon"
```

```
String s3=s1.substring(3);    // s3=" dia"
```

Si les posicions indicades no són vàlides ocorre una excepció de tipus ***IndexOutOfBoundsException***.
Es comença a comptar des de la posició 0.

5. LA CLASSE STRING

5.2 Mètodes més utilitzats

indexOf : Retorna la primera posició en la qual apareix un determinat text en la cadena. En el cas que la cadena buscada no es trobe, retorna -1. El text a buscar pot ser *char* o *String*.

```
String s1="Volia dir-te que vull que et vages";
```

```
System.out.println(s1.indexOf("que")); // Retorna 13
```

També es pot buscar des d'una determinada posició:

```
String s1="Volia dir-te que vull que et vages";
```

```
System.out.println(s1.indexOf("que",14)); // Ara retornaria 22
```

5. LA CLASSE STRING

5.2 Mètodes més utilitzats

lastIndexOf : Retorna l'última posició en la qual apareix un determinat text en la cadena. És quasi idèntica a l'anterior, només que cerca des del final

```
String s1="Volia dir-te que vull que et vages";
```

```
System.out.println(s1.lastIndexOf("que")); // Retornaria 22
```

També permet començar a buscar des d'una determinada posició

5. LA CLASSE STRING

5.2 Mètodes més utilitzats

endsWith : Retorna *true* si la cadena acaba amb un determinat text.

```
String s1="Volia dir-te que vull que et vages";  
System.out.println(s1.endsWith("vages")); // Retornaria true
```

startsWith : Retorna *true* si la cadena comença amb un determinat text.

```
String s1="Volia dir-te que vull que et vages";  
System.out.println(s1.startsWith("vages")); // Retornaria false
```

5. LA CLASSE STRING

5.2 Mètodes més utilitzats

replace: Canvia totes les aparicions d'un caràcter (o caràcters) per un altre/s en el *String* que s'indique i l'emmagatzema com a resultat. L'string original no canvia, pel que cal assignar el resultat de *replace* a un *String* per a emmagatzemar el text canviat.

Exemple1

```
String s1="Papallona";
```

```
System.out.println(s1.replace('a', 'e')); // Retorna ???
```

```
System.out.println(s1); // Retorna ???
```

DIY

5. LA CLASSE STRING

5.2 Mètodes més utilitzats

replace: Canvia totes les aparicions d'un caràcter (o caràcters) per un altre/s en el *String* que s'indique i l'emmagatzema com a resultat. L'string original no canvia, pel que cal assignar el resultat de *replace* a un *String* per a emmagatzemar el text canviat.

Exemple1

```
String s1="Papallona";  
System.out.println(s1.replace('a', 'e')); // Retorna ???  
System.out.println(s1); // Retorna ???
```

DIY

Per a guardar el valor hauríem de fer:

```
String s2 = s1.replace('a','e');
```

Exemple2

```
String s1="Buscar armadillos";  
System.out.println(s1.replace("ar","er"));  
System.out.println(s1);
```


5. LA CLASSE STRING

5.2 Mètodes més utilitzats

toUpperCase / toLowerCase : Obté la versió en majúscules / minúscules de la cadena.

```
String s1 = "Batalló de cigonyes però no amb ñ";
```

```
System.out.println(s1.toUpperCase()); //Escriu: BATALLÓ DE CIGONYES PERÒ NO AMB Ñ
```

toCharArray : Aconseguir un array de caràcters a partir d'una cadena. D'aquesta forma podem utilitzar les característiques dels arrays per a manipular el text, la qual cosa pot ser interessant per a manipulacions complicades.

```
String s="text de prova";
```

```
char c[]=s.toCharArray();
```

```
System.out.println(c[3]); //retorna la lletra 't'
```

```
System.out.println(s); //retorna el text sencer "text de prova"
```

5. LA CLASSE STRING

5.2 Mètodes més utilitzats

format : Modifica el format de la cadena a mostrar. Molt útil per a mostrar només els decimals que necessitem d'un nombre

```
System.out.println(String.format("%.2f", number)); // Mostra el float amb dos decimal
```

- “%” marca l'inici de la part sencera seguit del nombre de dígit a mostrar (res si es vol deixar per defecte)
- “.” marca l'inici de la part decimal seguit del nombre de dígit a mostrar
- “f” determina que es mostra un nombre de tipus *float*
- Les variables s'inclouen per ordre d'apariació després de la “,”

matches : Examina l'expressió regular que rep com a paràmetre (en forma de *String*) i retorna *true* si el text que examina compleix l'expressió regular.

Una expressió regular és una expressió textual que utilitza símbols especials per a fer cerques avançades.

```
System.out.println("ejemplo4: "+cadena.matches(".*[xyz].*"));
```

5. LA CLASSE STRING

5.2 Mètodes més utilitzats

matches : Examina l'expressió regular que rep com a paràmetre (en forma de *String*) i retorna vertader si el text que examina compleix l'expressió regular

Les expressions regulars poden contindre:

- **Caràcters:** Com **a**, **s**, **ñ**,... i els interpreta tal qual. Si una expressió regular continguera només un caràcter (p.e. 'g'), *matches* retornaria vertader si el text conté només aqueix caràcter 'g'
- **Caràcters de control:** **\n, **
- **Opcions de caràcters:** Es posen entre claudàtors. Per exemple **[abc]** significa 'a', 'b' o 'c'
- **Negació de caràcters:** Funciona a l'inrevés, impedeix que apareguen els caràcters indicats. Es posa amb claudàtors dins dels quals es posa el caràcter circumflex (^). **[^abc]** significa ni 'a' ni 'b' ni 'c'

5. LA CLASSE STRING

5.2 Mètodes més utilitzats

matches : Examina l'expressió regular que rep com a paràmetre (en forma de *String*) i retorna vertader si el text que examina compleix l'expressió regular

- **Rangs:** Es posen amb guions. Per exemple **[a-z]** significa qualsevol caràcter de la 'a' a la 'z'
- **Intersecció:** Usa &&. Per exemple **[a-x&&r-z]** significa de la 'r' a la 'x' (intersecció de totes dues expressions)
- **Sostracció:** Exemple **[a-x&&[^cde]]** significa de la 'a' a la 'x' excepte la 'c', 'd' o 'e'
- **Qualsevol caràcter:** Es fa amb el símbol punt '.'
- **Opcional:** El símbol **?** serveix per a indicar que l'expressió que li antecedeix pot aparèixer una o cap vegades. Per exemple **a?** indica que pot aparèixer la lletra 'a' o no

5. LA CLASSE STRING

5.2 Mètodes més utilitzats

matches : Examina l'expressió regular que rep com a paràmetre (en forma de *String*) i retorna vertader si el text que examina compleix l'expressió regular

- **Repetició:** S'usa amb l'asterisc '*'. Indica que l'expressió pot repetir-se diverses vegades o fins i tot no aparéixer mai
- **Repetició obligada:** Ho fa el signe '+'. L'expressió es repeteix una o més vegades (però almenys una)
- **Repetició un nombre exacte de vegades:** Un número entre claus '{ }' indica les vegades que es repeteix l'expressió. Per exemple `\d{7}` significa que el text ha de portar set números (set xifres del 0 al 9). Amb una ',' significa almenys, és a dir `\d{7,}` significa almenys set vegades (podria repetir-se més vegades). Si apareix un segon número indica un màxim nombre de vegades `\d{7,10}` significa de 7 a 10 vegades

5. LA CLASSE STRING

5.2 Mètodes més utilitzats

matches : Examina l'expressió regular que rep com a paràmetre (en forma de *String*) i retorna vertader si el text que examina compleix l'expressió regular

```
String cadena="Solo se que no se nada";

System.out.println("ejemplo1: "+cadena.matches("Solo"));

System.out.println("ejemplo2: "+cadena.matches("Solo.*"));

System.out.println("ejemplo3: "+cadena.matches(".*[qnd].*"));

System.out.println("ejemplo4: "+cadena.matches(".*[xyz].*"));
```

5. LA CLASSE STRING

5.2 Mètodes més utilitzats

matches : Examina l'expressió regular que rep com a paràmetre (en forma de *String*) i retorna vertader si el text que examina compleix l'expressió regular

```
System.out.println("ejemplo4: "+cadena.matches(".*[^xyz].*"));
```

```
System.out.println("ejemplo5: "+cadena.matches("So?lo se qu?e no se na?da"));
```

```
System.out.println("ejemplo6: "+cadena.matches("[a-z].*"));
```

```
System.out.println("ejemplo7: "+cadena.matches("[A-Z].*"));
```

5. LA CLASSE STRING

5.2 Mètodes més utilitzats

matches : Examina l'expressió regular que rep com a paràmetre (en forma de *String*) i retorna vertader si el text que examina compleix l'expressió regular

```
String cadena2="abc1234";

System.out.println("ejemplo8: "+cadena2.matches("[abc]+.*"));

System.out.println("ejemplo9: "+cadena2.matches("[abc]+\\d{4}"));

System.out.println("ejemplo10: "+cadena2.matches("[abc]+\\d{1,10}"));}
```


5. LA CLASSE STRING

5.2 Mètodes més utilitzats

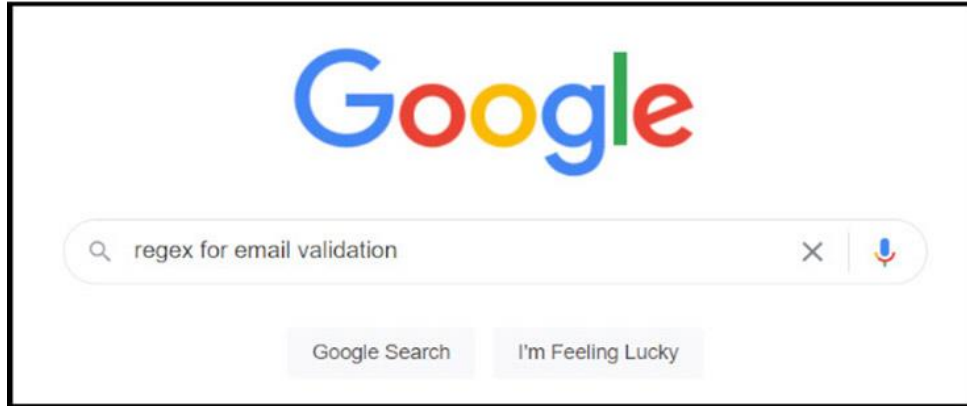
Regex: regular expressions



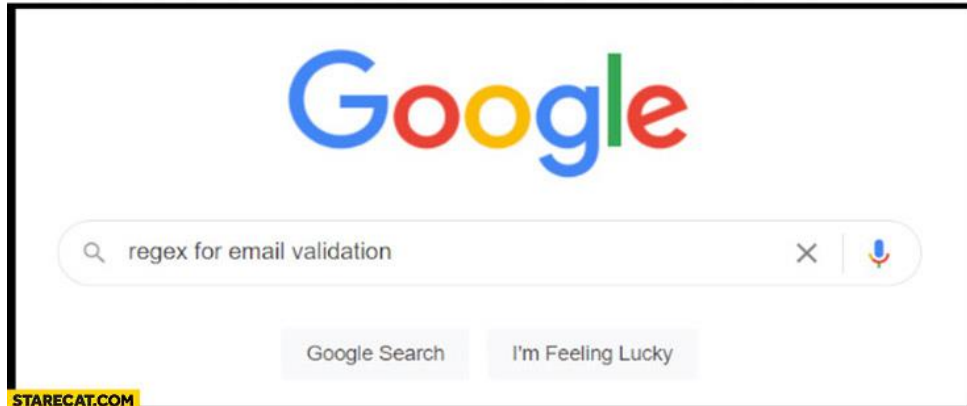
<https://www.oracle.com/technical-resources/articles/java/regex.html>

https://www.w3schools.com/java/java_regex.asp

DAY1 OF PROGRAMMING



10 YEARS OF PROGRAMMING



5. LA CLASSE STRING

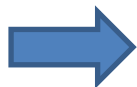
5.3 Lectura amb Scanner

Com ja sabem, la lectura d'un *String* utilitzant la classe *Scanner* es realitza amb el mètode *nextLine()*:

```
Scanner in = new Scanner(System.in);  
String s = in.nextLine();
```

Si llegim un tipus de dada numèrica, sencer per exemple, abans de llegir un *String* haurem de netejar el buffer d'entrada, en cas contrari llegirà el valor '\n' (salt de línia) introduït després del número i li ho assignarà a la variable *String*, amb el que no es llegirà bé l'entrada.

SOLUCIÓ



```
Scanner in = new Scanner(System.in);  
System.out.print("Introdueix un número: ");  
int n = in.nextInt();  
in.nextLine(); // Netegem el buffer d'entrada  
System.out.print("Introdueix un String: ");  
String s = in.nextLine();
```

EXERCICIS PROPOSATS

Classe String (B)



ARRAYS

ÍNDEX DE CONTINGUT

1. INTRODUCCIÓ
2. PROPIETATS
3. VECTORS (Arrays Unidimensionals)
4. LA CLASE ARRAYS
5. LA CLASE STRING
6. **MATRIUS (Arrays Multidimensionals)**

6. MATRIUS (Arrays Multidimensionals)

DEFINICIÓ

Els més utilitzats són els **arrays de 2 dimensions**, coneguts com a **matrius**.

Es defineixen de les següents formes:

```
tipus identificador[][];  
tipus[][] identificador;
```

Exemple: **declarem i instanciem un array de 3 x 3** (3 files x 3 columnes).

```
double preus[][] = new int[3][3];
```

Accedim als seus valors utilitzant **dobles claudàtors**.

```
preus[0][0] = 7.5;
```

```
preus[0][1] = 12;
```

```
preus[0][2] = 0.99;
```

```
preus[1][0] = 4.75;
```

// etc.

```
double [][] preus = {{7.5,12,0.9},{4.75, 8, 4.5}, {10,8.5,6.7}};
```

		Columnes		
		0	1	2
Files	0	(0,0)	(0,1)	(0,2)
	1	(1,0)	(1,1)	(1,2)
	2	(2,0)	(2,1)	(2,2)

6. MATRIUS (Arrays Multidimensionals)

DEFINICIÓ

Exemple: declarem i instanciem un array de 3 files x 6 columnes per a emmagatzemar les notes de 3 alumnes (la fila correspon a un alumne, i cada columna a les notes d'aquest alumne):

```
int notes[][] = new int[3][6]; // És equivalent a 3 vectors de grandària 6
```

Suposant que les notes ja estan emmagatzemades, mostrar les notes per pantalla:

```
for (int i = 0; i < notes.length; i++) {  
    System.out.print("Notes de l'alumne " + i + ": "); // Per a cada fila (alumne)  
    for (int j = 0; j < notes[i].length; j++) {  
        System.out.print(notes[i][j] + " "); // Per a cada columna (nota)  
    }  
}
```

Cal tindre en compte que com té **dues dimensions** necessitem **un bucle niat** (un per a les files i un altre per a les columnes de cada fila)



EXERCICIS PROPOSATS
Matrius (C)



Autor:

José Manuel Martí Fenollosa

Revisat per:

Àngel Olmos Giner

Llicència:



[CC BY-NC-SA 3.0 ES](#) Reconeixement – No Comercial – Compartir Igual (by-nc-sa)

No es permet un ús comercial de l'obra original ni de les possibles obres derivades, la distribució de les quals s'ha de fer amb una llicència igual a la que regula l'obra original. Aquesta és una obra derivada de l'obra original de José Manuel Martí Fenollosa