



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria Informàtica



Tema 7. Arrays: definició i aplicacions

Introducció a la Informàtica i a la Programació (IIP)

Curs 2019/20

Departament de Sistemes Informàtics i Computació



Continguts

Duració: 9 sessions

1. Introducció

- La necessitat d'utilitzar arrays per a representar dades del mateix tipus
- Arrays en Java: declaració, creació, atribut length i operador []
- Accés a les components: directe i seqüencial
- Ús d'un array com atribut, variable local, paràmetre o resultat d'un mètode

2. Representació i tractament de dades usant un array

- Formes bàsiques de representació, en funció de l'accés directe o seqüencial a les dades, i operacions elementals associades
- Recorregut d'un array: esquemes, exemples i exercicis
- Cerca d'una dada donada en un array: esquemes, exemples i exercicis

3. Arrays multidimensionals: declaració, creació i accés a les components



- **Crea** una carpeta **Tema 7** dins de la teua carpeta **W:\IIP**
- **Descarrega** (del Tema 7 de PoliformaT) els fitxers **BlueJ exemplesT7.jar** i **exercicisT7.jar**.
- Des de l'opció **Projecte** de **BlueJ**, **obre'ls** amb l'opció **Open ZIP/JAR...** i prepara't per usar-los.

Introducció

- Sovint interessa emmagatzemar i referenciar variables que representen una **col·lecció de valors**, per poder tractar-los de manera uniforme.
- Exemples:
 - Obtenir estadístiques bàsiques sobre mesures diàries de la temperatura mitjana en una determinada zona geogràfica.
 - Gestionar una col·lecció d'objectes homogenis, per exemple, una classe `Hospital` que pugui tenir associat un conjunt d'objectes de tipus `Pacient`.
 - ...
- Java proporciona l'*array* com a mecanisme per agrupar dades de tipus homogeni (tant de tipus objecte com de tipus primitiu).

Arrays unidimensionals

- **Definició:** Un **array** és una col·lecció d'elements **homogenis** (del mateix tipus de dades) agrupats de forma consecutiva en memòria.
- **Característiques**
 - Cada **element** d'un array té associat un **índex**, que és un nombre no negatiu que l'identifica inequívocament i permet accedir-hi.
 - L'**amplària** (nombre de components) de l'array s'ha d'establir en la seua declaració i és **invariable** al llarg de l'execució.
 - L'accés als elements d'un array és **directe** (gràcies al seu índex).
- Aquestes **estructures de dades** són adequades per a situacions en les que l'accés a les dades es realitze de forma aleatòria (és a dir, conjunts de dades que poden ser indexats) o seqüencial (posicionalment, una dada darrere de l'altra).

Arrays unidimensionals

Declaració i ús

- **Valors:** Es representen com una successió d'elements entre claus i separats per comes.

$$\{e_0, e_1, \dots, e_{n-1}\}$$

- **Operador d'arrays:** Operador d'accés a les components []

`nomVble[índex]`

índex és una expressió que s'avalua a un valor vàlid per l'array `nomVble`.

- **Declaració i creació:**

Vàlid però no aconsellat

- Declaració: `tipus[] nomVble`; ó ~~`tipus nomVble[]`~~;
- Inicialització (amb components): `nomVble = new tipus[amplària]`;
- Conjuntament: `tipus[] nomVble = new tipus[amplària]`;

`tipus` és qualsevol tipus primitiu o referència, `nomVble` és qualsevol identificador vàlid i `amplària` és una expressió que s'avalua a un enter no negatiu i determina la quantitat de components.

Arrays unidimensionals

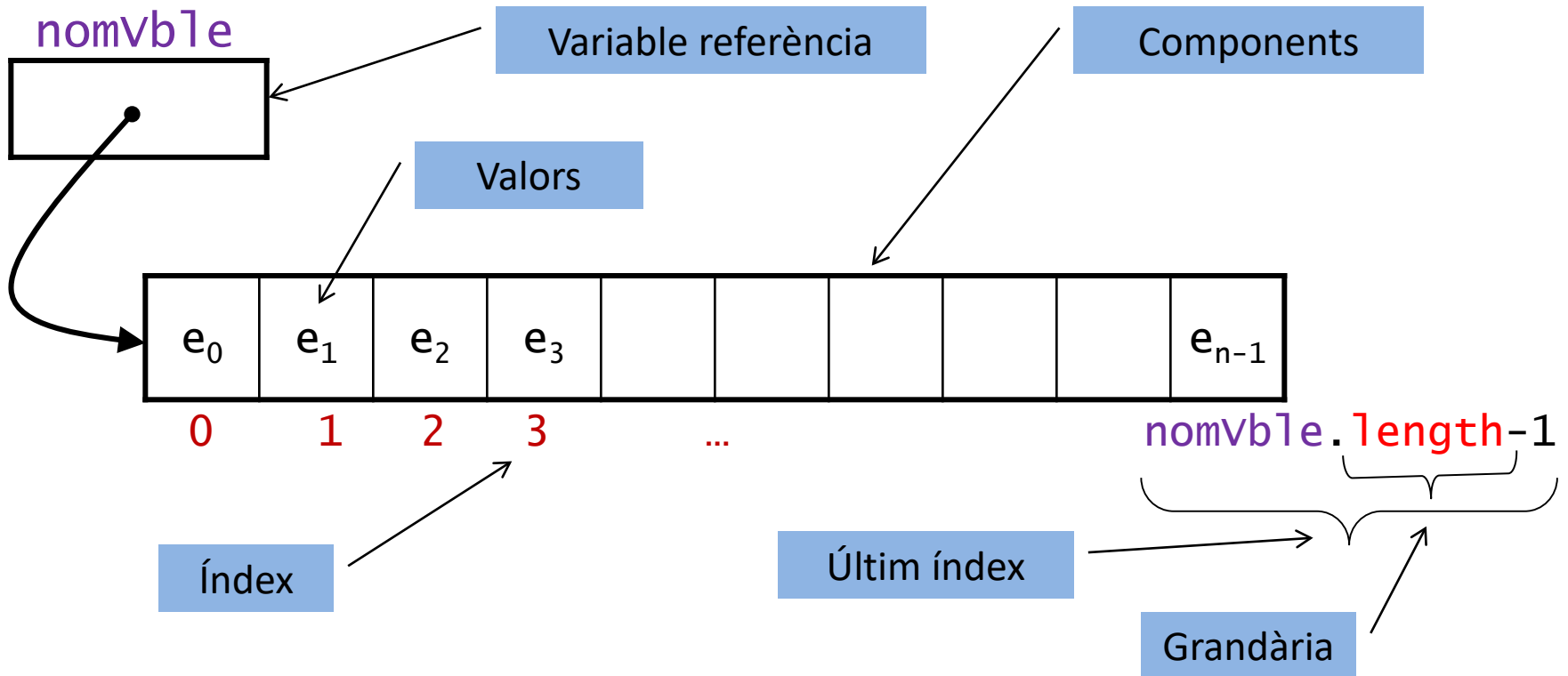
Declaració i ús

- Tots els arrays disposen d'un atribut constant (de només consulta) **length** que indica la quantitat de components (**amplària** o **grandària**) d'un array:
`nomVble.length`
- Els índexs vàlids d'un array van des de 0 fins **length** - 1.
- Una variable array es pot veure com una referència a la posició de memòria del primer element de l'array.
- Hi ha una referència especial que es pot utilitzar amb qualsevol array (o amb altres tipus referència): **null**.
- Tots els components d'arrays de **tipus numèrics** s'inicialitzen per defecte a **0**.
- Tots els components d'arrays de **tipus referència** (p.e., un array de String) s'inicialitzen per defecte a **null**.

Arrays unidimensionals

Declaració i ús

- Gràficament podem veure un array de la manera següent :



on $n = \text{nomVble.length}$ i tots els e_i , $0 \leq i < n$, són del mateix tipus.

Arrays unidimensionals

Declaració i ús

The screenshot shows the BlueJ IDE interface. The top menu bar includes 'Projecte', 'Edita', 'Eines', 'Vegeu', and 'Ajuda'. On the left, there are buttons for 'Nova classe...', a right arrow, and 'Compila'. The main workspace displays a class hierarchy with boxes for 'PasParametres', 'CopiaArraysPrimitius', 'Comptadors', 'Conjunt', and 'GrupB'. A thought bubble points to the 'llista' variable, stating: 'llista representa un **agregat** de 6 (sub)variables de tipus **int**'.

On the left sidebar, a variable card for 'llista' is shown with the type 'int[]'. The main editor displays the following code:

```
int[] llista = new int[6];  
llista  
    <object reference> (int[])  
llista.length  
    6 (int)  
llista[3]  
    0 (int)
```

A red inspection window for 'llista : int[]' is open on the right. It contains a table of the array's state:

int length	6
[0]	0
[1]	0
[2]	0
[3]	0
[4]	0
[5]	0

The inspection window also includes buttons for 'Inspecciona', 'Obté', 'Mostra camps estàtics', and 'Tanca'.

Arrays - Excepcions

- En un array d'amplària **N**, els índexs vàlids per accedir als seus components pertanyen a l'interval $[0, N-1]$.
- Un accés fora d'aquest interval produeix l'excepció en execució del tipus :
java.lang.ArrayIndexOutOfBoundsException
- L'accés als components d'un array s'ha de controlar utilitzant les fites inferior (**0**) i superior (**nomVble.length-1**) dels índexs.

llista:
int[]

```
int[] llista = new int[6];
```

```
llista
```

```
<object reference> (int[])
```

```
llista.length
```

```
6 (int)
```

```
llista[3]
```

```
0 (int)
```

```
llista[6]
```

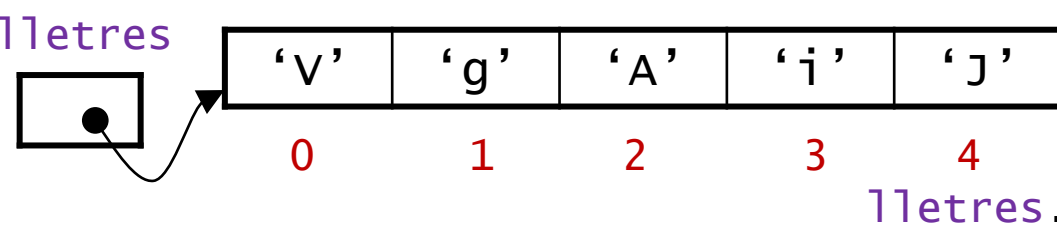
```
Exception: java.lang.ArrayIndexOutOfBoundsException (Index 6 out of bounds for length 6)
```

```
llista[-1]
```

```
Exception: java.lang.ArrayIndexOutOfBoundsException (Index -1 out of bounds for length 6)
```

Arrays unidimensionals

Declaració i ús - Exemples

- Array de 5 caràcters:
 - Variable referència: `char[] lletres;`
 - Objecte array (amb components): `lletres = new char[5];`
 - Conjuntament: `char[] lletres = new char[5];`
- 
- `lletres[0] = 'v';`
`lletres[1] = 'g';`
`lletres[2] = 'A';`
`lletres[3] = 'i';`
`lletres[4] = 'j';`
- Dos arrays d'enters:
`int[] nums1 = new int[5000], nums2 = new int[50];`
 - Array de 20 cadenes de caràcters: final `int NUM = 10;`
`String[] noms = new String[NUM * 2];`
 - Array de reals tal que la quantitat de components es decideix per teclat:
`double[] preus = new double[teclat.nextInt()];`
 - Creació i inicialització d'un array de 4 enters: `int[] v = {-5, 6, 10, 3};`

Arrays - Memòria

- Una variable referència array (**nomVble**) utilitza la memòria on haja estat definida (p.e. al registre de activació del mètode on s'ha definit).
- Els components de l'array es creen en el **monticle** o **heap** que és la zona de memòria reservada per guardar variables dinàmiques.
- Les **variables dinàmiques** són variables que es creen en temps d'execució mitjançant l'operació de creació **new**.
- Els components només són accessibles si hi ha una referència a ells.
- És possible suggerir la destrucció d'un array **v** al desreferenciar-lo (**v = null**)
 - Recorda que hi ha un mecanisme de recollida de fem (**garbage collector**) que s'activa automàticament en temps d'execució i allibera la memòria que està ocupada però no és accessible al no estar referenciada.
 - També es pot suggerir la seua execució mitjançant la instrucció **System.gc()**.

Arrays unidimensionals - Assignació

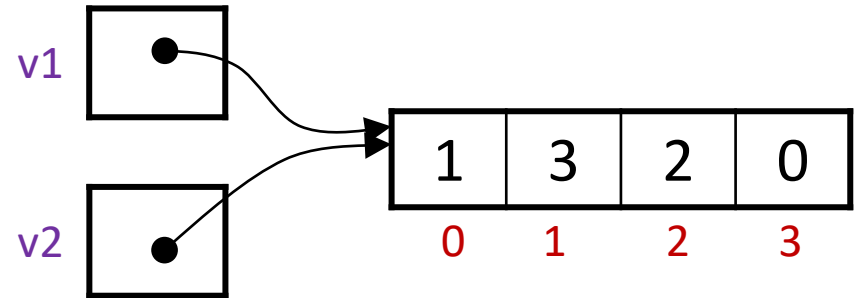
- Les components d'un array es poden veure com a variables del tipus dels elements de l'array.

`nomVble[índex] = expressió;`

`expressió` ha de ser del `tipus` dels components de l'array `nomVble`.

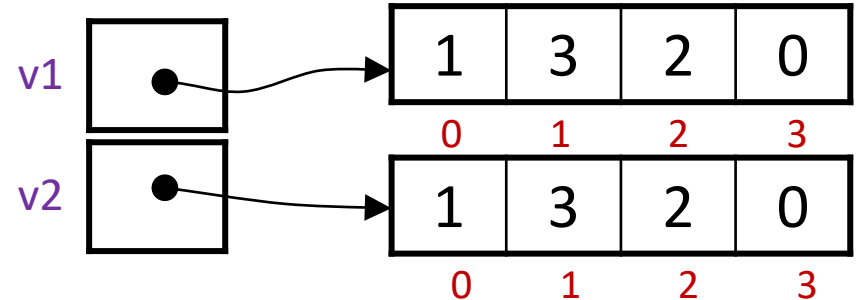
- L'assignació entre arrays tan sols afecta a les referències.

```
int[] v1 = {1, 3, 2, 0};  
int[] v2;  
v2 = v1;  
boolean igual = (v1 == v2);  
// igual val true
```



- Si es desitja una còpia d'un array s'han de crear nous components i realitzar l'assignació de cadascun dels valors.

```
v2 = new int[4];  
v2[0] = v1[0]; v2[1] = v1[1];  
v2[2] = v1[2]; v2[3] = v1[3];  
igual = (v1 == v2); // igual val false
```



Arrays unidimensionals – Mètodes

- Els arrays es defineixen com qualsevol paràmetre formal, indicant el tipus i el nom de la variable:

```
public static int mètode1(int[] v1, int[] v2) { ... }  
public static void main(String[] args) { ... }
```

- En la crida, només s'utilitza el nom de la variable :

```
int[] a1 = new int[10], a2 = new int[5];  
...  
int i = mètode1(a1, a2);
```

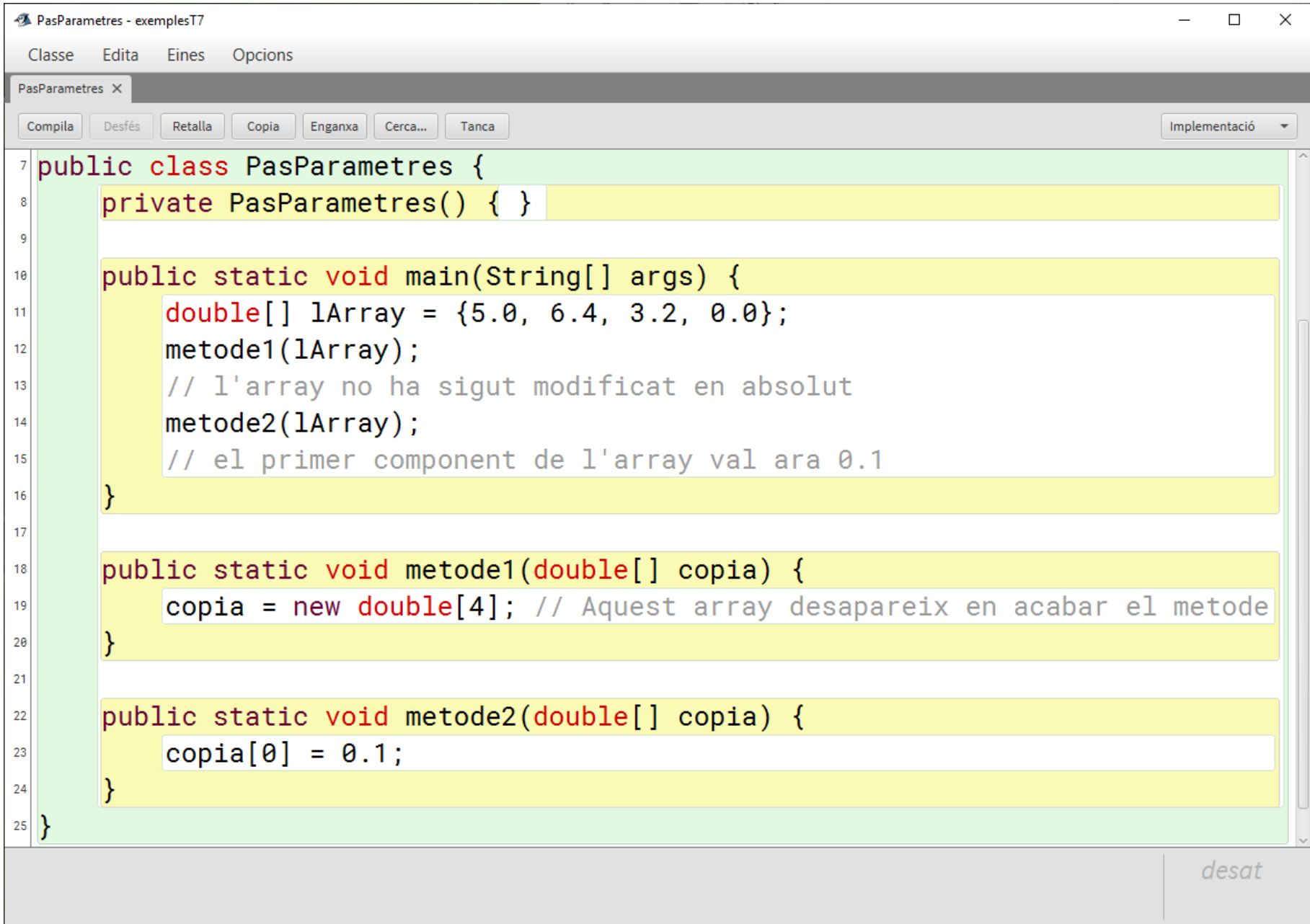
- Quan s'invoca a un mètode amb algun argument de tipus referència només es còpia la referència del paràmetre real en el formal.
- Els mètodes poden retornar com a resultat un array (referència als components). Per exemple:

```
public static char[] mètode2(int[] v1) {  
    char[] nou = new char[v1.length + 10];  
    ...  
    return nou;  
}
```

Invocació:

```
char[] res = mètode2(a1);
```

Exemple de pas d'arrays com paràmetres



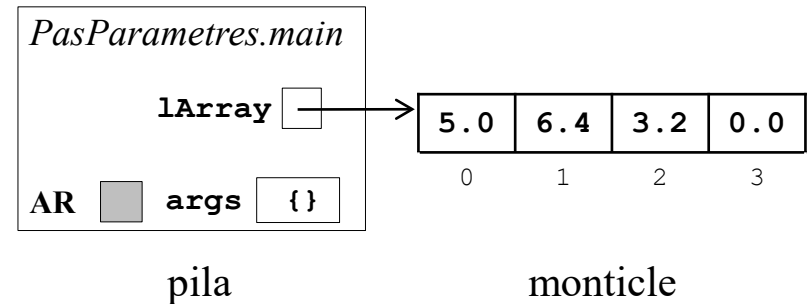
The screenshot shows a Java IDE window titled "PasParametres - exemplesT7". The menu bar includes "Classe", "Edita", "Eines", and "Opcions". The toolbar contains buttons for "Compila", "Desfés", "Retalla", "Copia", "Enganxa", "Cerca...", "Tanca", and a dropdown for "Implementació". The code editor displays the following Java code:

```
7 public class PasParametres {  
8     private PasParametres() { }  
9  
10    public static void main(String[] args) {  
11        double[] lArray = {5.0, 6.4, 3.2, 0.0};  
12        metode1(lArray);  
13        // l'array no ha sigut modificat en absolut  
14        metode2(lArray);  
15        // el primer component de l'array val ara 0.1  
16    }  
17  
18    public static void metode1(double[] copia) {  
19        copia = new double[4]; // Aquest array desapareix en acabar el metode  
20    }  
21  
22    public static void metode2(double[] copia) {  
23        copia[0] = 0.1;  
24    }  
25 }
```

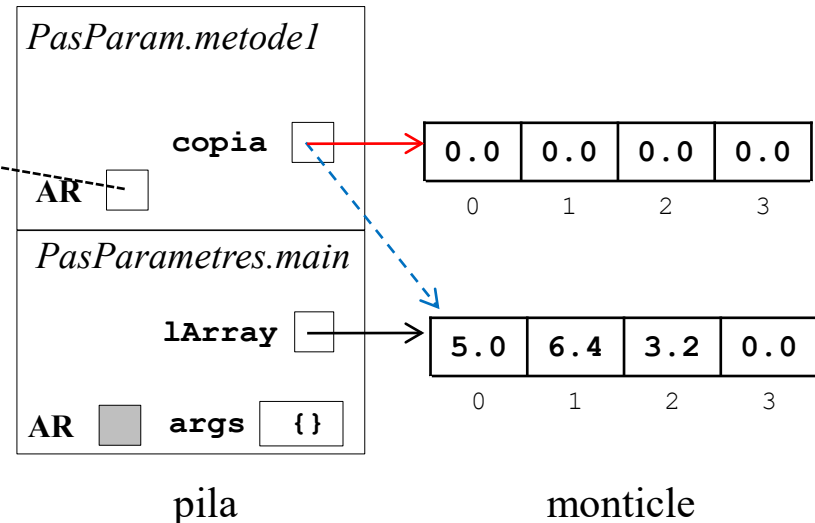
The status bar at the bottom right shows the text "desat".

Exemple de pas d'arrays com paràmetres

```
public static void main(String[] args) {  
    double lArray = {5.0, 6.4, 3.2, 0.0};  
    metode1(lArray);  
    // lArray no s'ha modificat  
    metode2(lArray);  
    // lArray[0] val 0.1  
}  
...
```

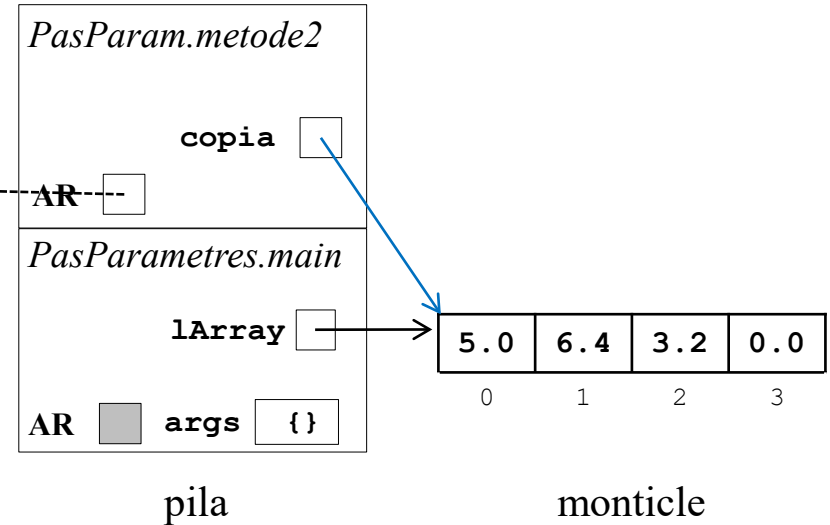


```
public static void main(String[] args) {  
    double lArray = {5.0, 6.4, 3.2, 0.0};  
    metode1(lArray);  
    // lArray no s'ha modificat  
    metode2(lArray);  
    // lArray[0] val 0.1  
}  
public static void metode1(double[] copia) {  
    copia = new double[4];  
}  
...
```

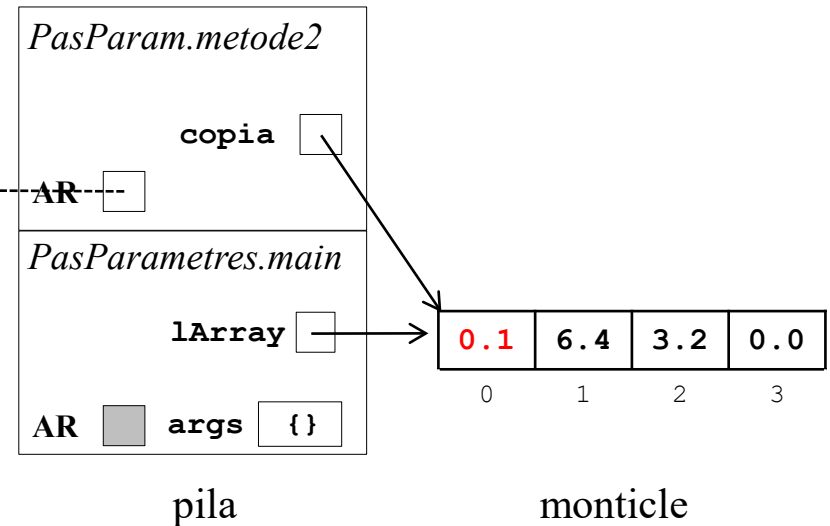


Exemple de pas d'arrays com paràmetres

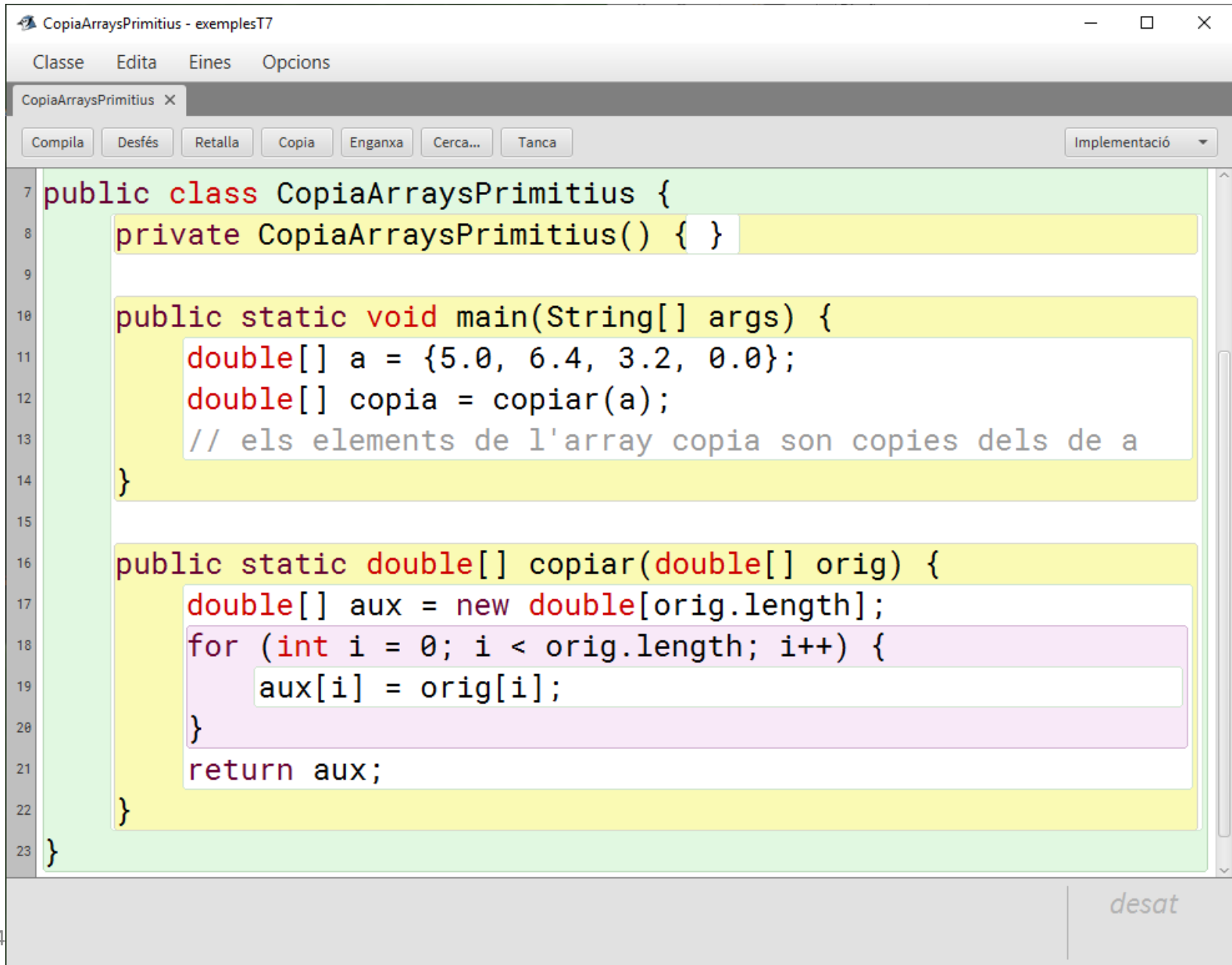
```
public static void main(String[] args) {  
    double lArray = {5.0, 6.4, 3.2, 0.0};  
    metode1(lArray);  
    //lArray no s'ha modificat  
    metode2(lArray);  
    //lArray[0] val 0.1  
}  
...  
public static void metode2(double[] còpia) {  
    còpia[0] = 0.1;  
}
```



```
public static void main(String[] args) {  
    double lArray = {5.0, 6.4, 3.2, 0.0};  
    mètode1(lArray);  
    // lArray no s'ha modificat  
    mètode2(lArray);  
    // lArray[0] val 0.1  
}  
...  
public static void metode2(double[] copia) {  
    copia[0] = 0.1;  
}
```



Exemple de còpia d'arrays de tipus primitius

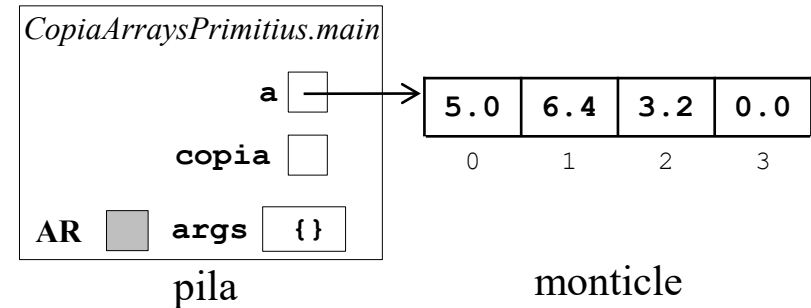


```
7 public class CopiaArraysPrimitius {
8     private CopiaArraysPrimitius() { }
9
10    public static void main(String[] args) {
11        double[] a = {5.0, 6.4, 3.2, 0.0};
12        double[] copia = copiar(a);
13        // els elements de l'array copia son copies dels de a
14    }
15
16    public static double[] copiar(double[] orig) {
17        double[] aux = new double[orig.length];
18        for (int i = 0; i < orig.length; i++) {
19            aux[i] = orig[i];
20        }
21        return aux;
22    }
23 }
```

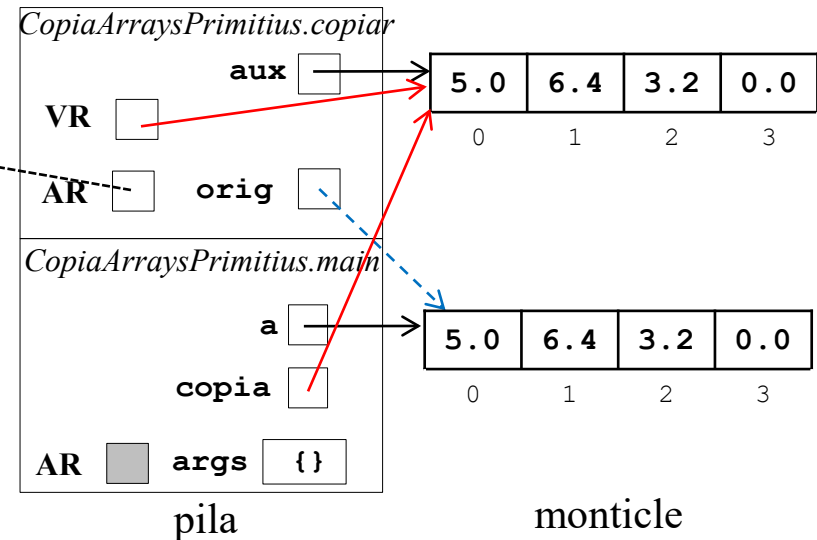
desat

Exemple de còpia d'arrays de tipus primitius

```
public static void main(String[] args) {  
    double[] a = {5.0, 6.4, 3.2, 0.0};  
    double[] copia = copiar(a);  
    // els elements de l'array copia  
    // son còpies dels de a  
}  
...
```



```
public static void main(String[] args) {  
    double[] a = {5.0, 6.4, 3.2, 0.0};  
    double[] copia = copiar(a);  
    // els elements de l'array copia  
    // son còpies dels de a  
}  
  
public static double[] copiar(double[] orig) {  
    double[] aux = new double[orig.length];  
    for (int i = 0; i < orig.length; i++)  
        aux[i] = orig[i];  
    return aux;  
}
```



Arrays d'objectes

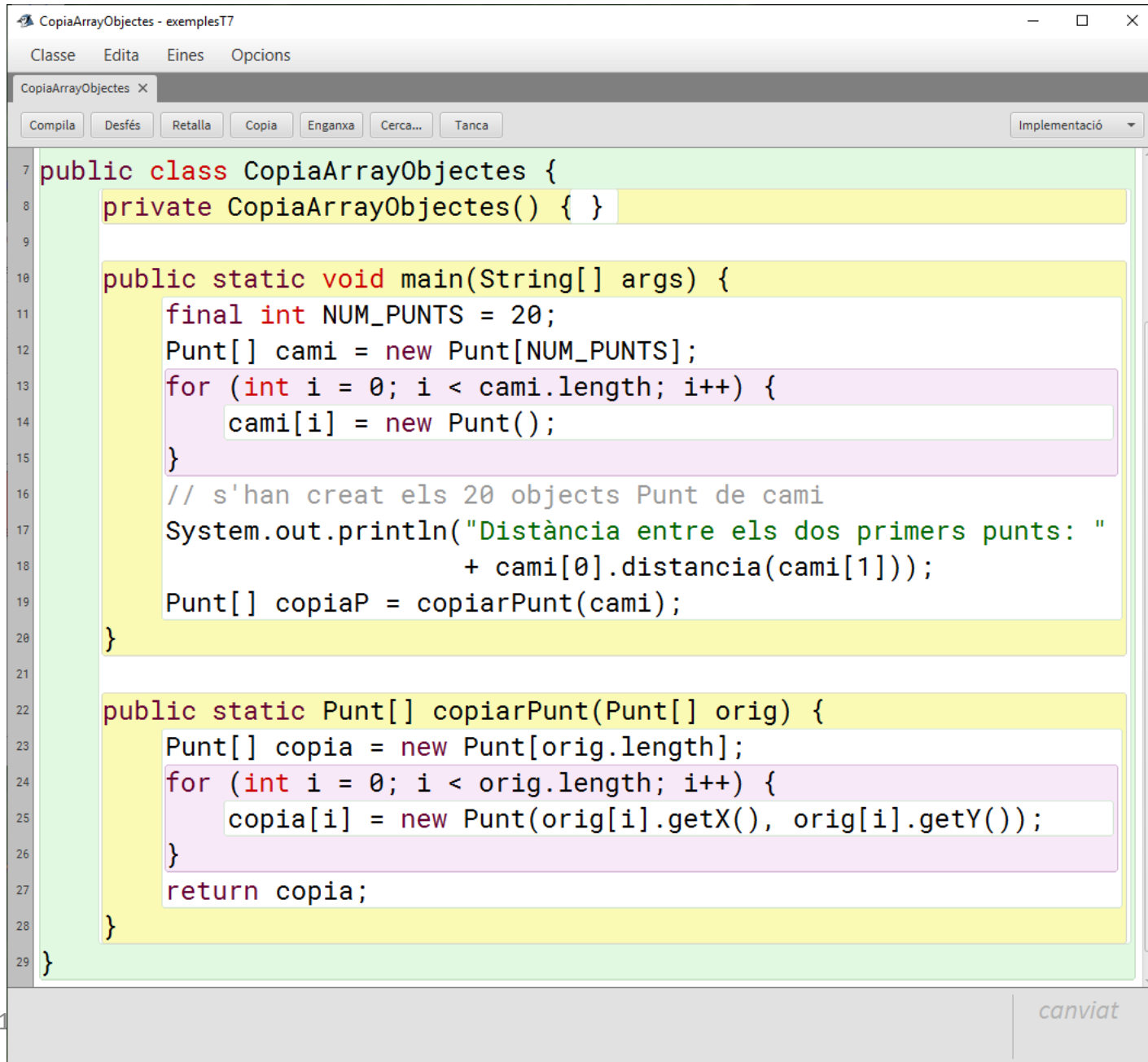
The screenshot shows the BlueJ IDE interface. The top menu bar includes 'Projecte', 'Edita', 'Eines', 'Vegeu', and 'Ajuda'. The left sidebar contains buttons for 'Nova classe...', a right arrow, and 'Compila'. The main workspace displays a class hierarchy with 'PasParametres', 'CopiaArraysPrimitius', 'CopiaArrayObjectes', 'Comptadors', 'Conjunt', 'GrupB', and 'Punt'. A red box on the left indicates the variable 'cami' is of type 'Punt[]'. The central code editor shows the following code:

```
Punt[] cami = new Punt[20];
cami
<object reference> (Punt[])
cami[0] = new Punt();
cami[1] = new Punt(3.5, 6);
```

Three inspection windows are open:

- cami : Punt[]**: Shows the array's state. The 'length' is 20. The first two elements, '[0]' and '[1]', are highlighted in yellow and contain object references (indicated by arrows). Elements '[2]' through '[19]' are 'null'. Buttons for 'Inspecciona', 'Obté', 'Mostra camps estàtics', and 'Tanca' are present.
- [0] : Punt**: Shows the state of the first element. It has two private double fields: 'x' (0.0) and 'y' (0.0). Buttons for 'Inspecciona', 'Obté', 'Mostra camps estàtics', and 'Tanca' are present.
- [1] : Punt**: Shows the state of the second element. It has two private double fields: 'x' (3.5) and 'y' (6.0). Buttons for 'Inspecciona', 'Obté', 'Mostra camps estàtics', and 'Tanca' are present.

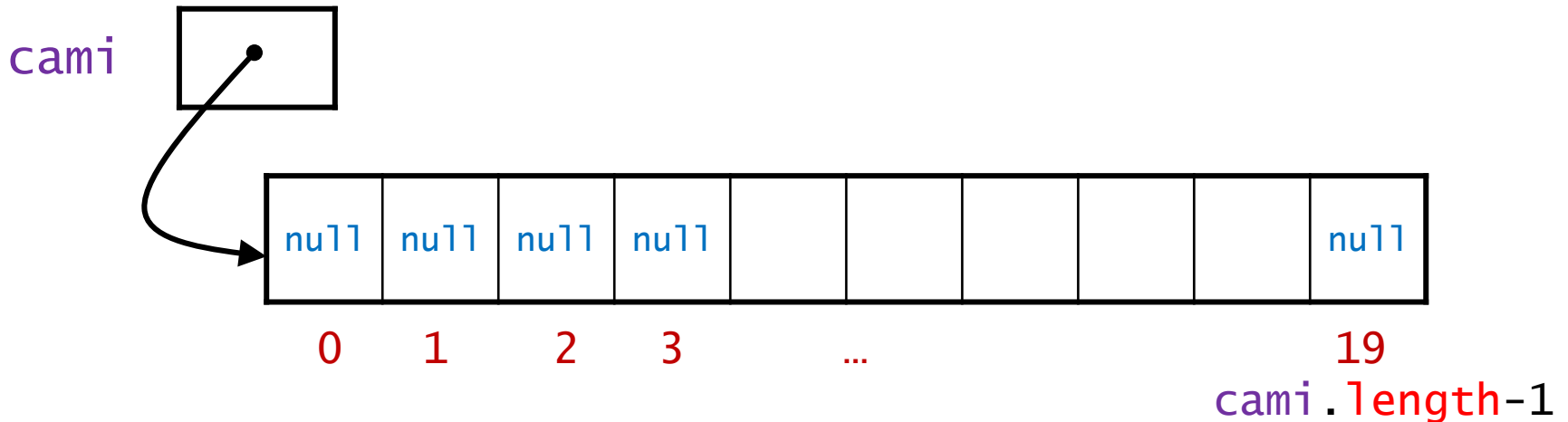
Exemple de còpia d'arrays d'objectes

The image shows a screenshot of an IDE window titled "CopiaArrayObjectes - exemplesT7". The window has a menu bar with "Classe", "Edita", "Eines", and "Opcions". Below the menu bar is a toolbar with buttons for "Compila", "Desfés", "Retalla", "Copia", "Enganxa", "Cerca...", and "Tanca". On the right side of the toolbar is a dropdown menu labeled "Implementació". The main area of the IDE displays Java code for a class named "CopiaArrayObjectes". The code is as follows:

```
7 public class CopiaArrayObjectes {  
8     private CopiaArrayObjectes() { }  
9  
10    public static void main(String[] args) {  
11        final int NUM_PUNTS = 20;  
12        Punt[] cami = new Punt[NUM_PUNTS];  
13        for (int i = 0; i < cami.length; i++) {  
14            cami[i] = new Punt();  
15        }  
16        // s'han creat els 20 objectes Punt de cami  
17        System.out.println("Distància entre els dos primers punts: "  
18                            + cami[0].distancia(cami[1]));  
19        Punt[] copiaP = copiarPunt(cami);  
20    }  
21  
22    public static Punt[] copiarPunt(Punt[] orig) {  
23        Punt[] copia = new Punt[orig.length];  
24        for (int i = 0; i < orig.length; i++) {  
25            copia[i] = new Punt(orig[i].getX(), orig[i].getY());  
26        }  
27        return copia;  
28    }  
29 }
```

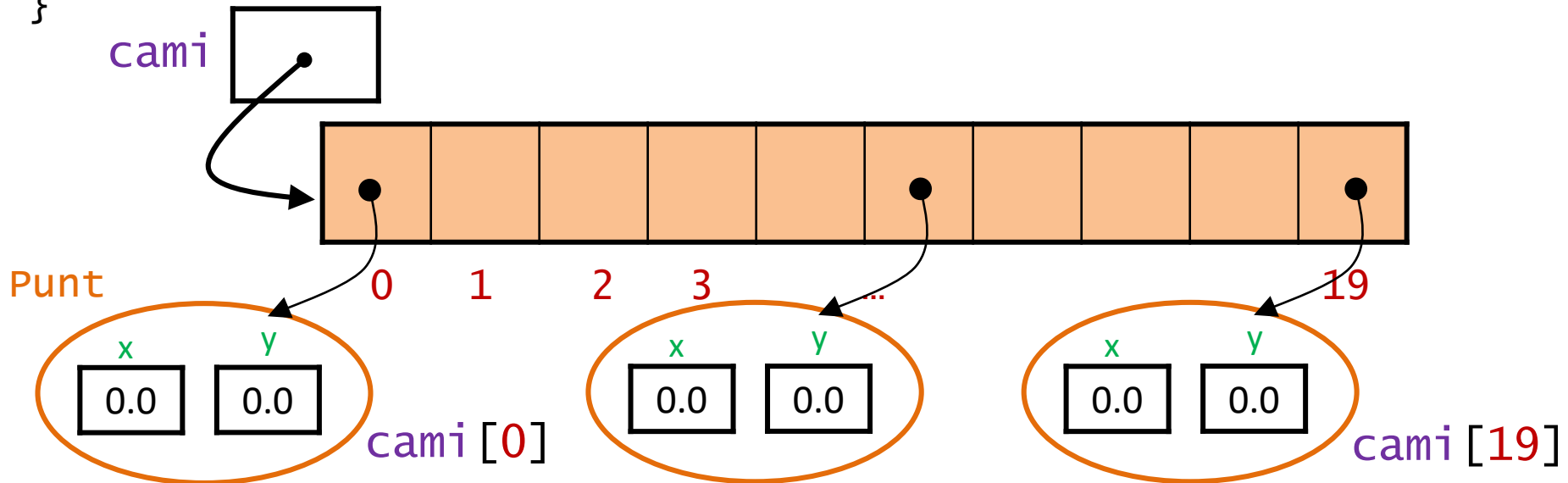
Exemple de còpia d'arrays d'objectes

```
public static void main(String[] args) {  
    final int NUM_PUNTS = 20;  
    Punt[] cami = new Punt[NUM_PUNTS];  
    for (int i = 0; i < cami.length; i++) {  
        cami[i] = new Punt();  
    }  
    // s'han creat els 20 objects Punt de cami  
    ...  
    System.out.println("Distància entre els dos primers  
                        punts: " + cami[0].distancia(cami[1]));  
    Punt[] copiaP = copiarPunt(cami);  
}
```



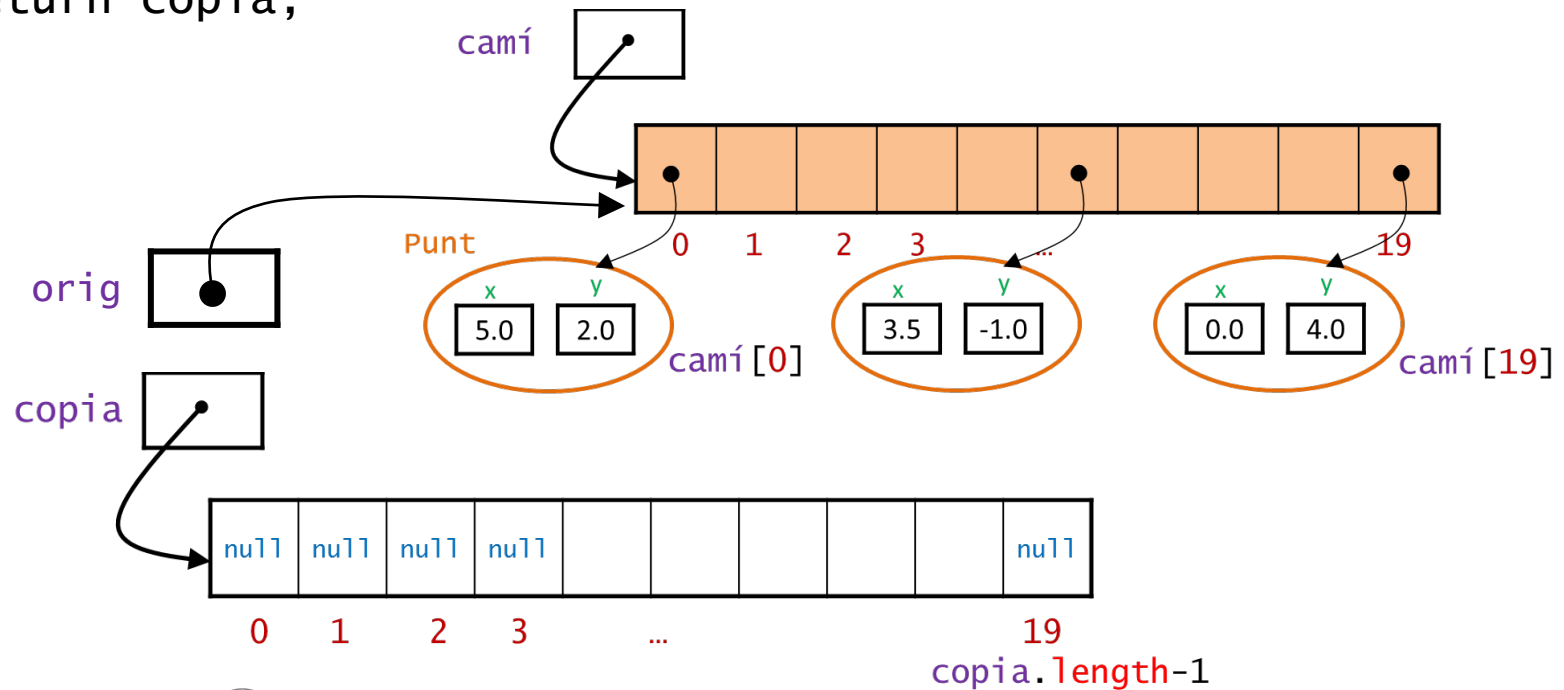
Exemple de còpia d'arrays d'objectes

```
public static void main(String[] args) {  
    final int NUM_PUNTS = 20;  
    Punt[] cami = new Punt[NUM_PUNTS];  
    for (int i = 0; i < cami.length; i++) {  
        cami[i] = new Punt();  
    }  
    // s'han creat els 20 objects Punt de cami  
    ...  
    System.out.println("Distància entre els dos primers  
                        punts: " + cami[0].distancia(cami[1]));  
    Punt[] copiaP = copiarPunt(cami);  
}
```



Exemple de còpia d'arrays d'objectes

```
public static void main(String[] args) {  
    ...  
    Punt[] copiaP = copiarPunt(camí);  
}  
public static Punt[] copiarPunt(Punt[] orig) {  
    Punt[] copia = new Punt[orig.length];  
    for (int i = 0; i < orig.length; i++) {  
        copia[i] = new Punt(orig[i].getX(), orig[i].getY());  
    }  
    return copia;  
}
```



Exemple de còpia d'arrays d'objectes

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

```
    ...
```

```
    Punt[] copiaP = copiarPunt(camí);
```

```
}
```

```
public static Punt[] copiarPunt(Punt[] orig) {
```

```
    Punt[] copia = new Punt[orig.length];
```

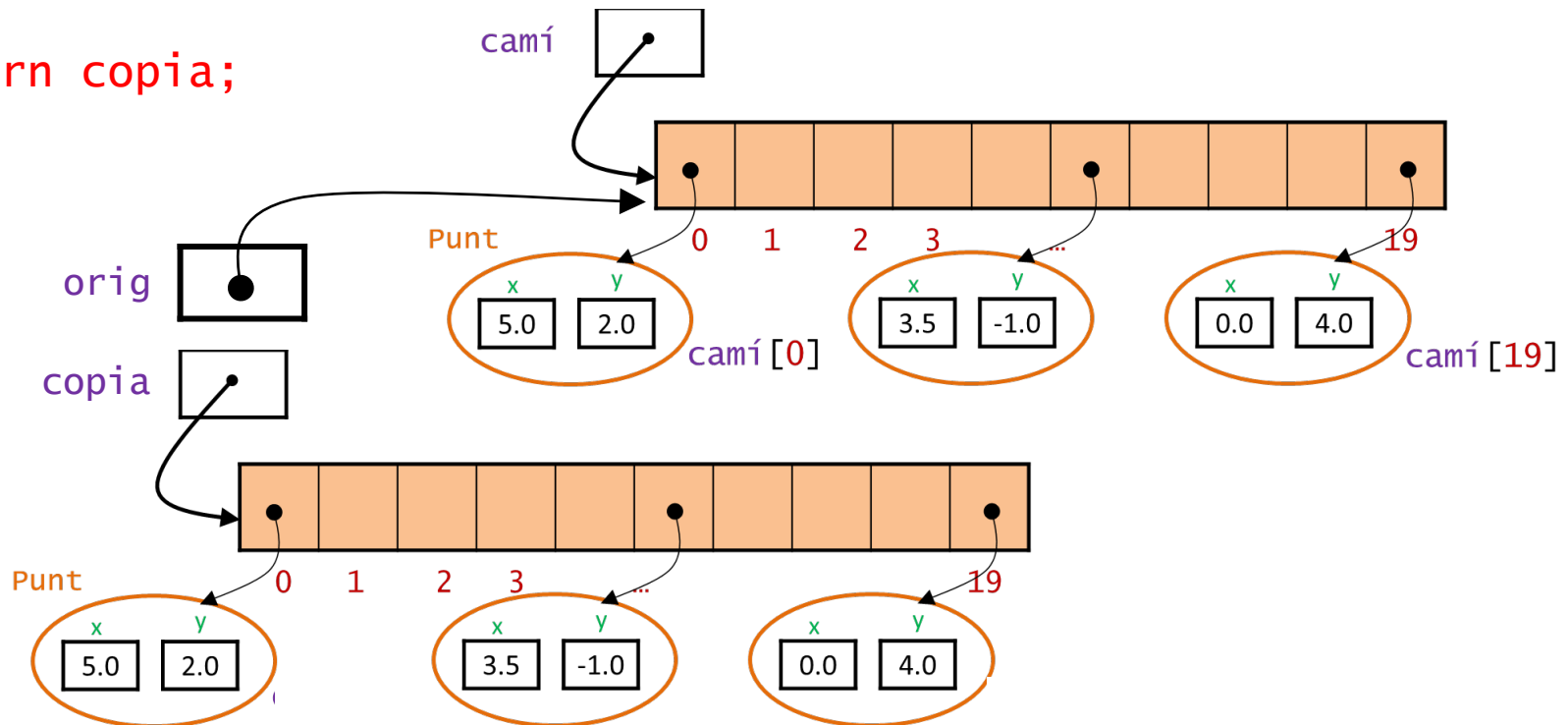
```
    for (int i = 0; i < orig.length; i++) {
```

```
        copia[i] = new Punt(orig[i].getX(), orig[i].getY());
```

```
    }
```

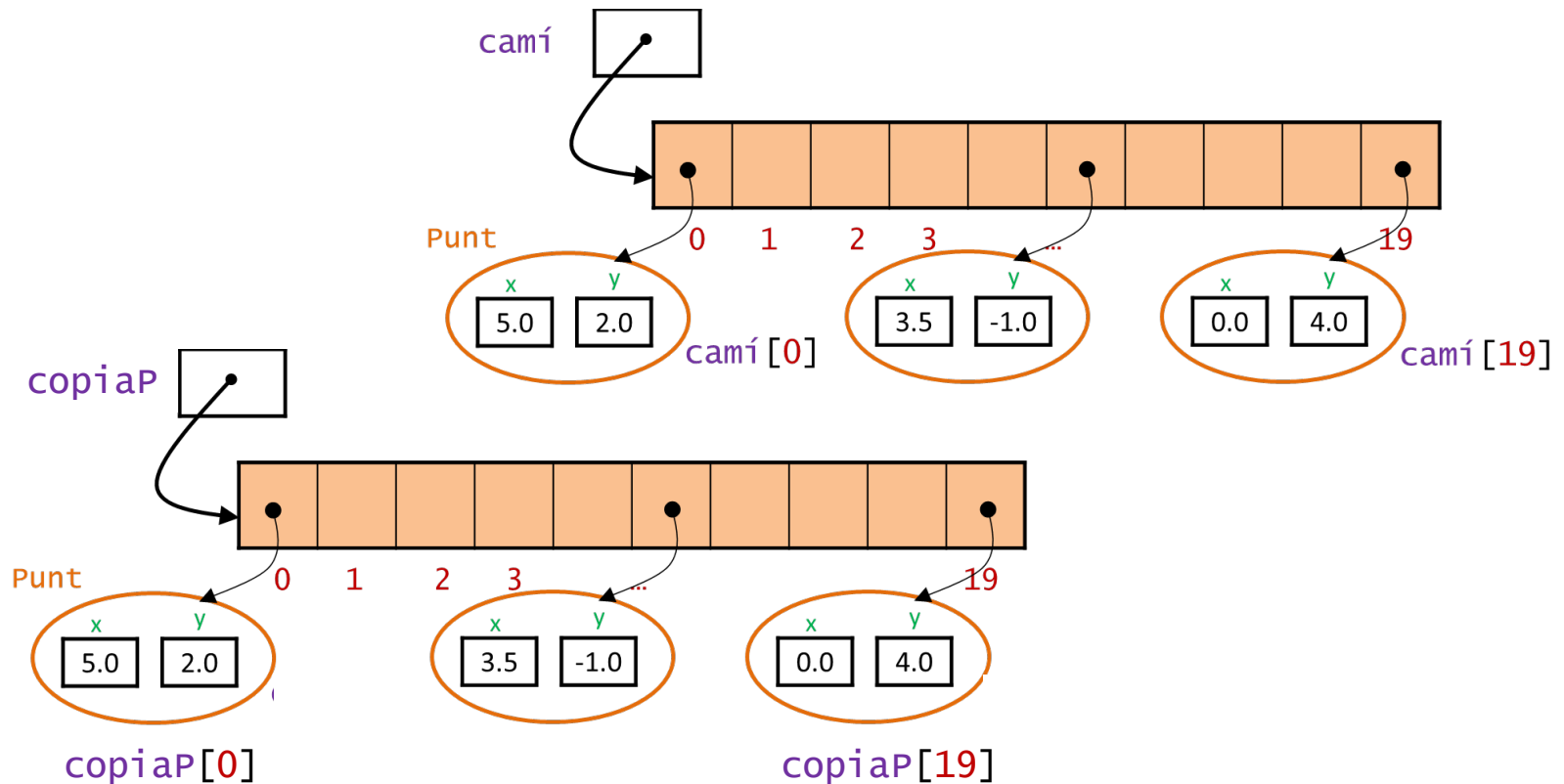
```
    return copia;
```

```
}
```



Exemple de còpia d'arrays d'objectes

```
public static void main(String[] args) {  
    ...  
    Punt[] copiaP = copiarPunt(camí);  
}
```



Exercicis amb arrays

Exercicis de CAP



- La classe **Re1oj**: clau CCDJG4ai
- La classe **TestRe1oj**: clau CCDJH4ai

Solució visible des del 27/11

Representació de dades amb arrays

- Amb un array es pot representar una col·lecció de dades del mateix tipus. Per exemple, suposem que volem comptar la freqüència d'aparició de cadascuna de les cares d'un dau de 10 cares (numerades del 0 al 9).

Comptadors - exemplesT7

- Un conjunt de NUM_COMPT comptadors:

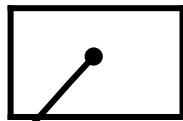
```
int[] compt = new int[NUM_COMPT];
```

Índex i , $0 \leq i < \text{compt.length}$	Component $\text{compt}[i]$
Valor o codi a comptar	Freqüència d'aparició d' i

- Representació de la freqüència d'aparició de cadascuna de les cares d'un dau
- Índex i : valor del dau
- Cada component $\text{compt}[i]$: freqüència d'aparició del valor i



compt



Actualitzar comptador: $\text{compt}[i]++$;

1	0	0	1	3	0	0	0	0	1
---	---	---	---	---	---	---	---	---	---

0 1 2 3 4 5 6 7 8 9

El 0: 1 vegada

El 3: 1 vegada

El 4: 3 vegades

El 9: 1 vegada

Representació de dades amb arrays

- Amb un array es pot representar una col·lecció de dades del mateix tipus. Per exemple:

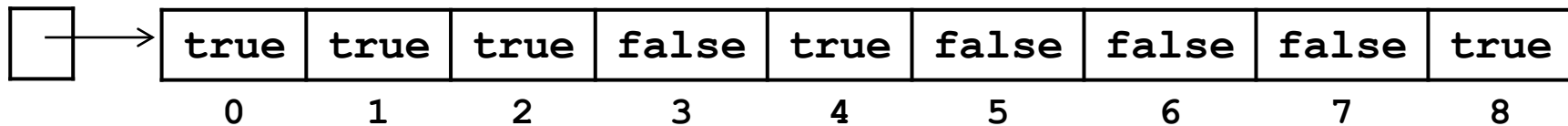
- Un conjunt de nombres naturals de qualsevol talla a l'interval $[0, N]$:

```
boolean[] conjunt = new boolean[N + 1];
```

Índex i , $0 \leq i < \text{conjunt.length}$	Component $\text{conjunt}[i]$
Valor natural	Pertany (true) o no (false) al conjunt

- Índex i : número natural
- Cada component $\text{conjunt}[i]$: true si el valor i pertany al conjunt i false en cas contrari.

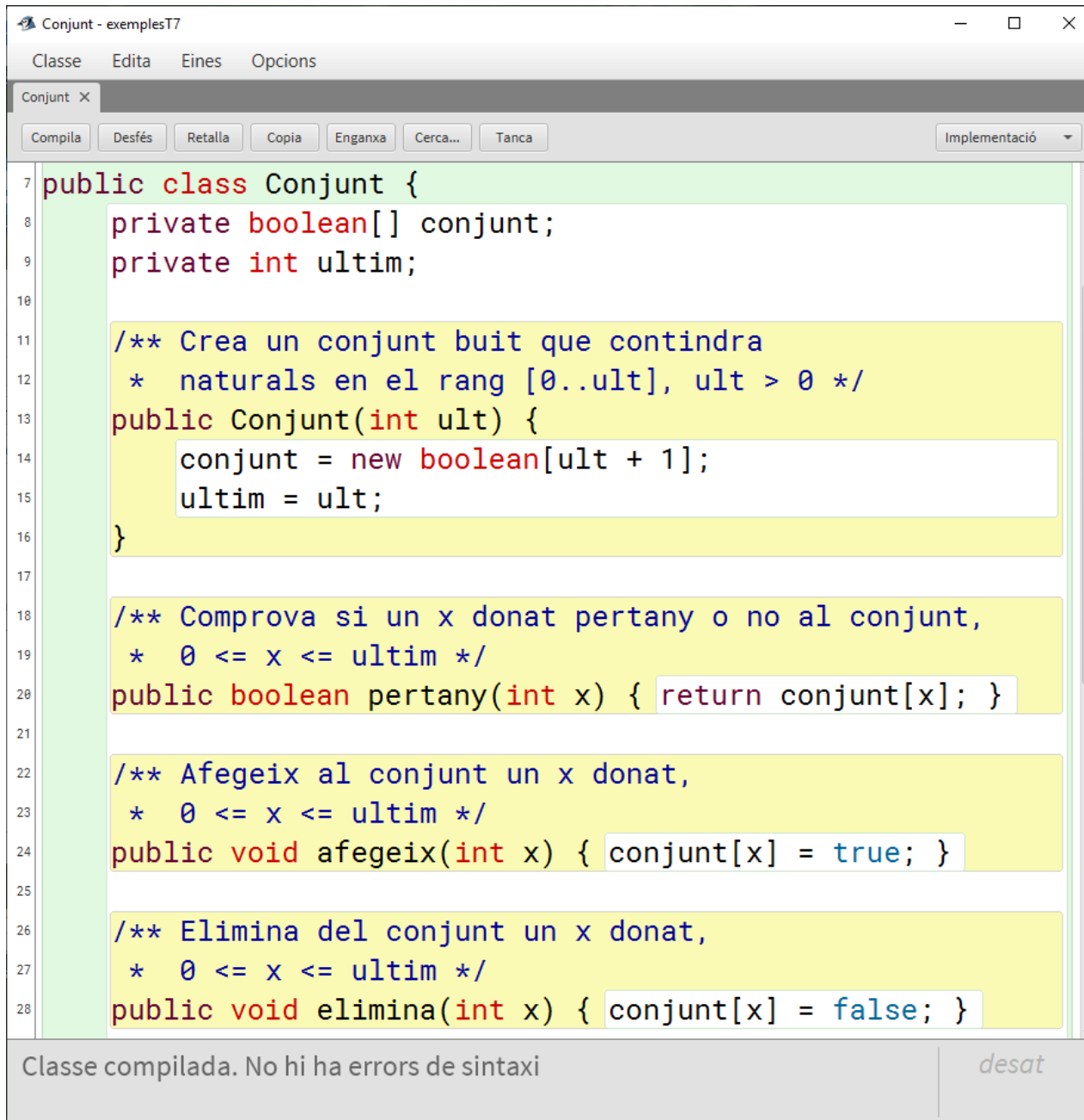
conjunt



Afegir element al conjunt: $\text{conjunt}[i] = \text{true};$
Eliminar element del conjunt: $\text{conjunt}[i] = \text{false};$

- Aquesta representació permet conèixer de forma molt ràpida (temps independent del nombre d'elements) si un natural pertany o no al conjunt.

Representació de dades amb arrays



The screenshot shows a Java IDE window titled "Conjunt - exemplesT7". The menu bar includes "Classe", "Edita", "Eines", and "Opcions". The toolbar contains buttons for "Compila", "Desfés", "Retalla", "Copia", "Enganxa", "Cerca...", "Tanca", and a dropdown for "Implementació". The code editor displays the following Java code for a class named "Conjunt":

```
7 public class Conjunt {
8     private boolean[] conjunt;
9     private int ultim;
10
11     /** Crea un conjunt buit que contindra
12      * naturals en el rang [0..ult], ult > 0 */
13     public Conjunt(int ult) {
14         conjunt = new boolean[ult + 1];
15         ultim = ult;
16     }
17
18     /** Comprova si un x donat pertany o no al conjunt,
19      * 0 <= x <= ultim */
20     public boolean pertany(int x) { return conjunt[x]; }
21
22     /** Afegeix al conjunt un x donat,
23      * 0 <= x <= ultim */
24     public void afegeix(int x) { conjunt[x] = true; }
25
26     /** Elimina del conjunt un x donat,
27      * 0 <= x <= ultim */
28     public void elimina(int x) { conjunt[x] = false; }
```

The status bar at the bottom indicates "Classe compilada. No hi ha errors de sintaxi" and "desat".

Representació de dades amb arrays

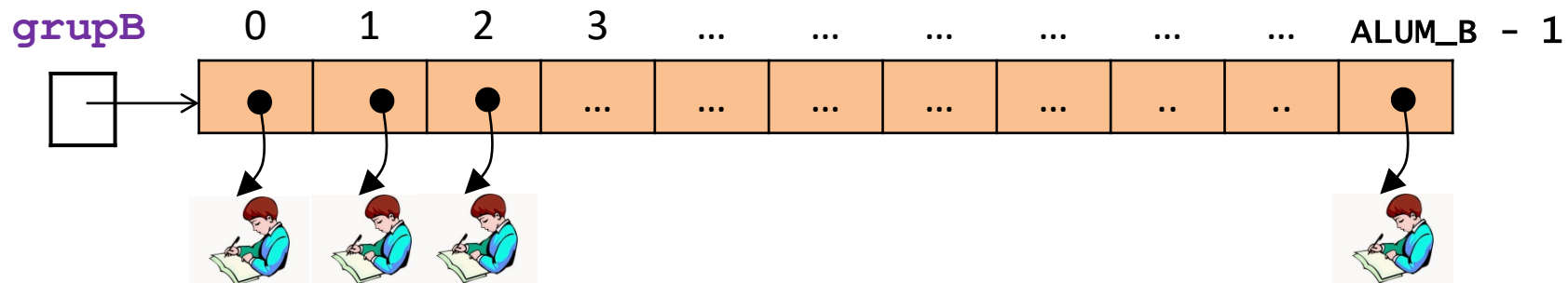
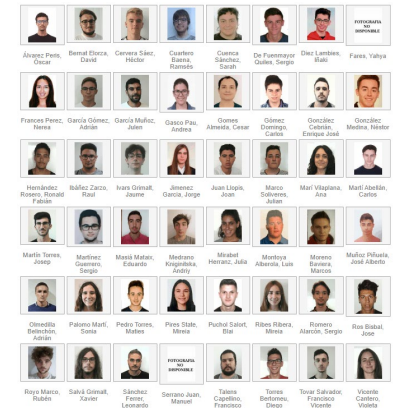
- Amb un array es pot representar una col·lecció de dades del mateix tipus. Per exemple:

- Els alumnes d'un grup de grandària ALUM_B:

```
Alumne[] grupB = new Alumne[ALUM_B];
```

Índex i, $0 \leq i < \text{grupB.length}$	Component grupB[i]
Ordre d'inserció	Informació relativa a un alumne

- Índex i: ordre en la llista de l'alumne, pot ser irrellevant.
- Cada component grupB[i]: dades de l'alumne i-ésim.



```
public class Alumne {
    private long dni;
    private double nota;
    private String nom;
    private boolean asistencia;
    ...
}
```

Afegir informació de l'alumne al grup:

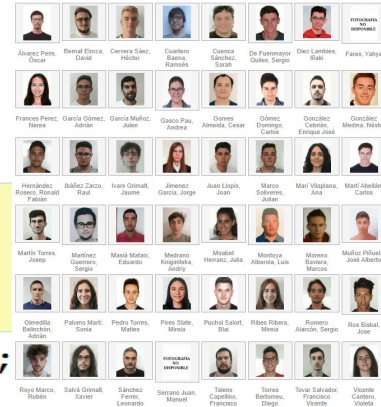
```
grupB[i] = new Alumne();
```

Representació de dades amb arrays

Alumne - exemplesT7

```
8 public class Alumne {
9     private long dni;
10    private double nota;
11    private String nom;
12    private boolean assistencia;
13    private char grup;
14
15    /** Crea un objecte Alumne, el nom i dni del
16     * qual s'introdueixen per teclat. */
17    public Alumne(Scanner tec) {
18        System.out.println("Introdueix les dades d'un alumne");
19        System.out.print("Nom: "); nom = tec.nextLine();
20        System.out.print("Dni: "); dni = tec.nextLong();
21        assistencia = true;
22        nota = 0;
23        grup = ' ';
24    }
```

```
8 public class GrupB {
9     public static final int ALUM_B = 56;
10    private Alumne[] grupB;
11
12    /** Crea un GrupB amb els alumnes del grup B,
13     * les dades dels quals es lligen des de teclat.
14     */
15    public GrupB(Scanner tec) {
16        grupB = new Alumne[ALUM_B];
17        for (int i = 0; i < grupB.length; i++) {
18            grupB[i] = new Alumne(tec);
19            grupB[i].setGrup('B');
20        }
21    }
```



Grup B - exemplesT7

Representació de dades amb arrays

- Amb un array es pot representar una col·lecció de dades del mateix tipus. P.e.:

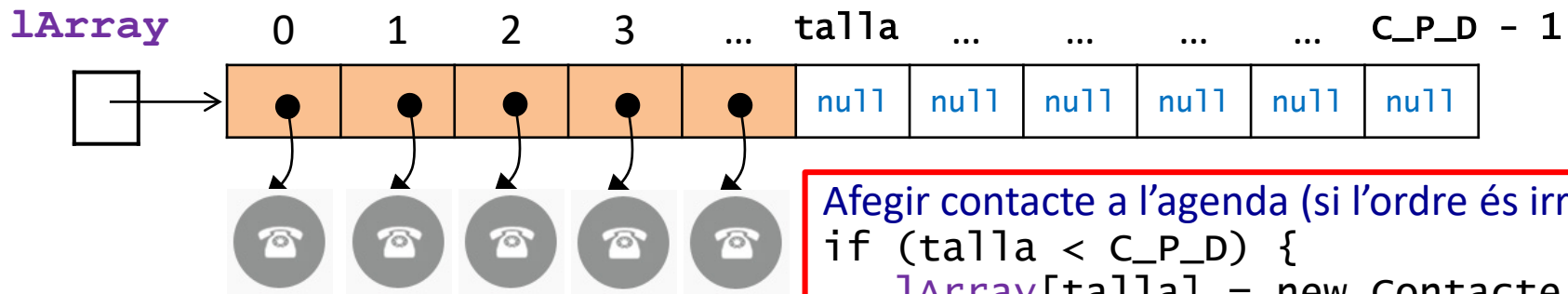
- Agenda de contactes de grandària variable (màxim C_P_D):

Contacte[] **lArray** = new Contacte[C_P_D];

Índex i, $0 \leq i < \text{agenda.length}$	Component agenda[i]
Ordre d'inserció	Informació relativa a un contacte



- Índex i: ordre en la llista del contacte, pot ser irrellevant.
- Cada component **lArray**[i]: dades del contacte i-ésim.



```
public class Contacte {  
    private String telefon;  
    private String nom;  
    ...  
}
```

Afegir contacte a l'agenda (si l'ordre és irrellevant):

```
if (talla < C_P_D) {  
    lArray[talla] = new Contacte(n, t);  
    talla++;  
}
```

Esborrar contacte en posició i de l'agenda:

```
if (i < talla) {  
    Desplaçar una posició cap a l'esquerra els  
    contactes de lArray en [i+1, talla-1]  
    talla--;  
}
```


Representació de dades amb arrays



Blue:exercicisT7 [agendaSenseOrdre]

```
35 public class Agenda {
36     private static final int C_P_D = 250;
37     private Contacte[] lArray;
38     private int talla;
39
40     /** Crea una Agenda buida, amb 0 contactes. */
41     public Agenda() {
42         lArray = new Contacte[C_P_D];
43         talla = 0;
44     }
45
46     /**
47      * Torna la talla de l'agenda, es a dir, del numero de contactes.
48      * @return int numero de contactes de l'agenda actual.
49      */
50     public int getTalla() { return talla; }
```

```
19 public class Contacte {
20     private String telefon, nom;
21
22     /**
23      * Crea un Contacte a partir d'un telefon i un nom.
24      * @param telefon String que representa un telefon.
25      * @param nom String que representa un nom.
26      */
27     public Contacte(String n, String t) {
28         nom = n; telefon = t;
29     }
```

Representació de dades amb arrays

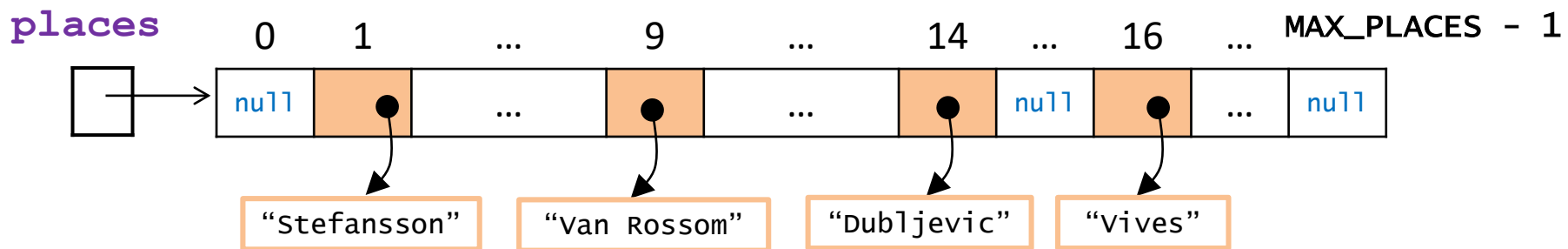
- Amb un array es pot representar una col·lecció de dades del mateix tipus. Per exemple:

- Els noms dels ocupants de les places d'un autobús de grandària MAX_PLACES:
`String[] places = new String[MAX_PLACES];`

Índex i , $0 \leq i < \text{places.length}$	Component $\text{places}[i]$
Número de seient	Nom del passatger



- Índex i : número de seient.
- Cada component $\text{places}[i]$: nom del passatger que ocupa el seient i .



Reservar seient: `places[i] = "nom";`

Cancel.lar reserva: `places[i] = null;`

Representació de dades amb arrays

Blue:exercicisT7 [autobus]



```
7 public class Autobus {
8     public static final int MAX_PLACES = 50;
9     private String[] places;
10
11     /** Crea un Autobus buit */
12     public Autobus() {
13         places = new String[MAX_PLACES];
14     }
15
16     /** Reserva el seient i al passatger nom
17      * Precondició: 0 <= i < MAX_PLACES i places[i] lliure
18      */
19     public void reservar(int i, String nom) { places[i] = nom; }
20
21     /** Cancel·la la reserva del seient i
22      * Precondició: 0 <= i < MAX_PLACES i places[i] ocupat
23      */
24     public void cancelar(int i) { places[i] = null; }
```

Exercici: Gestió d'un hostel rural

- Des de l'opció **Projecte** de **BlueJ**, obre **hostal-val.jar** amb l'opció **Open ZIP/JAR...**
- A la carpeta del projecte **hostal-val**, trobaràs l'enunciat del problema **Hostal-Val.pdf** i una subcarpeta **doc** amb la documentació (en el format habitual de Java) de les classes implicades. Obre el fitxer **allclasses-index.html** amb qualsevol navegador per consultar aquesta documentació.

Field Summary

Fields

Modifier and Type	Field and Description
static int	AD Regim d'allotjament i desdijuni.
static int	MP Regim de mitja pensio.
static int	PC Regim de pensio completa.

Constructor Summary

Constructors

Constructor and Description
Client (java.lang.String nf, java.lang.String n, Data a, Data e) Crea un Client donats un nif, un nom, una data d'arribada i una data d'eixida (sent l'arribada anterior a l'eixida) i regim un valor enter aleatori en [0..2], indicant si esta en regim d'al·lojament i desdijuni, mitja pensio o pensio completa, respectivament.

Method Summary

All Methods	Instance Methods	Concrete Methods
Modifier and Type	Method and Description	
Data	getArribada()	Torna la data d'arribada.
Data	getEixida()	Torna la data d'eixida.
java.lang.String	getNif()	Torna el NIF.
java.lang.String	getNom()	Torna el nom.
int	getRegim()	Torna el regim.
java.lang.String	toString()	Torna un String amb les dades d'un Client.

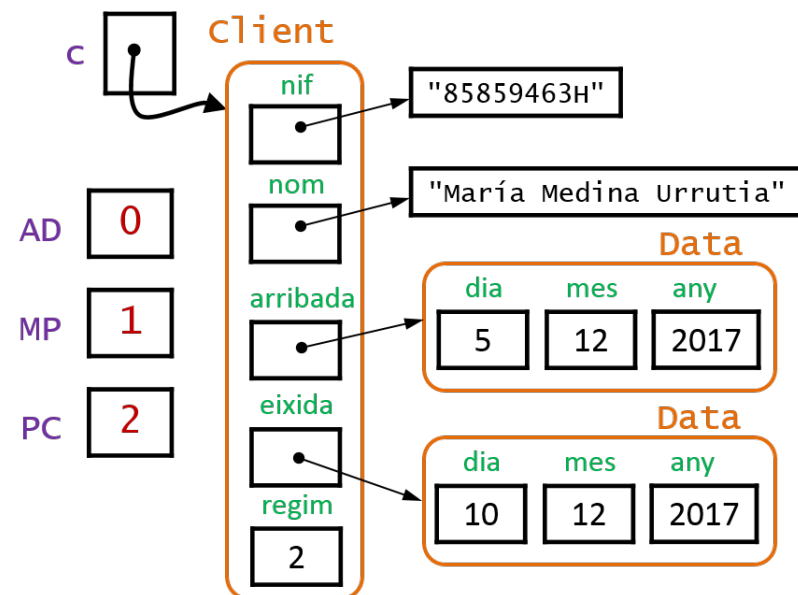
Constructor Summary

Constructors

Constructor and Description
Data () Crea una Data amb els valors de la data del sistema.
Data (int d, int m, int a) Crea una Data amb els valors donats de dia, mes i any.

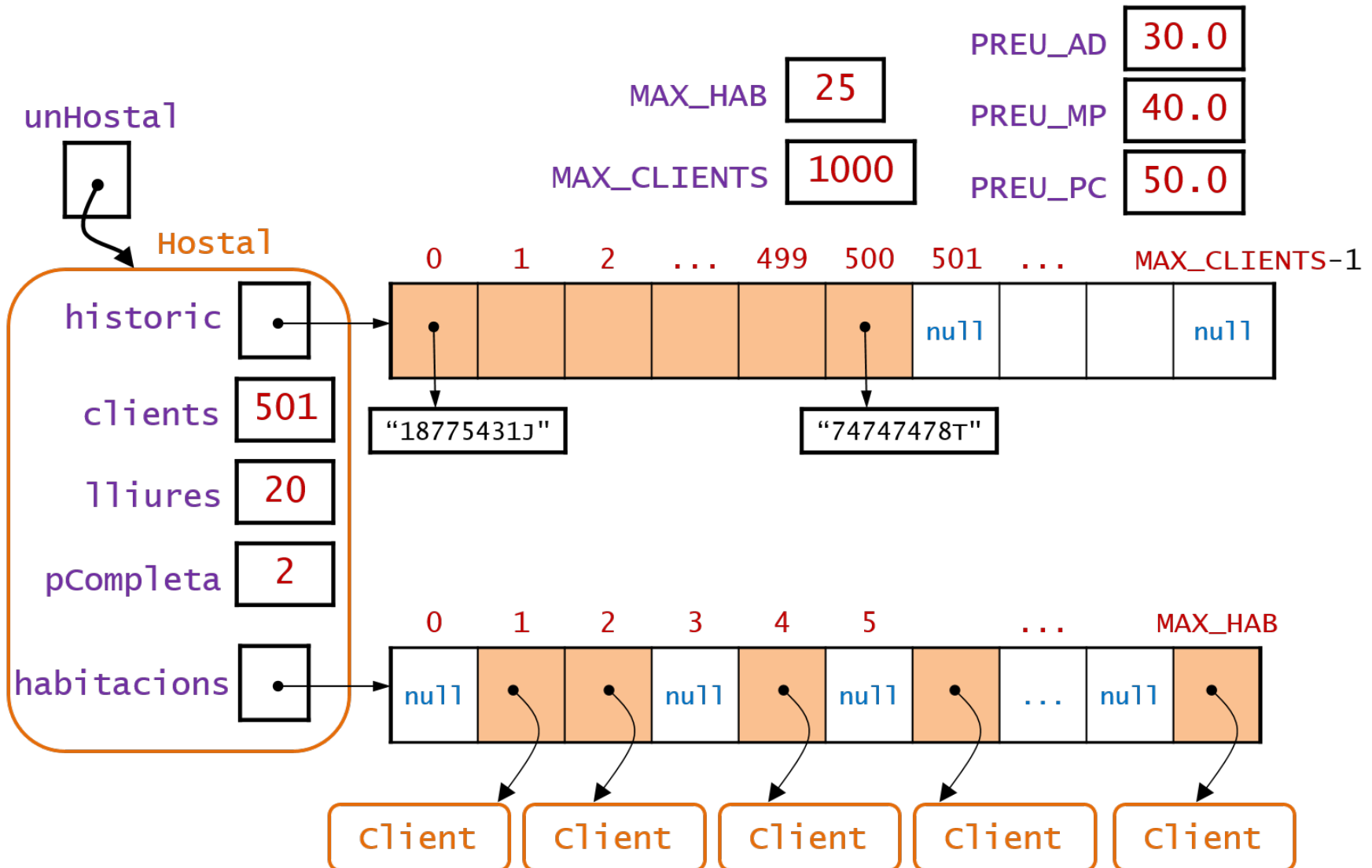
Method Summary

All Methods	Instance Methods	Concrete Methods
Modifier and Type	Method and Description	
int	difDies (Data d)	Torna el numero de dies entre la Data this i un altra Data d donada no inclosa, es a dir, el numero de dies en [this, d[.
boolean	equals (java.lang.Object o)	Comprova si la Data this es igual a un altra donada.
boolean	esAnterior (Data d)	Comprova si la Data this es anterior a un altra Data donada, suposant ambdues correctes.
boolean	esCorrecta ()	Comprova si la Data this és correcta.
java.lang.String	toString ()	Torna un String amb la informacio de la Data this en el format dd/mm/aaaa.



Exercici:

Gestió d'un hostal rural



Exercici:

Gestió d'un hostal rural



Field Summary	
Fields	
Modifier and Type	Field and Description
static int	MAX_CLIENTS Numero maxim de clients.
static int	MAX_HAB Numero maxim d'habitacions.
static double	PREU_AD Preu/nit d'una habitacio en regim d'allotjament i desdijuni.
static double	PREU_MP Preu/nit d'una habitacio en regim de mitja pensio.
static double	PREU_PC Preu/nit d'una habitacio en regim de pensio completa.

Constructor Summary	
Constructors	
Constructor and Description	
Hostal()	Crea un Hostal amb totes les habitacions lliures, es a dir, no hi ha clients i, per tant, no hi ha clients en regim de pensio completa ni historic de NIF.

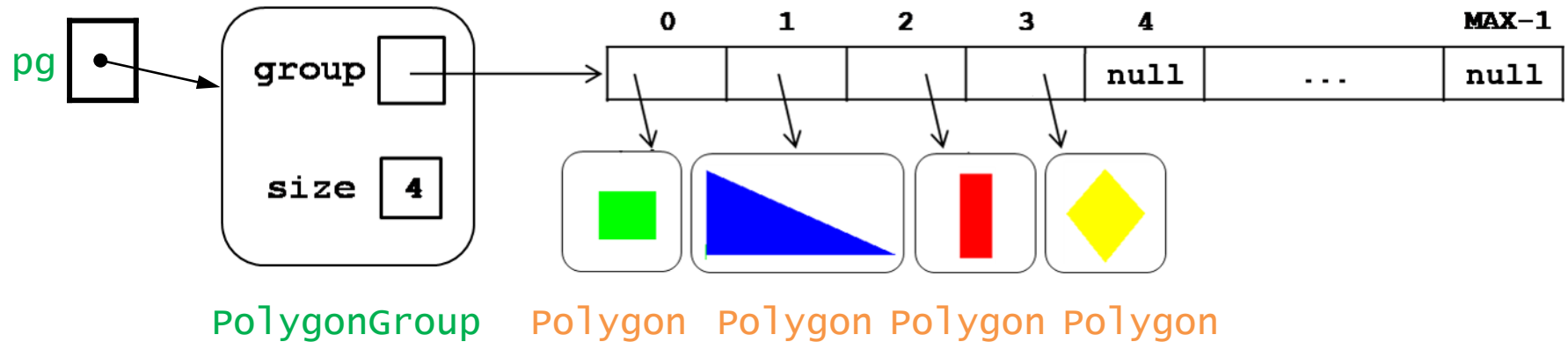
Method Summary

All Methods	Instance Methods	Concrete Methods
Modifier and Type	Method and Description	
boolean	checkIn (java.lang.String nif, java.lang.String nom, Data arribada, Data eixida) Check in d'un client de nif nif, nom nom, data d'arribada arribada i data d'eixida eixida, tornant true si s'ha pogut fer i false en cas contrari (si no hi ha habitacions lliures).	
double	checkOut (Data d) Check out de tots els clients tals que la seua data d'eixida es la Data d donada, tornant el preu total a pagar o 0 si no hi ha cap client amb aquesta data d'eixida.	
double	checkOut (int i) Check out del client que ocupa l'habitacio i (sent i un numero d'habitacio valid), tornant el preu a pagar o 0 si l'habitacio no estava ocupada.	
Client	getClient (int i) Torna el Client que ocupa l'habitacio i (sent i un numero d'habitacio valid) o null si l'habitacio esta lliure.	
int	getClientsHistoric () Torna el numero de clients en l'historic.	
int	getLliures () Torna el numero d'habitacions lliures.	
int	getPC () Torna el numero d'habitacions ocupades per clients en regim de pensio completa.	
boolean	hiHaLliures () Torna true si hi ha habitacions lliures i torna false en cas contrari.	
int[]	pensioCompleta () Torna un array amb els numeros d'habitacions ocupades pels clients en regim de pensio completa.	
int	primeraLliure () Torna el numero de la primera habitacio lliure (la de numero menor) si hi ha habitacions lliures o torna un -1 si no hi ha.	
java.lang.String	toString () Torna un String que descriu l'Hostal, es a dir, quins clients ocupen quines habitacions i quines habitacions estan lliures.	

Tractament seqüencial i directe d'un array

- En general es consideren dos tipus de tractament dels elements d'una estructura lineal, els anomenats **tractament directe** i **seqüencial**:
 - **Directe**: S'accedeix als elements per la localització, sense cap patró d'accés específic.
 - Exemples d'accés per posició: Problemes que usen l'estructura com un conjunt de comptadors o com a referències posicionals.
 - Exemples més complexos: Problemes que aprofiten l'accés directe dels arrays i les propietats d'ordenació dels seus elements com la cerca binària o els algorismes d'ordenació, etc. (Es veuran més endavant i en PRG)
 - **Seqüencial**: S'accedeix als elements de l'estructura (o d'una part d'ella) posicionalment, un darrere l'altre.
 - Exemples: Problemes de recorregut i cerca seqüencial.
- Com els arrays són estructures **d'accés directe**, qualsevol dels dos tipus de tractament és possible.

Exemple: array de Polygon d'un PolygonGroup



accès directe

`public boolean add(Polygon pol)`. Afegeix al grup, damunt del tot, un polígon donat, tornant true. Si s'excedeix la capacitat del grup, el polígon no s'afegeix i el mètode torna false.

accès seqüencial

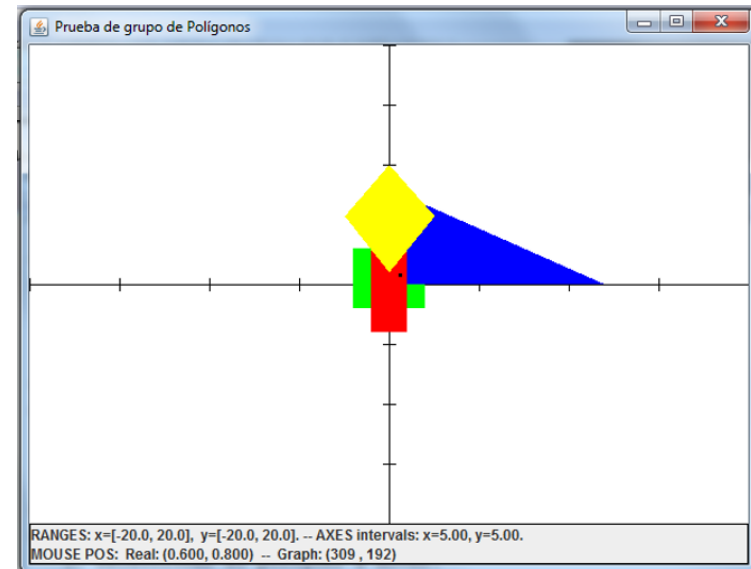
recorregut

`public Polygon[] toArray()`. Torna un array amb la seqüència de polígons del grup, per ordre des del de més avall al de més amunt.

accès seqüencial

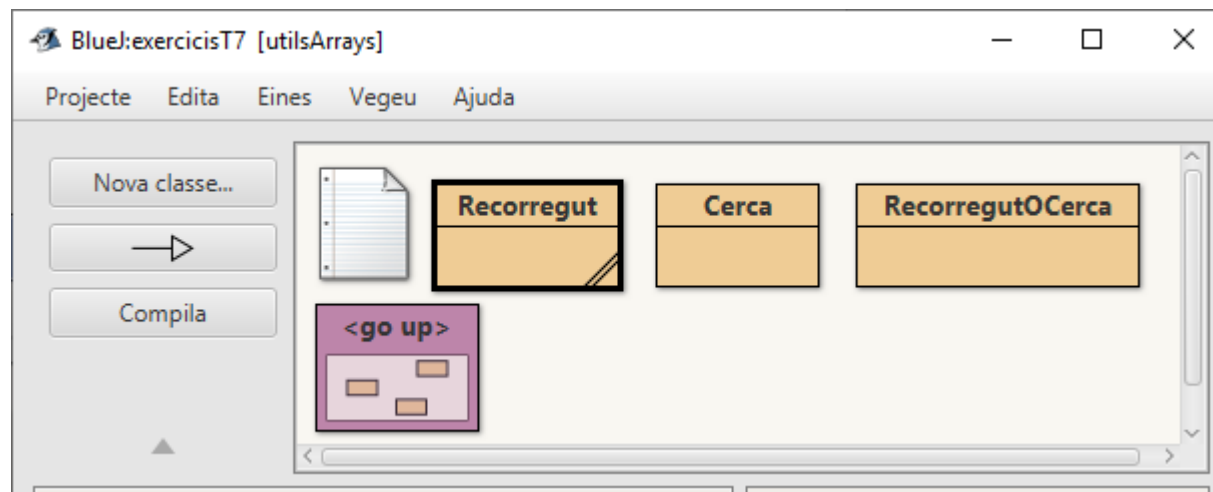
cerca

`private int search(Point p)`. Busca en el grup descendentment, de més amunt a més avall, el primer polígon que conté a un punt donat, tornant la seua posició en el grup. Si no existeix, torna -1.



Accés seqüencial: recorregut i cerca

- Un **recorregut** es caracteritza per haver de visitar **tots** els elements de l'array per poder trobar la solució del problema.
 - Per contra, una **cerca** persegueix trobar el primer element que compleix una característica donada.
- El recorregut d'arrays s'usa per resoldre problemes que necessiten processar totes les dades per a poder determinar la solució.
- Exemples: Obtindre el **màxim** o el **mínim** d'un conjunt de nombres, obtenir la **suma** o el **producte** de tots els números d'un conjunt donat, obtenir la **mitjana**, etc.
- Es duen a terme mitjançant variables enteres que s'usen com a índexs per accedir a les seues diferents posicions.
- S'ha de portar el control de quines posicions ja s'han visitat i quantes s'han de visitar per poder resoldre el problema.



Recorregut ascendent d'un array

des d'una posició inici fins a una posició fi

$$0 \leq \text{inici} \leq \text{fi} < \text{a.length}$$

- Recorregut iteratiu ascendent amb un bucle while:

```
int i = inici;
while (i <= fi) {
    tractar(a[i]); // operacions amb l'element i-èsim
    avançar(i);
}
```

- Recorregut iteratiu ascendent amb un bucle for:

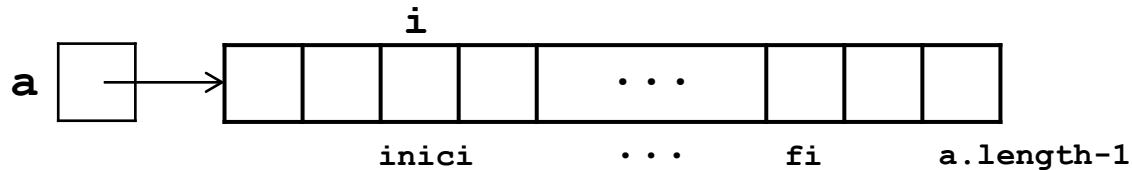
```
for (int i = inici; i <= fi; avançar(i)) {
    tractar(a[i]); // operacions amb l'element i-èsim
}
```

- El mètode avançar representa l'increment de l'índex.

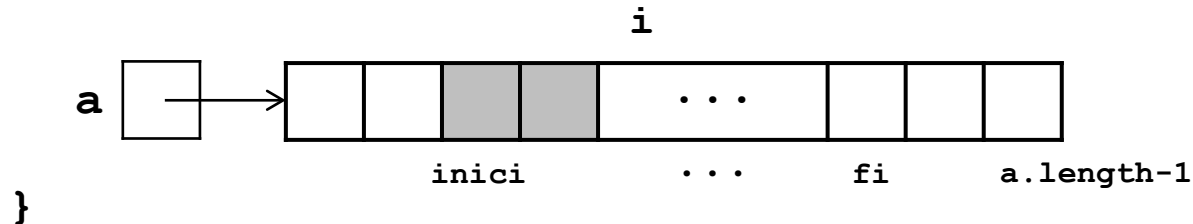
Recorregut ascendent d'un array

des d'una posició inici fins a una posició fi
 $0 \leq \text{inici} \leq \text{fi} < \text{a.length}$

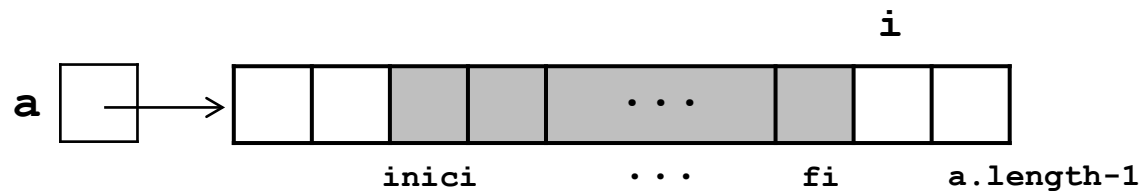
```
int i = inici;
```



```
while (i <= fi) {  
    tractar(a[i]);  
    avançar(i);
```



```
}
```



Recorregut descendent d'un array

des d'una posició *fi* fins a una posició *inici*

$$0 \leq \text{inici} \leq \text{fi} < \text{a.length}$$

- Recorregut iteratiu descendent amb un bucle `while`:

```
int i = fi;
while (i >= inici) {
    tractar(a[i]); // operacions amb l'element i-èsim
    retrocedir(i);
}
```

- Recorregut iteratiu descendent amb un bucle `for`:

```
for (int i = fi; i >= inici; retrocedir(i)) {
    tractar(a[i]); //Operacions amb l'element i-èsim
}
```

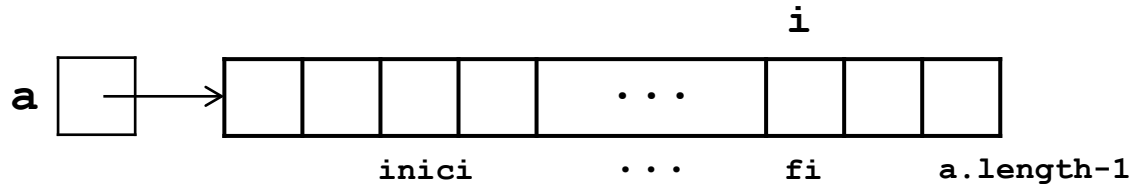
- El mètode `retrocedir` representa el decrement de l'índex.

Recorregut descendent d'un array

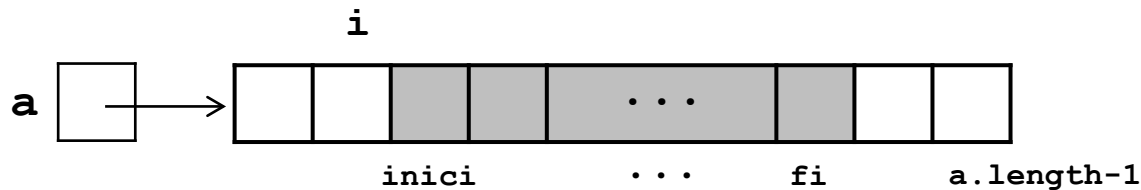
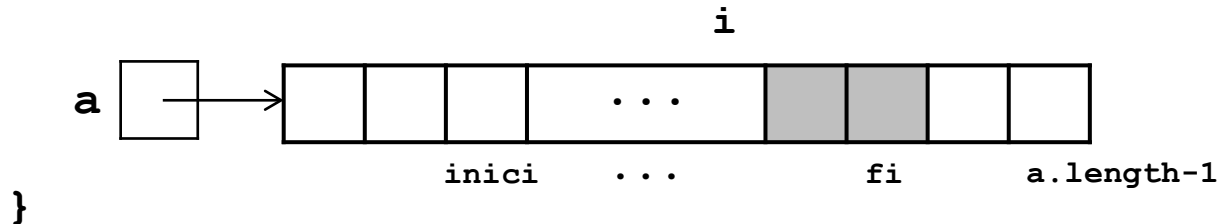
des d'una posició fi fins a una posició inici

$$0 \leq \text{inici} \leq \text{fi} < \text{a.length}$$

```
int i = fi;
```



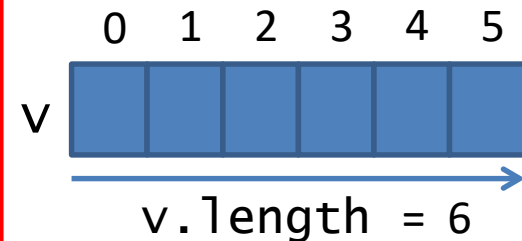
```
while (i >= inici) {  
    tractar(a[i]);  
    retrocedir(i);
```



Problemes de recorregut en arrays numèrics

- Exemple de **recorregut iteratiu ascendent**: mètode que suma tots els elements d'un array d'enters ($v.length > 0$).

```
public static int sumaIteAsc(int[] v) {  
    int suma = 0;  
    for (int i = 0; i < v.length; i++) {  
        suma = suma + v[i];  
    }  
    return suma;  
}
```



I si volem calcular la mitjana dels elements de l'array?

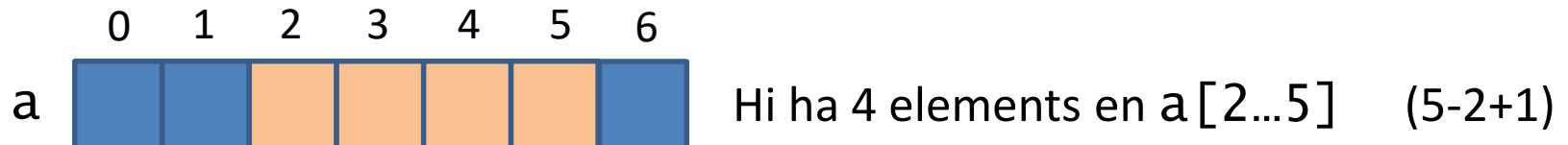
```
public static double mitjanaIteAsc(int[] v) {  
    double suma = 0;  
    for (int i = 0; i < v.length; i++) {  
        suma = suma + v[i];  
    }  
    return suma / v.length;  
}
```

Problemes de recorregut en arrays numèrics

- Exemple de processament de part d'un array: mètode que calcula la mitjana aritmètica del subarray d'enters $a[\text{esq} \dots \text{dre}]$, $0 \leq \text{esq} \leq \text{dre} < a.\text{length}$

```
public static double mitjana(int[] a, int esq, int dre) {  
    double suma = 0;  
    for (int i = esq; i <= dre; i++) { suma += a[i]; }  
    return suma / (dre - esq + 1);  
}
```

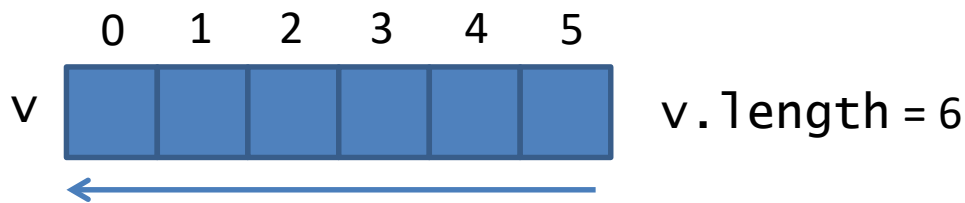
- El nombre d'elements entre esq i dre ($\text{esq} \leq \text{dre}$) és: $\text{dre} - \text{esq} + 1$



Problemes de recorregut en arrays numèrics

- Exemple de **recorregut iteratiu descendent**: mètode que suma tots els elements d'un array d'enters ($v.length > 0$).

```
public static int sumaIteDesc(int[] v) {  
    int suma = 0;  
    for (int i = v.length - 1; i >= 0; i--) {  
        suma = suma + v[i];  
    }  
    return suma;  
}
```



- El bucle es deté quan $i < 0$, és a dir, després de processar l'element que ocupa la posició 0 de l'array, quan i val -1 .

Problemes de recorregut en arrays d'String

- Exemple de **recorregut iteratiu ascendent**: Calcula la posició del màxim a l'array a ($a.length > 0$).

```
public static int màxim(String[] a) {  
    int posMax = 0;  
    for (int i = 1; i < a.length; i++) {  
        if (a[i].compareTo(a[posMax]) > 0) {  
            posMax = i;  
        }  
    }  
    return posMax;  
}
```

- Fixa't que el recorregut s'inicia en 1, ja que la posició 0 ja ha sigut tractada.
- Recorda que els tipus primitius es comparen amb `==`, `>`, `>=`, `<`, `<=` mentre que els tipus objecte es comparen amb els mètodes `equals()` i/o `compareTo()`.

Problemes de recorregut en arrays numèrics

- Exemple de **recorregut iteratiu descendent**: Calcula la posició del màxim a l'array a ($a.length > 0$).

```
public static int màxim(double[] a) {  
    int posMax = a.length - 1;  
    for (int i = a.length - 2; i >= 0; i--) {  
        if (a[i] > a[posMax]) { posMax = i; }  
    }  
    return posMax;  
}
```

- El recorregut s'inicia en $a.length - 2$, ja que la posició $a.length - 1$ ja ha sigut tractada.

Problemes de recorregut en arrays numèrics

BlueJ:exemplesT7

El mètode `main` de la classe `Comptadors` del projecte BlueJ *exemplesT7*, simula el llançament d'un dau de 10 cares (numerades del 0 al 9) i obté la freqüència d'aparició de cadascun dels valors.



```
Comptadors - exemplesT7
Classe  Edita  Eines  Opcions

Comptadors X
Compila  Desfés  Retalla  Copia  Enganxa  Cerca...  Tanca  Implementació ▼

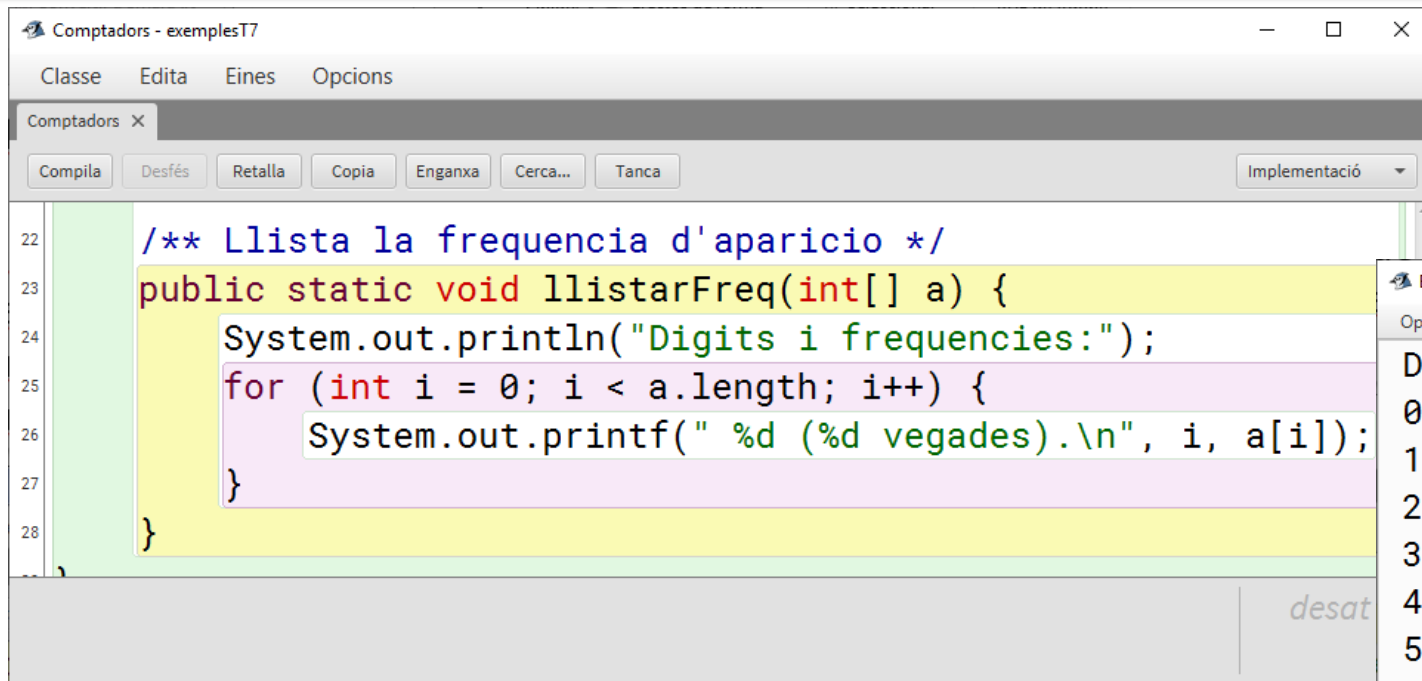
9 public class Comptadors {
10     private Comptadors() { }
11
12     public static void main(String[] args) {
13         int[] compt = new int[10];
14         int val;
15         for (int i = 0; i < 1000; i++) {
16             val = (int) (Math.random() * 10);
17             compt[val]++;
18         }
19         llistarFreq(compt);
20     }
}
```

desat

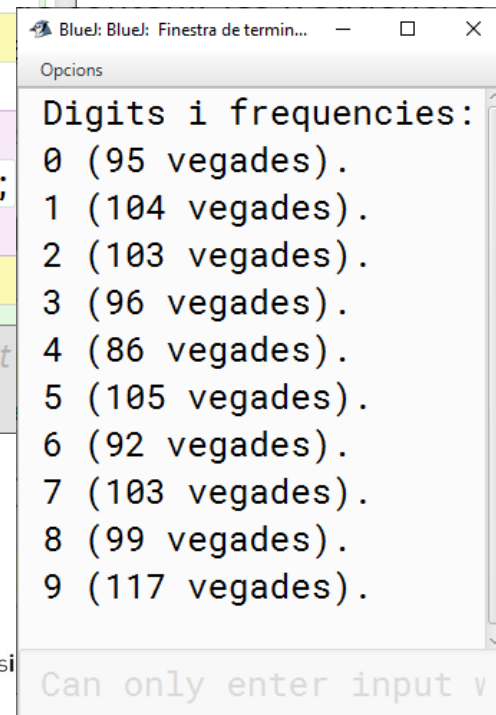
Problemes de recorregut en arrays numèrics

BlueJ:exemplesT7

El mètode `l·listarFreq` de la classe `Comptadors` del projecte BlueJ *exemplesT7*, té com a paràmetre un array d'enters `a` que ha de contenir les freqüències d'aparició dels índex `i` i fa un recorregut ascendent d'aquest array mostrant per pantalla cada índex (`i`) i la seua freqüència d'aparició (`a[i]`).



```
/** Llista la frecuencia d'aparicio */
public static void l·listarFreq(int[] a) {
    System.out.println("Digits i frecuencies:");
    for (int i = 0; i < a.length; i++) {
        System.out.printf(" %d (%d vegades).\n", i, a[i]);
    }
}
```



BlueJ: BlueJ: Finestra de termin...

Opcions

Digits i frecuencies:

- 0 (95 vegades).
- 1 (104 vegades).
- 2 (103 vegades).
- 3 (96 vegades).
- 4 (86 vegades).
- 5 (105 vegades).
- 6 (92 vegades).
- 7 (103 vegades).
- 8 (99 vegades).
- 9 (117 vegades).

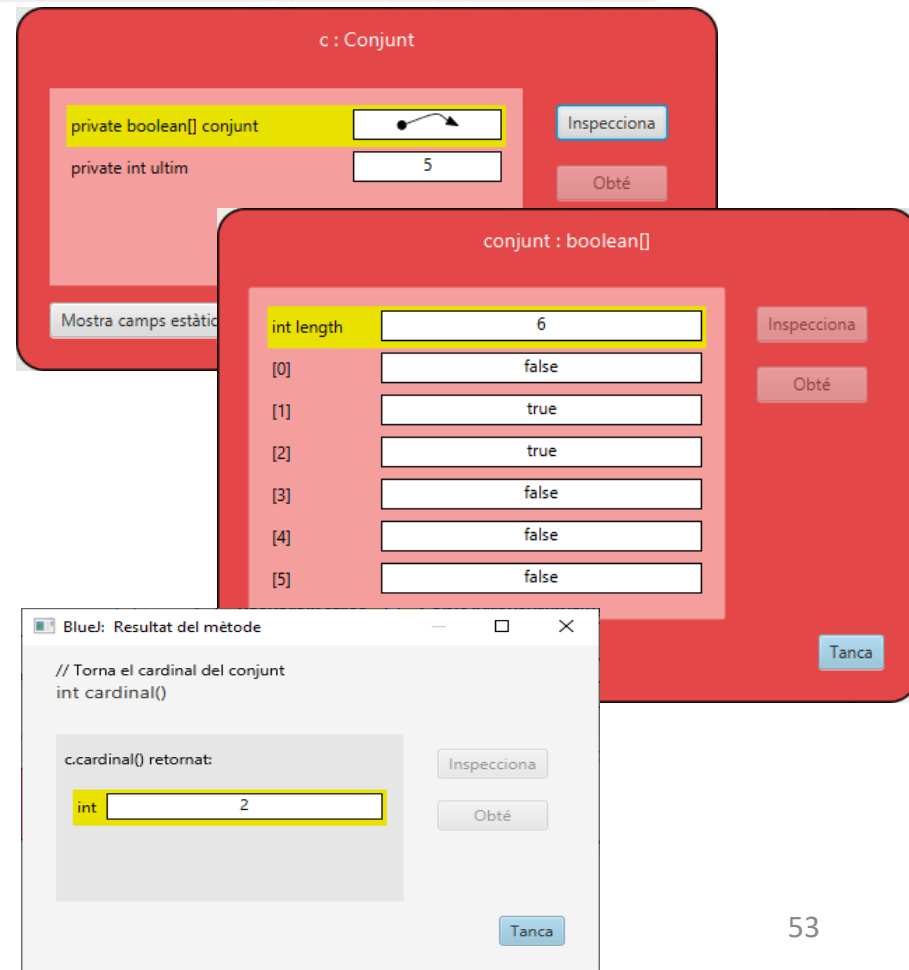
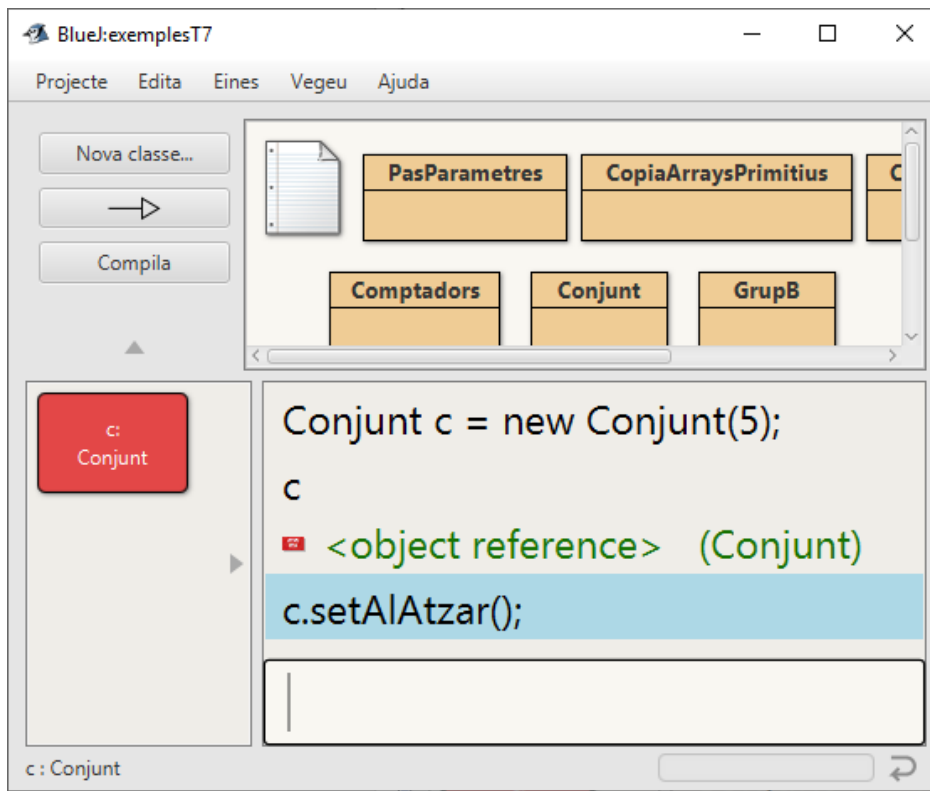
Can only enter input v

Problemes de recorregut en arrays numèrics

BlueJ:exemplesT7

El mètodes `cardinal` i `setAllAtzar` de la classe `Conjunt` del projecte BlueJ *exemplesT7* són exemples de mètodes de recorregut. Fixa't en la seua implementacio i, després, prova'ls des del *Code Pad* i el *Object Bench* de BlueJ.

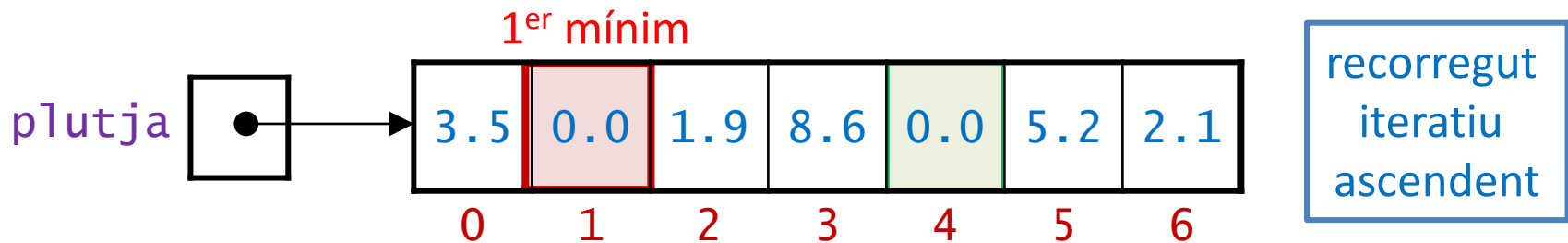
0 1 2
4 8



Problemes de recorregut en arrays numèrics

Blue!exercicisT7 [utilsArrays]

Completa en la classe **Recorregut** del paquet **utilsArrays**, el mètode **minimPlutja** per tal que, a partir de les mesures diàries de pluviositat d'una setmana, mostre per pantalla quina és la mesura **mínima** i el **dia** de la setmana que es va produir.

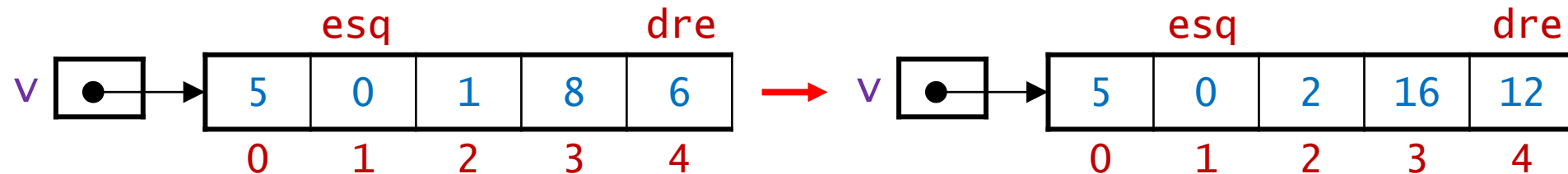


```
public static void minimPlutja(double[] plutja) {  
    double min = plutja[0];  
    int dia = 0;  
  
    System.out.println("Mínim: " + min + "Dia: " + (dia + 1));  
}
```

Problemes de recorregut en arrays numèrics

BlueJ:exercicisT7 [utilsArrays]

Completa en la classe **Recorregut** del paquet **utilsArrays**, el mètode **duplicar** per tal que donat un array d'enters **v** i dues posicions donades, **esq** i **dre**, de l'array, $0 \leq \text{esq} \leq \text{dre} \leq \text{v.length} - 1$, **duplique** el valor dels elements de l'array situats entre aquestes posicions.

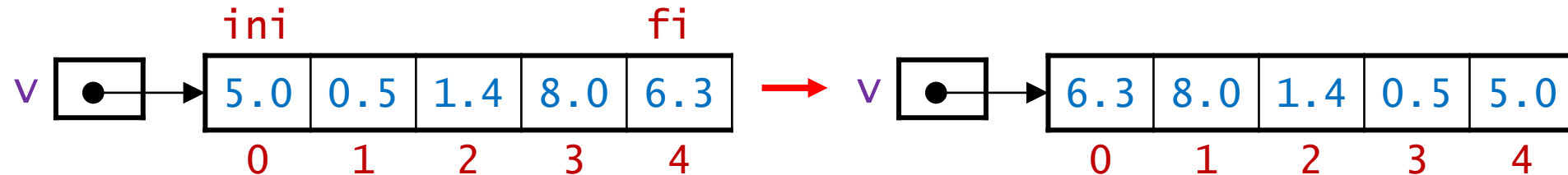


```
public static void duplicar(int[] v, int esq, int dre) {  
  
    recorregut iteratiu ascendent  
  
}
```

Problemes de recorregut en arrays numèrics

Blue!exercicisT7 [utilsArrays]

Completa en la classe **Recorregut** del paquet **utilsArrays**, el mètode **invertir** per tal que, donat un array de reals **v** i dues posicions donades, **ini** i **fi**, de l'array, $0 \leq \text{ini} \leq \text{fi} \leq \text{v.length} - 1$, **invertisca** tots els elements de l'array situats entre aquestes posicions.



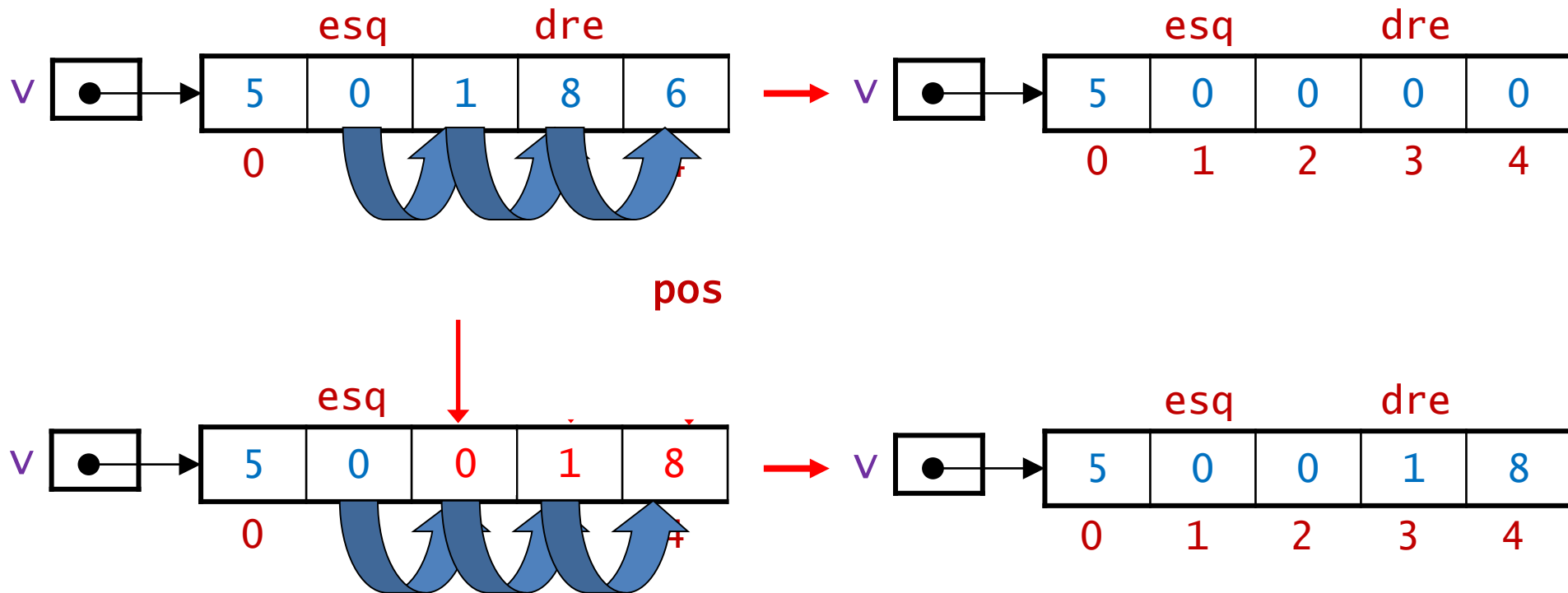
```
public static void invertir(double[] v, int ini, int fi) {  
  
  
  
  
  
  
  
  
  
}
```

recorregut iteratiu

Problemes de recorregut en arrays numèrics

Blue:exercicisT7 [utilsArrays]

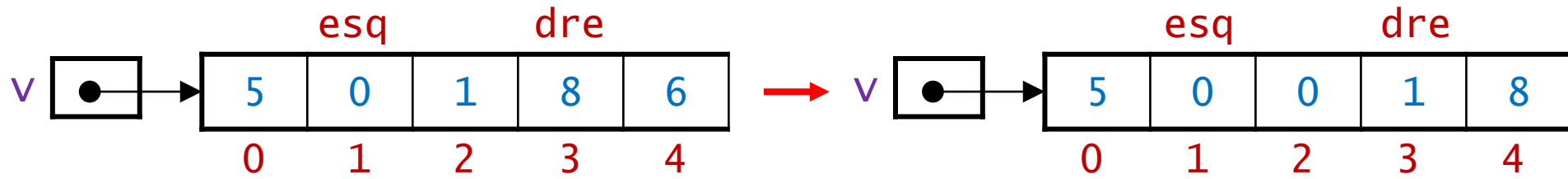
Completa en la classe **Recorregut** del paquet **utilsArrays**, el mètode **desplasarDreta** per tal que, donat un array d'enters **v** i dues posicions donades, **esq** i **dre**, de l'array, $0 \leq \text{esq} \leq \text{dre} < v.length - 1$, **desplace** una posició cap a la **dreta** tots els elements de **v** compresos entre les posicions **esq** i **dre**, ambues incloses.



Problemes de recorregut en arrays numèrics

BlueI:exercicisT7 [utilsArrays]

Completa en la classe **Recorregut** del paquet **utilsArrays**, el mètode **desplaçarDreta** per tal que, donat un array d'enters **v** i dues posicions donades, **esq** i **dre**, de l'array, $0 \leq \text{esq} \leq \text{dre} < v.length - 1$, **desplace** una posició cap a la **dreta** tots els elements de **v** compresos entre les posicions **esq** i **dre**, ambues incloses.



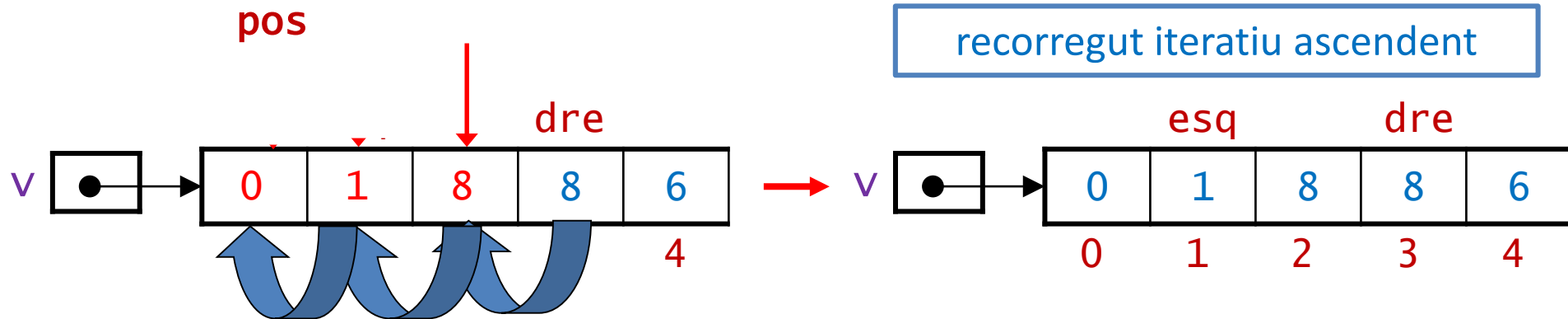
recorregut iteratiu descendent

```
public static void desplaçar(int[] v, int esq, int dre) {  
  
  
  
  
}
```

Problemes de recorregut en arrays numèrics

Blue!exercicisT7 [utilsArrays]

Completa en la classe **Recorregut** del paquet **utilsArrays**, el mètode **desplaçarEsquerra** per tal que, donat un array d'enters **v** i dues posicions donades, **esq** i **dre**, de l'array, $0 < \text{esq} \leq \text{dre} \leq \text{v.length} - 1$, **desplace** una posició cap a l'**esquerra** tots els elements de **v** compresos entre les posicions **esq** i **dre**, ambues incloses.



```
public static void desplaçar(int[] v, int esq, int dre) {  
  
  
  
  
  
  
}
```

Problemes de recorregut en arrays d'objectes

BlueJ:exercicisT7 [autobus]



- El mètode `numeroReserves` de la classe `Autobus` del paquet `autobus` torna el número de places ocupades en l'autobús comptant quantes posicions del seu array `places` són diferents de `null`.

```
21 public int numeroReserves() {  
22     int num = 0;  
23     for (int i = 0; i < places.length; i++) {  
24         if (places[i] != null) { num++; }  
25     }  
26     return num;  
27 }
```

bus:
Autobus

```
Autobus bus = new Autobus();  
bus.reservar(14, "Dubljevic");  
bus.reservar(9, "Van Rossom");  
bus.reservar(17, "Martinez");  
bus.reservar(1, "Stefansson");  
bus.reservar(16, "Vives");  
int n = bus.numeroReserves();  
n  
5 (int)  
bus  
<object reference> (Autobus)
```

bus : Autobus

private String[] places

Inspecciona

Obté

Mostra camps estàtics

Tanca

places : String[]

int length 50

Inspecciona

Obté

[0]	null
[1]	"Stefansson"
[2]	null
[3]	null
[4]	null
[5]	null
[6]	null
[7]	null
[8]	null
[9]	"Van Rossom"
[10]	null
[11]	null
[12]	null
[13]	null
[14]	"Dubljevic"
[15]	null
[16]	"Vives"
[17]	"Martinez"
[18]	null

Mostra camps estàtics

Tanca

Problemes de recorregut en arrays d'objectes

BlueJ:exercicisT7 [autobus]

- El mètode **nomMajor** de la classe **Autobus** del paquet **autobus** torna el nom del passatger de l'autobús que tinga el "nom major" segons l'ordre lèxicogràfic.
- Estableix un punt de ruptura en la línia de l'if i observa l'estat de les variables **i** i **res** al llarg de l'execució del codi.



```
29 public String nomMajor() {
30     String res = "";
31     for (int i = 0; i < places.length; i++) {
32         if (places[i] != null && places[i].compareTo(res) >= 0) {
33             res = places[i];
34         }
35     }
36     return res;
37 }
```

BlueJ: Resultat del mètode

// Torna el nom del passatger de l'Autobus que
// tinga el "nom major" segons l'ordre lèxicogràfic
String nomMajor()

bus.nomMajor() retornat:

String

Inspecciona

Obté

Tanca

Problemes de recorregut en arrays d'objectes

Blue:exercicisT7 [autobus]

- Completa en la classe **Autobus** del paquet **autobus** els següents mètodes:
 - **llistaPassatgers** que torna un String amb els noms dels passatgers i les places que ocupen.
 - **llistaSeientsLliures** que torna un String amb els números de seients que no estan ocupats.



Blue:exercicisT7 [agendaSenseOrdre]

- Completa en la classe **Agenda** del paquet **agendaSenseOrdre** el mètode **toString** que torna un String amb la informació de tots els contactes de l'agenda o, en cas que no hi hagen contactes a l'agenda, torna "Agenda buida".



Problemes de recorregut en arrays d'objectes

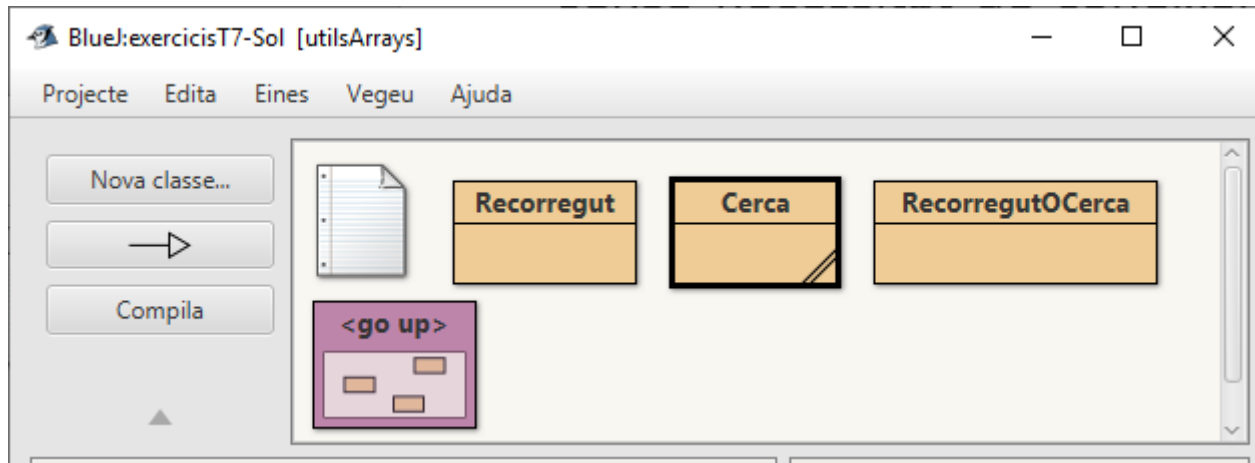
Exercici: Gestió d'un hostel rural



Method Summary	
All Methods	Instance Methods
Concrete Methods	
Modifier and Type	Method and Description
boolean	checkIn (java.lang.String nif, java.lang.String nom, Data arribada, Data eixida) Check in d'un client de nif nif, nom nom, data d'arribada arribada i data d'eixida eixida, tornant true si s'ha pogut fer i false en cas contrari (si no hi ha habitacions lliures).
double	checkOut (Data d) Check out de tots els clients tals que la seua data d'eixida es la Data d donada, tornant el preu total a pagar o 0 si no hi ha cap client amb aquesta data d'eixida.
double	checkOut (int i) Check out del client que ocupa l'habitacio i (sent i un numero d'habitacio valid), tornant el preu a pagar o 0 si l'habitacio no estava ocupada.
Client	getClient (int i) Torna el Client que ocupa l'habitacio i (sent i un numero d'habitacio valid) o null si l'habitacio esta lliure.
int	getClientsHistoric () Torna el numero de clients en l'historic.
int	getLliures () Torna el numero d'habitacions lliures.
int	getPC () Torna el numero d'habitacions ocupades per clients en regim de pensio completa.
boolean	hiHaLliures () Torna true si hi ha habitacions lliures i torna false en cas contrari.
int[]	pensioCompleta () Torna un array amb els numeros d'habitacions ocupades pels clients en regim de pensio completa.
int	primeraLliure () Torna el numero de la primera habitacio lliure (la de numero menor) si hi ha habitacions lliures o torna un -1 si no hi ha.
java.lang.String	toString () Torna un String que descriu l'Hostal, es a dir, quins clients ocupen quines habitacions i quines habitacions estan lliures.

Accés seqüencial: recorregut i cerca

- Una **cerca** requereix determinar si hi ha algun element de l'array que compleix una certa propietat.
- Aquest tipus de problemes impliquen operacions que poden obtenir la solució sense necessitat de conèixer totes les dades: **trobar el primer** que compleix cert requisit, tractar totes les dades **fins** que es compleixca **certa condició**, etc.
 - Les cerques a un array requereixen l'ús de variables de tipus **int** per accedir a les seues diferents posicions, de manera semblant als recorreguts, i de variables de tipus **boolean** per a comprovar la condició de terminació.
 - S'ha de portar el control de quines posicions ja s'han visitat i quines s'han de visitar per poder resoldre el problema.



Problemes de cerca en arrays

- S'estableixen dues estratègies principals a l'hora de buscar:
 - **Cerca lineal:** es va reduint l'**espai de cerca** (quantitat d'informació sobre la que buscar) element a element. Es pot aplicar sempre independentment de si les dades estan ordenades o no a l'array.
 - **Cerca binària o dicotòmica:** es va reduint l'**espai de cerca**, eliminant cada vegada la meitat d'elements. Aquesta estratègia necessita que les dades dintre de l'array estiguen **ordenades** d'una forma coneguda (per exemple, de menor a major), però és més eficient que les cerques lineals.

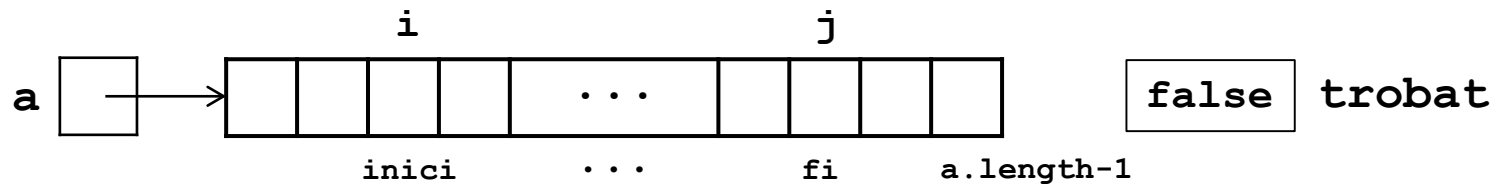
Cerca ascendent en un array

- Estructura de **cerca ascendent** usant una **variable lògica**: Existeix algun element $a[i]$ que compleix la propietat?

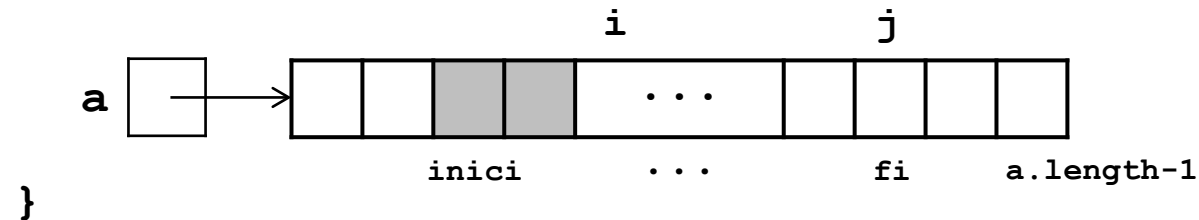
```
int i = inici, j = fi;
boolean trobat = false;
while (i <= j && !trobat) {
    if (propietat(a[i])) { trobat = true; }
    else { avançar(i); }
}
// Resoldre la cerca
if (trobat) ... // a[i] compleix la propietat
else ...      // cap element compleix la propietat
```

```
int i = inici, j = fi;
```

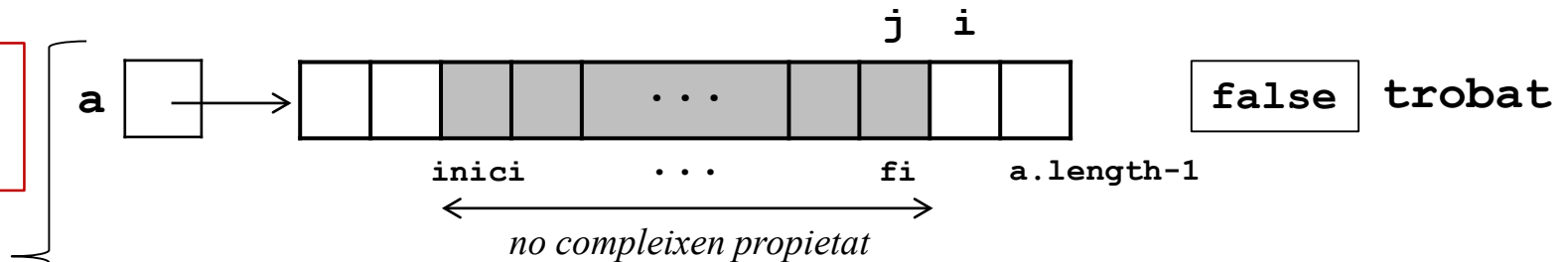
```
boolean trobat = false;
```



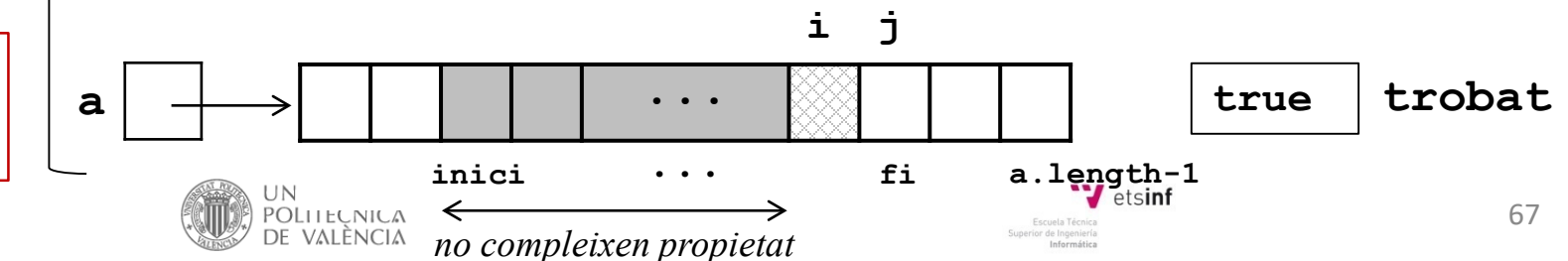
```
while (i <= j && !trobat) {  
    if (propietat(a[i])) { trobat = true; }  
    else { avançar(i); }  
}
```



cerca
sense èxit



cerca
amb èxit



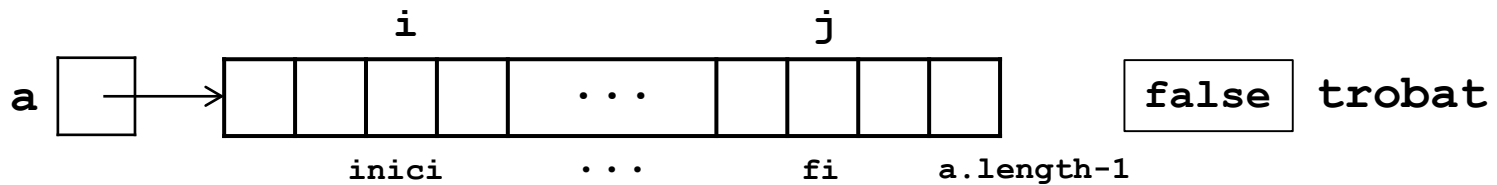
Cerca descendent en un array

- Estructura de **cerca descendent** usant una **variable lògica**: **Existeix algun element $a[i]$ que compleixca la propietat?**

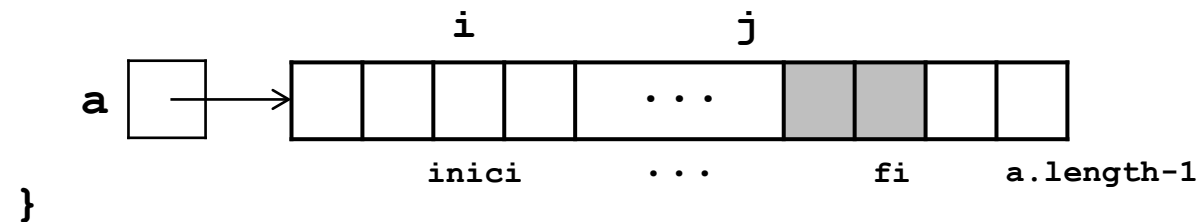
```
int i = inici, j = fi;
boolean trobat = false;
while (j >= i && !trobat) {
    if (propietat(a[j])) { trobat = true; }
    else { retrocedir(j); }
}
// Resoldre la cerca
if (trobat) ... // a[j] compleix la propietat
else ...      // cap element compleix la propietat
```

```
int i = inici, j = fi;
```

```
boolean trobat = false;
```

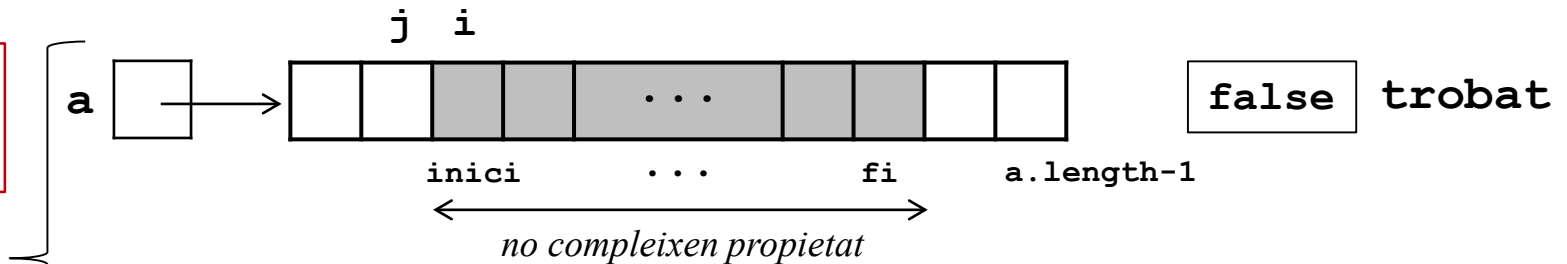


```
while (j >= i && !trobat) {  
    if (propietat(a[j])) { trobat = true; }  
    else { retrocedir(j); }
```

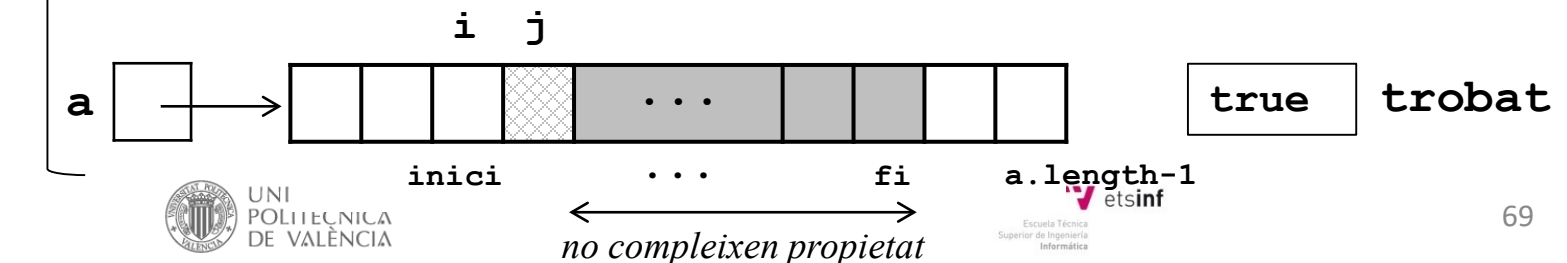


```
}
```

cerca
sense èxit



cerca
amb èxit



Estratègies de cerca en arrays

- Estructura de **cerca lineal iterativa** (sense garantia d'èxit) **amb guarda que avalua la propietat**. Algun element $a[i]$ compleix la propietat?

```
int i = inici, j = fi;
while (i <= j && !propietat(a[i])) {
    avançar(i);
}
// El bucle acaba perquè:
// i <= fi → a[i] compleix la propietat o
// i val fi + 1

// Resoldre la cerca
if (i <= fi) ... // Es compleix propietat(a[i])
else ...        // Cap element compleix propietat(a[i])
```

Estratègies de cerca en arrays

- Estructura de **cerca ascendent amb garantia d'èxit**: es coneix que hi ha algun element $a[i]$ que compleix la propietat.

```
int i = inici;  
while (!propietat(a[i])) {  
    avançar(i);  
}  
// Resoldre la cerca: En la posició i hi ha  
// un element que compleix la propietat
```

- És possible incloure deliberadament un element que compleixi la propietat. Aquest element s'anomena **sentinella**. En aquest cas, la cerca es diu **cerca amb sentinella**.

Problemes de cerca en arrays

- Cerca **ascendent** d'un element en un array de String (`a.length>0`) **sense garantia d'èxit**. Si no el troba, torna -1.

```
public static int cercarPos(String[] a, String dada) {  
    int i = 0;  
    while (i < a.length && !a[i].equals(dada)) { i++; }  
    if (i < a.length) { return i; }  
    else { return -1; }  
}
```

- Cerca **descendent** d'un element en un array de double (`a.length>0`) **sense garantia d'èxit**. Si no el troba, torna -1.

```
public static int cercarPos(double[] a, double dada) {  
    int i = a.length - 1;  
    while (i >= 0 && a[i] != dada) { i--; }  
    return i;  
}
```


Problemes de cerca en arrays

- Cerca **ascendent** d'un element en un array de String (`a.length>0`) amb garantia d'èxit.

```
public static int cercarPosEsta(String[] a, String dada) {  
    int i = 0;  
    while (!a[i].equals(dada)) { i++; }  
    return i;  
}
```

- Cerca **descendent** d'un element en un array de double (`a.length>0`) amb garantia d'èxit.

```
public static int cercarPosEsta(double[] a, double dada) {  
    int i = a.length - 1;  
    while (a[i] != dada) { i--; }  
    return i;  
}
```

Problemes de cerca en arrays

- Comprova si hi ha algun element de l'array (`a.length > 0`) més gran que `dada`.

```
public static boolean hiHaMajor(double[] a, double dada) {  
    int i = 0;  
    while (i < a.length && a[i] <= dada) { i++; }  
    return (i < a.length);  
}
```

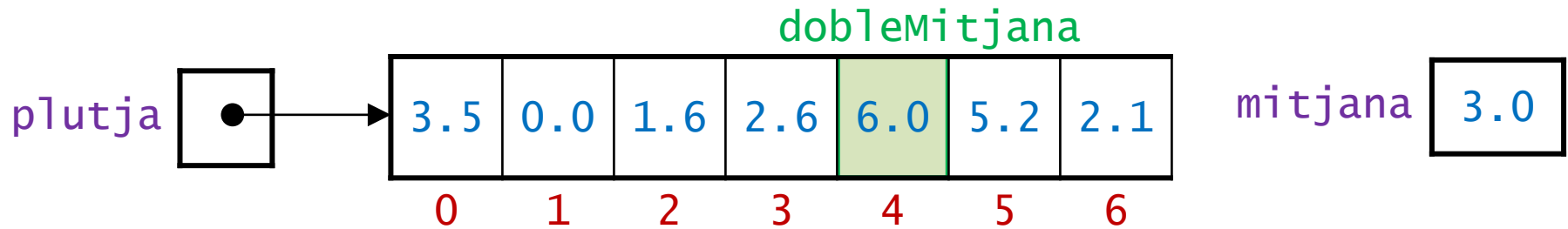
- Retorna, si existeix, la posició del primer element de l'array (`a.length > 0`) més gran que la suma dels anteriors. Si no, torna -1.

```
public static int cercaPosMajorQueSuma(int[] a) {  
    int i = 1, suma = a[0];  
    while (i < a.length && a[i] <= suma) { suma += a[i]; i++; }  
    if (i < a.length) { return i; }  
    else { return -1; }  
}
```

Problemes de cerca en arrays numèrics

Blue!exercicisT7 [utilsArrays]

Completa en la classe **Cerca** del paquet **utilsArrays**, el mètode **dobleMitjana** per tal que, a partir de les mesures diàries de pluviositat d'una setmana i de la mitjana d'aquestes mesures, indique si **algun dia** ha plogut el **doble** de la **mitjana**.



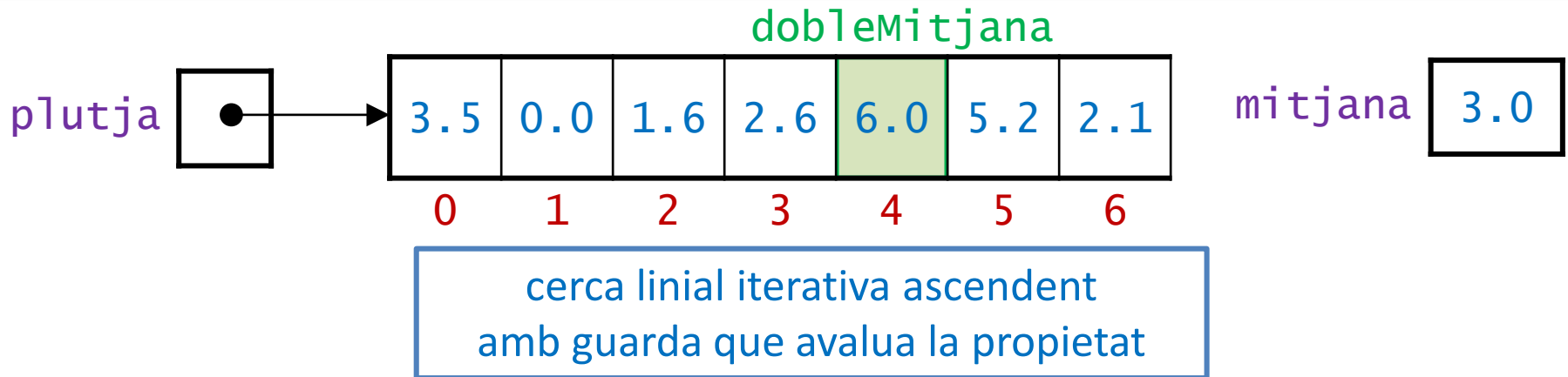
cerca linial iterativa ascendent amb variable boolean

```
public static boolean dobleMitjana(double[] plutja, double mitjana) {  
    double dobleMitjana = mitjana * 2;  
    boolean hiHaDoble = false;  
    int i = 0;  
  
    return hiHaDoble;  
}
```

Problemes de cerca en arrays numèrics

Blue:exercicisT7 [utilsArrays]

Completa en la classe **Cerca** del paquet **utilsArrays**, el mètode **dobleMitjana** per tal que, a partir de les mesures diàries de pluviositat d'una setmana i de la mitjana d'aquestes mesures, indique si **algun dia** ha plogut el **doble** de la **mitjana**.

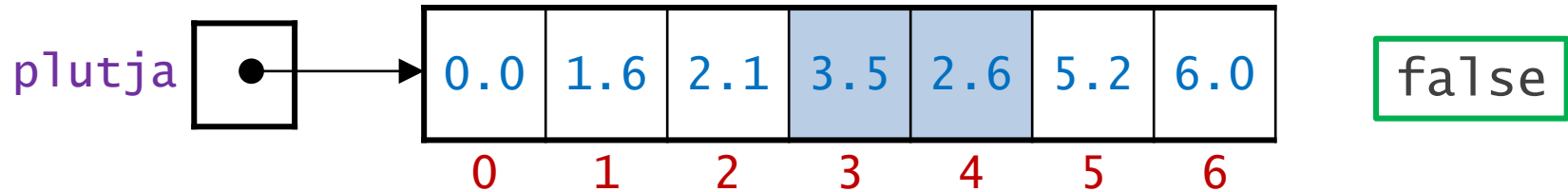
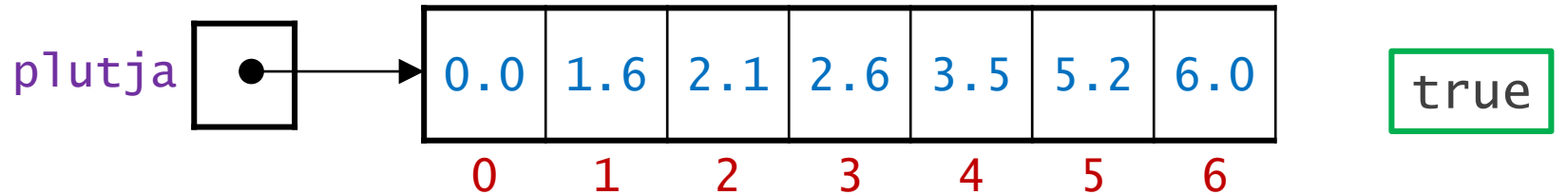


```
public static boolean dobleMitjana(double[] plutja, double mitjana) {  
    double dobleMitjana = mitjana * 2;  
    int i = 0;  
  
}
```

Problemes de cerca en arrays numèrics

Blue:exercicisT7 [utilsArrays]

Completa en la classe **Cerca** del paquet **utilsArrays**, el mètode **ordenAsc** per tal que, a partir de les mesures diàries de pluviositat d'una setmana a l'array **plutja**, indique si l'array està **ordenat ascendentment**.



cerca linial iterativa ascendent
amb guarda que avalua la propietat

```
public static boolean ordenAsc(double[] plutja) {  
    int i = 1;  
  
}
```

Problemes de cerca en arrays

```
public static boolean ordenAsc(double[] plutja) {  
    int i = 1;  
    while (plutja[i-1] <= plutja[i] && i < plutja.length) { i++; }  
    return i == plutja.length;  
}
```

És correcta?

```
public static boolean ordenAsc(double[] plutja) {  
    int i = 1;  
    while (i < plutja.length & plutja[i-1] <= plutja[i]) { i++; }  
    return i == plutja.length;  
}
```

És correcta?

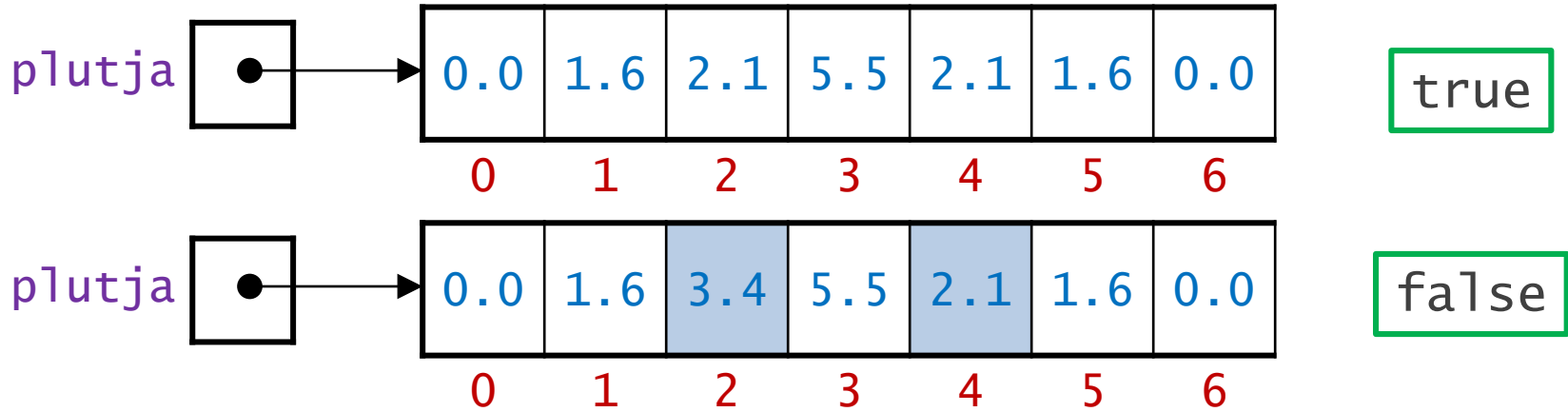
```
public static boolean ordenAsc(double[] plutja) {  
    int i = 1;  
    while (i < plutja.length && plutja[i-1] <= plutja[i]) { i++; }  
    if (plutja[i-1] > plutja[i]) { return false; }  
    else { return true; }  
}
```

És correcta?

Problemes de cerca en arrays numèrics

Blue:exercicisT7 [utilsArrays]

Completa en la classe **Cerca** del paquet **utilsArrays**, el mètode **esCapicua** per tal que, a partir de les mesures diàries de pluviositat d'una setmana a l'array **plutja**, indique si l'array és **capicua**.

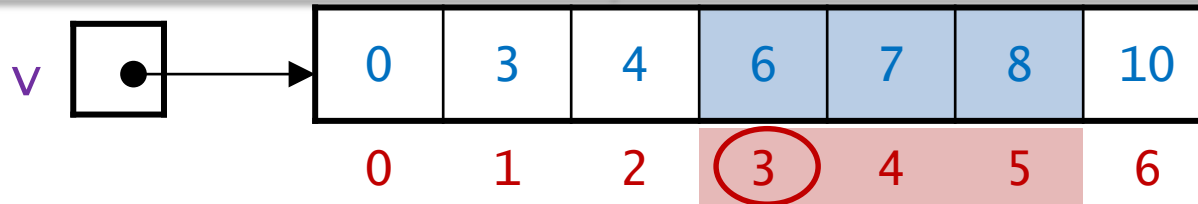


cerca linial iterativa ascendent
amb guarda que avalua la propietat

```
public static boolean esCapicua(double[] plutja) {  
    int i = 0, j = plutja.length - 1;  
  
    while (i < j) {  
        if (plutja[i] != plutja[j]) return false;  
        i++; j--;  
    }  
    return true;  
}
```

Problemes de cerca en arrays numèrics

Completa en la classe **Cerca** del paquet **utilsArrays**, el mètode **tresConsec** per tal que, donat **v** un array de **double**, determine la **posició**, si n'hi ha, de la primera subseqüència de l'array que compregui, almenys, **tres** nombres **enters consecutius** en **tres posicions consecutives** de l'array.

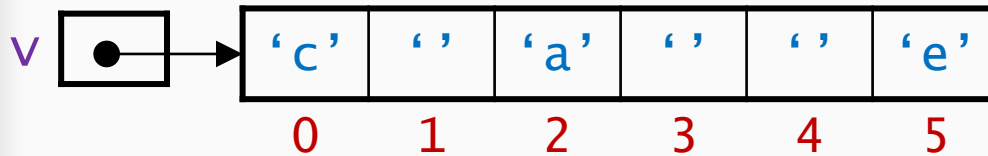
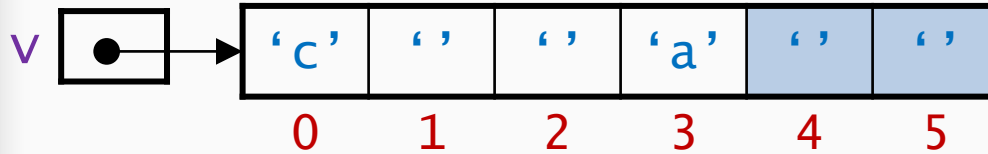


cerca linial iterativa ascendent amb variable boolean

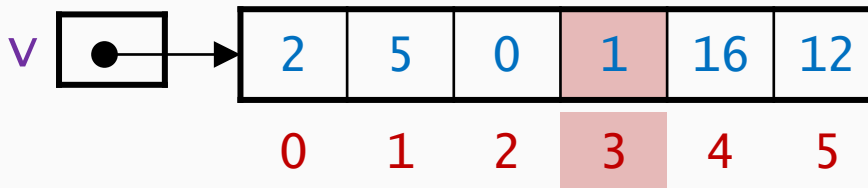
```
public static int tresConsec(double[] v) {  
    int i = 0, n = v.length;  
    boolean subsec = false;  
  
    if (subsec) { System.out.println("Comença en " + i);  
                return i; }  
    else { System.out.println("No es troba"); return -1; }  
}
```


Problemes de recorregut o de cerca?

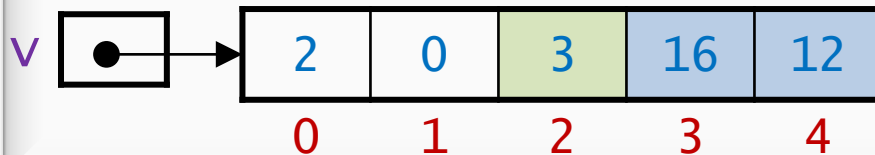
- Completa en la classe `RecorregutOCerca` del paquet `utilsArrays`:
 - El mètode `blancsFinal` per tal que, donat un array de caràcters `v`, **compte** quants **blancs** apareixen al **final** d'aquest.



- El mètode `posUltimSenar` per tal que, donat un array d'enters `v`, obtinga la **posició** de l'**últim** element **senar** de l'array.



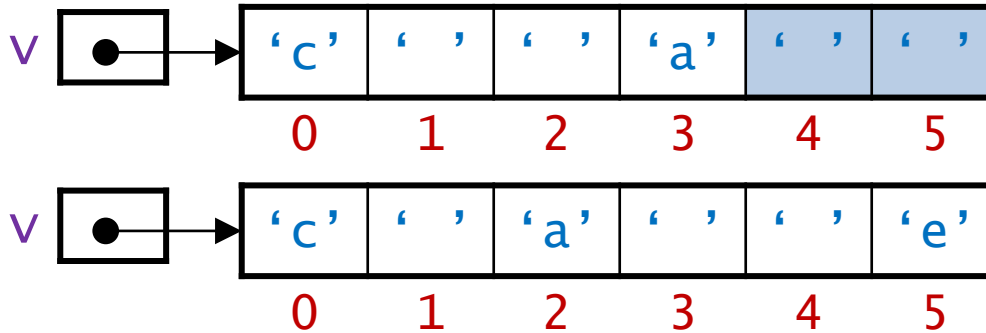
- El mètode `sumaDespresPrimerSenar` per tal que, donat un array d'enters positius `v`, **sume** els elements que apareixen **després** del **primer senar**.



Problemes de recorregut o de cerca?

Blue:exercicisT7 [utilsArrays]

- Completa en la classe **RecorregutOCerca** del paquet **utilsArrays** el mètode **blancsFinalAsc** per tal que, donat un array de caràcters **v**, **compte** quants **blancs** apareixen al **final** d'aquest.



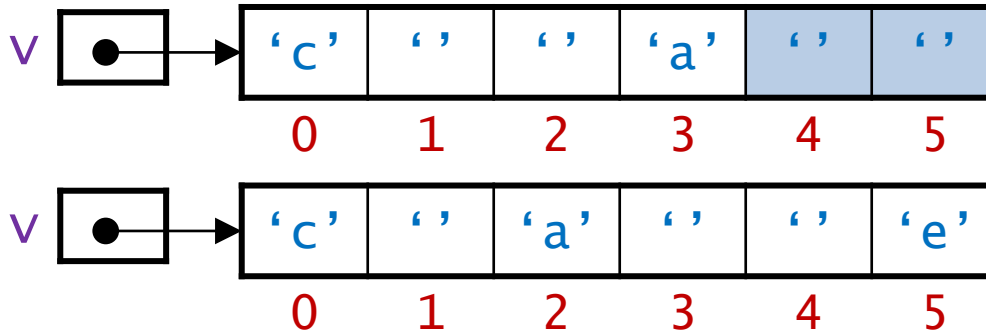
recorregut iteratiu
ascendent

```
public static int blancsFinalAsc(char[] v) {  
    int blancsFinal = 0;  
  
    return blancsFinal;  
}
```

Problemes de recorregut o de cerca?

Blue:exercicisT7 [utilsArrays]

- Completa en la classe **RecorregutOCerca** del paquet **utilsArrays** el mètode **blancsFinalDesc** per tal que, donat un array de caràcters **v**, **compte** quants **blancs** apareixen al **final** d'aquest.



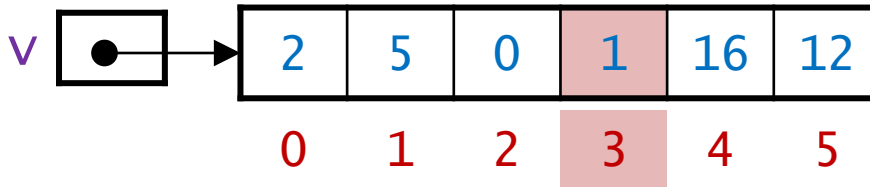
cerca linial iterativa
descendent

```
public static int blancsFinalDesc(char[] v) {  
    int i = v.length - 1, blancsFinal = 0;  
  
    return blancsFinal;  
}
```

Problemes de recorregut o de cerca?

Blue:exercicisT7 [utilsArrays]

- Completa en la classe **RecorregutOCerca** del paquet **utilsArrays** el mètode **posUltimSenarAsc** per tal que, donat un array d'enters **v**, obtinga la **posició** de l'**últim** element **senar** de l'array.



recorregut iteratiu
ascendent

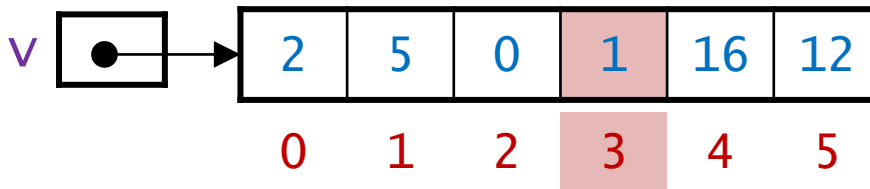
```
public static int posUltimSenarAsc(int[] v) {  
    int posUltSenar = -1;  
  
    return posUltSenar;  
}
```

- Si **posUltSenar** ≥ 0 la **posició** de l'**últim senar** és **posUltSenar** sino **no hi ha senars** a l'array i **posUltSenar** és -1

Problemes de recorregut o de cerca?

Blue:exercicisT7 [utilsArrays]

- Completa en la classe **RecorregutOCerca** del paquet **utilsArrays** el mètode **posUltimSenarDesc** per tal que, donat un array d'enters **v**, obtinga la **posició** de l'**últim** element **senar** de l'array.



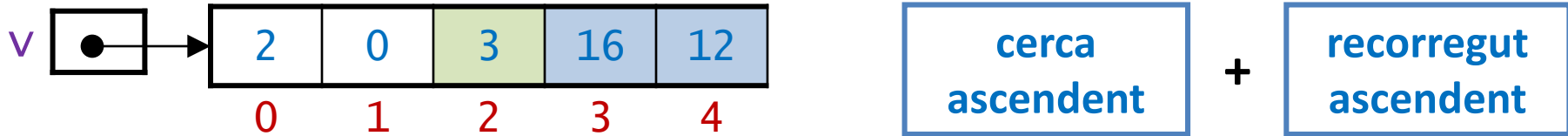
**cerca linial iterativa
descendent**

```
public static int posUltimSenarDesc(int[] v) {  
    int i = v.length - 1;  
  
    return i;  
}
```

- Si **i** ≥ 0 la **posició** de l'**últim senar** és **i**
sino **no hi ha senars** a l'array i **i** és **-1**

Problemes de recorregut o de cerca?

- Completa en la classe **RecorregutOCerca** del paquet **utilsArrays** el mètode **sumaDespresPrimerSenar** per tal que, donat un array d'enters positius **v**, **sume** els elements que apareixen **després** del **primer senar**.



```
public static int sumaDespresPrimerSenar(int[] v) {
    int i = 0, suma = 0;

    if (i == v.length) {
        System.out.println("No hi ha senars a l'array");
        return -1;
    }
    else {

        return suma;
    }
}
```

Problemes de cerca en arrays d'objectes

Blue!exercicisT7 [autobus]

- El mètode `estaCompleto` de la classe `Autobus` del paquet `autobus` torna true si l'autobus està complet (no queden places lliures) i, en cas contrari, torna false.



```
72 public boolean estaCompleto() {  
73     int i = 0;  
74     while (i < places.length && places[i] != null) { i++; }  
75     return i == places.length;  
76 }
```

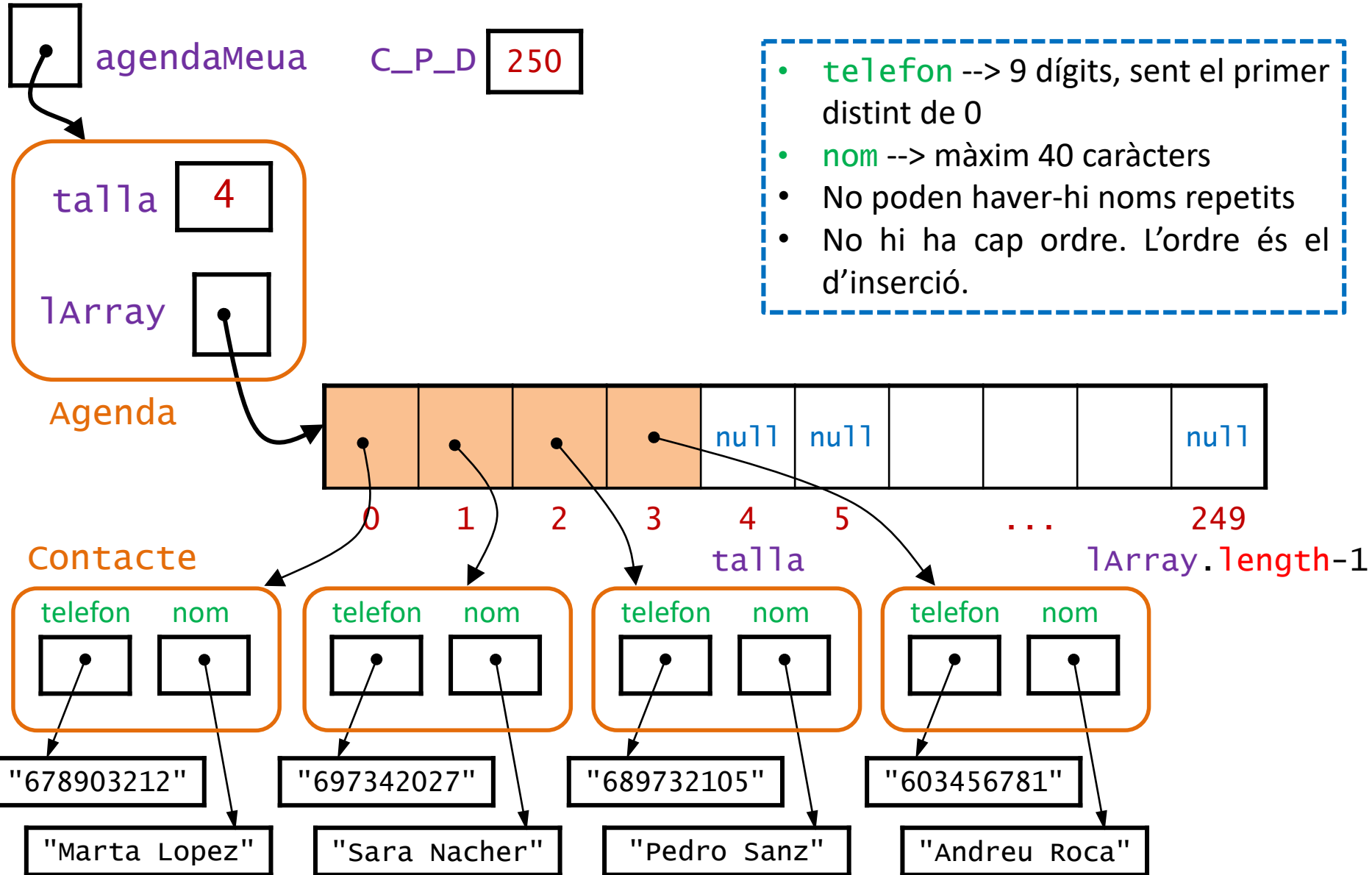
Problemes de cerca en arrays d'objectes

BlueI:exercicisT7 [agendaSenseOrdre]

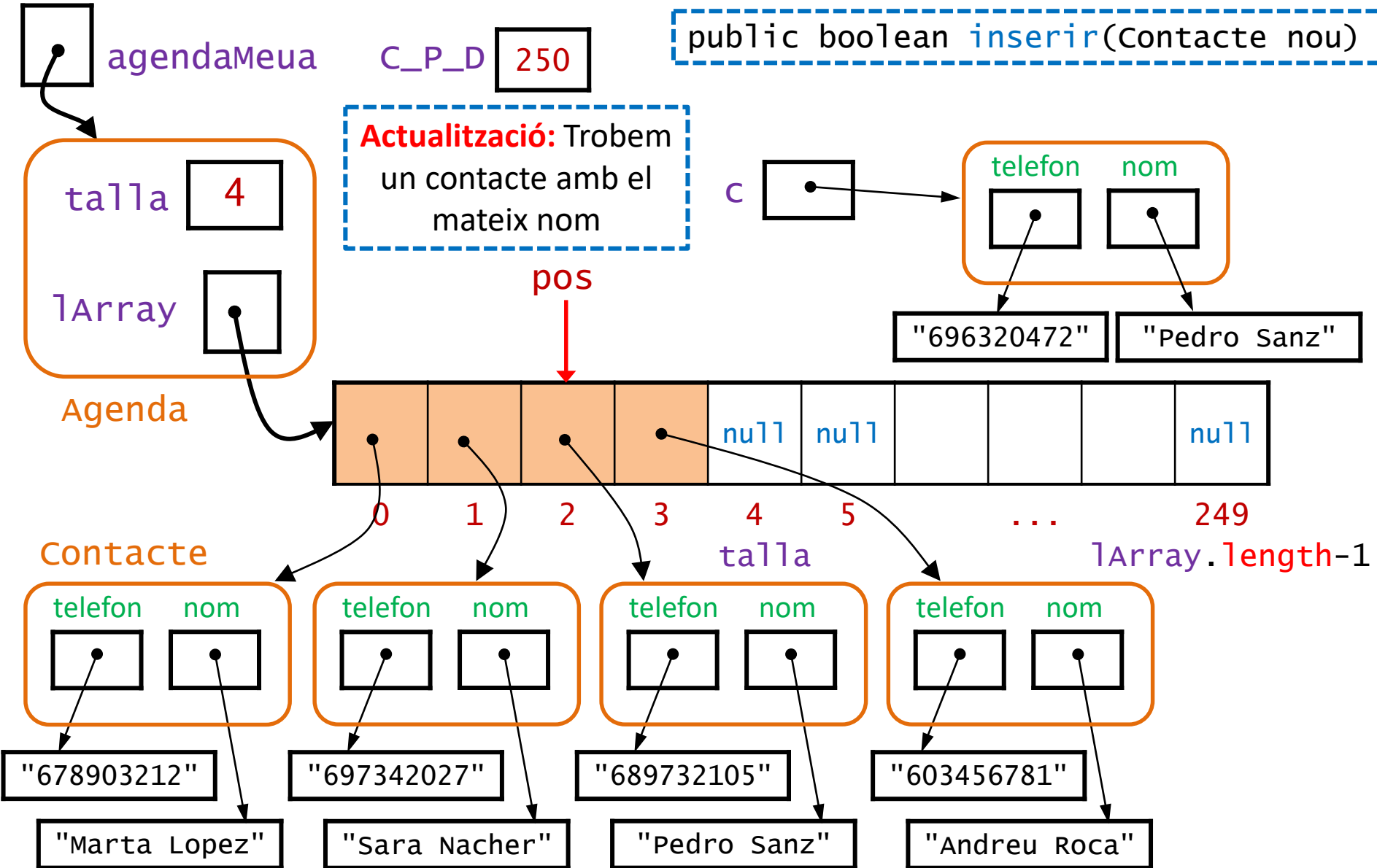
- Completa en la classe **Agenda** del paquet **agendaSenseOrdre** els mètodes:
 - **cercar** que cerca un contacte a l'agenda donat un nom. Torna la posició de l'element si el troba o -1 si no el troba.
 - **inserir** que afegeix un nou contacte vàlid a l'agenda o l'actualitza si ja existeix. Torna `true` si s'ha afegit amb èxit o `false` en cas que l'agenda estiga plena.
 - **eliminar** que donat un nom vàlid, elimina el **Contacte** de l'agenda amb eixe nom. Torna `true` si s'ha pogut eliminar i `false` si no s'ha pogut eliminar perquè no existeix un contacte amb eixe nom a l'agenda.



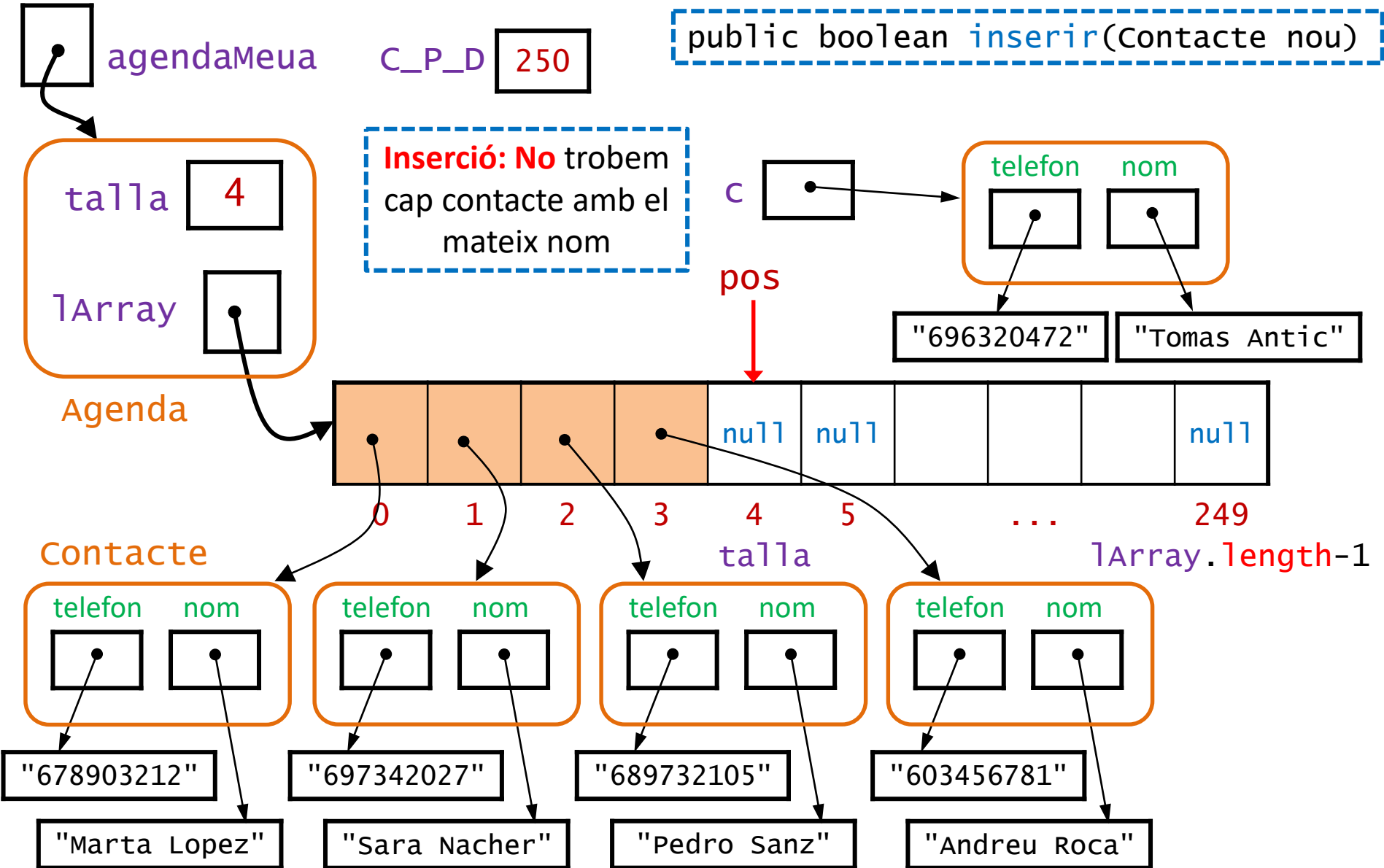
Agenda telefònica no ordenada



Agenda telefònica no ordenada



Agenda telefònica no ordenada



```

/** Cerca un contacte a l'agenda donat un nom. Torna la posició
 * de l'element si el troba o -1 si no el troba. */
private int cercar(String nom) {
    int i = 0;

}

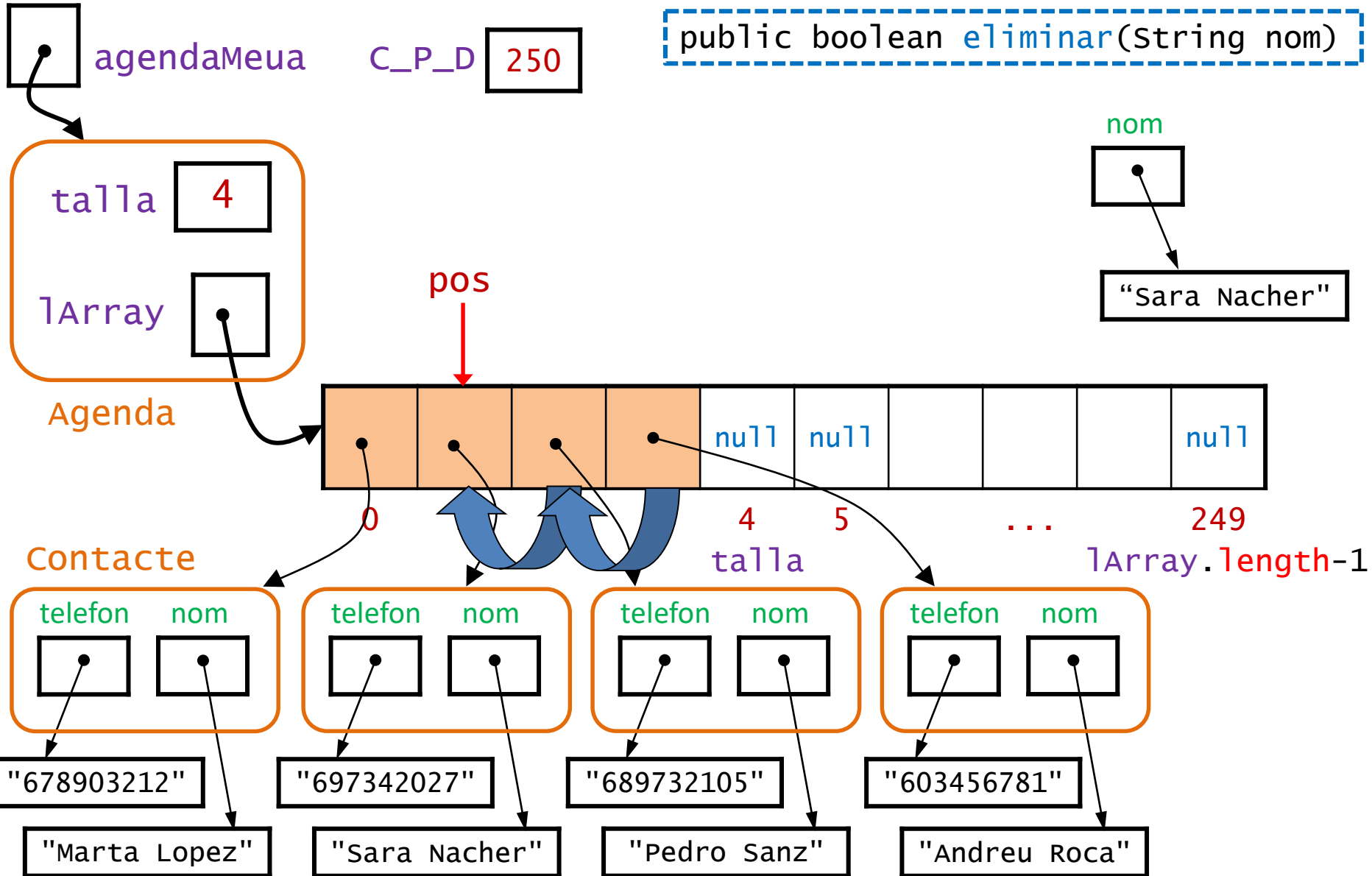
/** Afegeix un nou contacte vàlid a l'agenda o l'actualitza si ja
 * existeix. Torna true si s'ha afegit amb èxit o false en cas de
 * que l'agenda estiga plena. */
public boolean inserir(Contacte nou) {
    boolean cap = true;
    // cerca del nou contacte: si existeix, el reemplaça pel nou, l'actualitza;
    // sino, si i només si cap, insereix el nou contacte darrere de l'últim
    // existent, és a dir, en posició talla.
    int pos = cercar(nou.getNom());

    return cap;
}

```



Agenda telefònica no ordenada



```

/** Desplaça una posició cap a l'esquerra tots els elements de l'Array
 *  compresos entre les posicions ini i fi, ambdues incloses,
 *  0<ini<=fi<=lArray.length-1. */
private void desplacarEsq(int ini, int fi) {
    for (int pos = ini-1; pos < fi; pos++) { lArray[pos] = lArray[pos+1]; }
}

/** Donat un nom vàlid, elimina el Contacte de l'agenda amb eixe nom.
 *  Torna true si s'ha pogut eliminar i false si no s'ha pogut eliminar
 *  per que no existeix un contacte amb eixe nom a l'agenda.*/
public boolean eliminar(String nom) {
    boolean esta = true;
    // cerca, per nom, el contacte que es vol eliminar: si existeix, l'elimina:
    // desplaça una posició cap a l'esquerra tots els contactes posteriors al
    // que es va a eliminar, actualitza la talla, i torna true com resultat.
    // Si no troba el contacte a eliminar, torna false per a indicar-ho.
    int pos = cercar(nom);

    return esta;
}

```

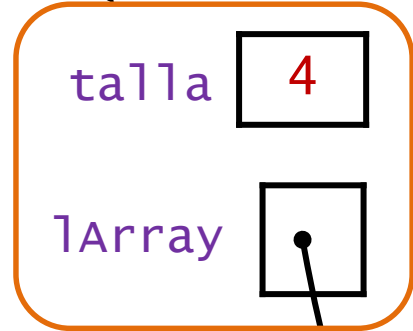


Agenda telefònica ordenada per nom

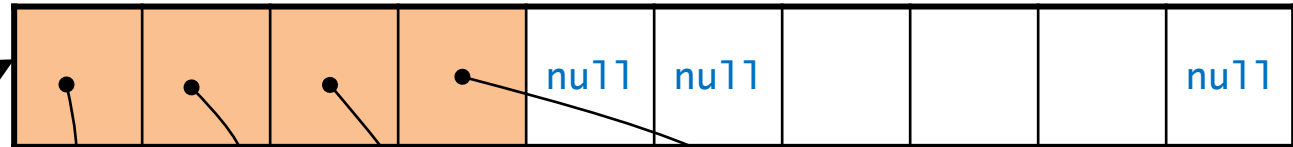


La **inserció** s'ha de fer **ordenada** pel **nom** del Contacte

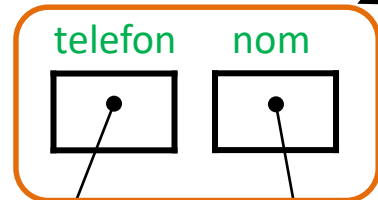
En quina **posició** s'afegirà el Contacte nou?



Agenda

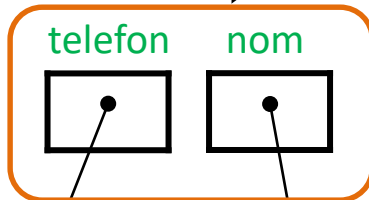


Contacte



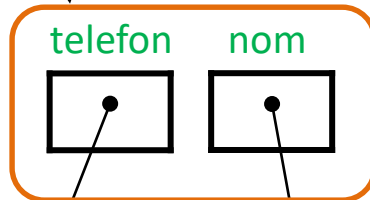
"603456781"

"Andreu Roca"



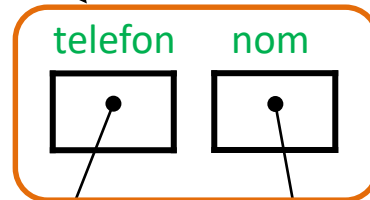
"678903212"

"Marta Lopez"



"689732105"

"Pedro Sanz"



"697342027"

"Sara Nacher"

talla

lArray.length-1

Agenda telefònica ordenada per nom

agendaMeua

La **inserció** s'ha de fer **ordenada** pel **nom** del Contacte

En quina **posició** s'afegirà el Contacte nou?

Actualització: Trobem un contacte amb el mateix nom

c

telefon

nom

"696320472"

"Pedro Sanz"

Agenda

pos

Contacte

telefon nom

"603456781"

"Andreu Roca"

telefon nom

"678903212"

"Marta Lopez"

telefon nom

"689732105"

"Pedro Sanz"

telefon nom

"697342027"

"Sara Nacher"

Agenda telefònica ordenada per nom

agendaMeua

La **inserció** s'ha de fer **ordenada** pel **nom** del Contacte

En quina **posició** s'afegirà el Contacte nou?

Inserció: Trobem un contacte amb un nom major alfabèticament

c

telefon

nom

"696320472"

"Nuria Antic"

Agenda

pos

Contacte

telefon nom



"603456781"

"Andreu Roca"

telefon nom



"678903212"

"Marta Lopez"

telefon nom



"689732105"

"Pedro Sanz"

telefon nom



"697342027"

"Sara Nacher"

Agenda telefònica ordenada per nom

agendaMeua

La **inserció** s'ha de fer **ordenada** pel **nom** del Contacte

En quina **posició** s'afegirà el Contacte nou?

Inserció: Trobem un contacte amb un nom major alfabèticament

c

telefon nom

"696320472"

"Ana Antic"

Agenda

pos

Contacte

telefon nom

"603456781"

"Andreu Roca"

telefon nom

"678903212"

"Marta Lopez"

telefon nom

"689732105"

"Pedro Sanz"

telefon nom

"697342027"

"Sara Nacher"

Agenda telefònica ordenada per nom

agendaMeua

La **inserció** s'ha de fer **ordenada** pel **nom** del Contacte

En quina **posició** s'afegirà el Contacte nou?

Inserció: No trobem cap contacte amb un nom major alfabèticament

c

pos

telefon

nom

"696320472"

"Tomas Antic"

Agenda

Contacte

telefon nom



"603456781"

"Andreu Roca"

telefon nom



"678903212"

"Marta Lopez"

telefon nom



"689732105"

"Pedro Sanz"

telefon nom



"697342027"

"Sara Nacher"

La **inserció** s'ha de fer **ordenada** pel **nom** del Contacte

Com hauríem de modificar **cercar** i **inserir** (de l'Agenda no ordenada)?



```
private int cercar(String nom) {  
    int i = 0;  
    while (i < talla && !lArray[i].getNom().equals(nom)) { i++; }  
    if (i < talla) { return i; }  
    else { return -1; }  
}
```

```
public boolean inserir(Contacte nou) {  
    boolean cap = true;  
    int pos = cercar(nou.getNom());  
    // cerca del nou contacte: si existeix, el reemplaça pel nou, l'actualitza;  
    // sino, si i només si cap, insereix el nou contacte darrere de l'últim  
    // existent, és a dir, en posició talla. en posició pos.  
    if (pos != -1) {  
        lArray[pos] = nou;  
    }  
    else if (talla < C_P_D) {  
        lArray[talla++] = nou;  
    } else { cap = false; }  
    return cap;  
}
```

```
private void desplaçarDre(int ini, int fi) {  
    for (int pos = fi+1; pos > ini; pos--) {  
        lArray[pos] = lArray[pos-1];  
    }  
}
```

Problemes de cerca en arrays d'objectes

Exercici: Gestió d'un hostel rural



Method Summary

All Methods

Instance Methods

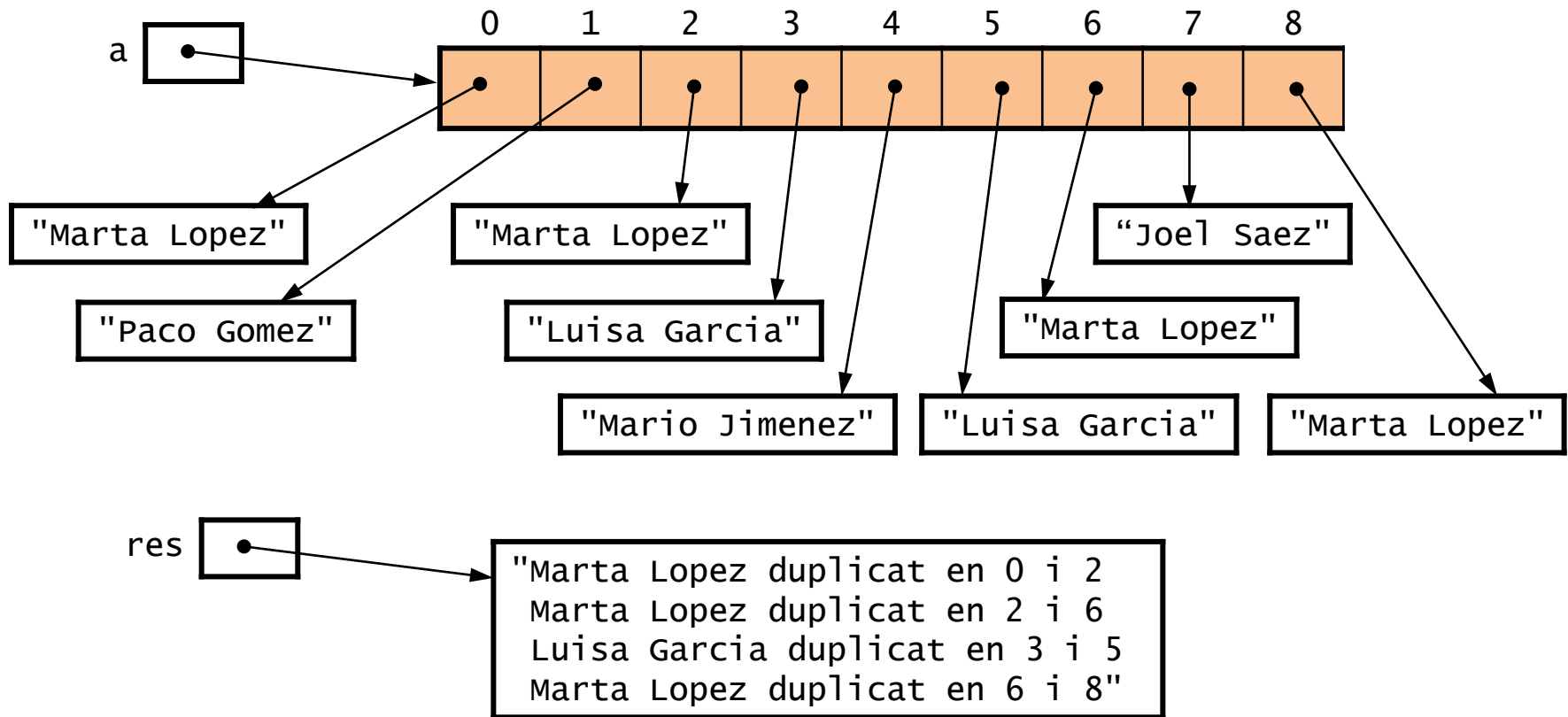
Concrete Methods

Modifier and Type	Method and Description
boolean	checkIn (java.lang.String nif, java.lang.String nom, Data arribada, Data eixida) Check in d'un client de nif nif, nom nom, data d'arribada arribada i data d'eixida eixida, tornant true si s'ha pogut fer i false en cas contrari (si no hi ha habitacions lliures).
double	checkOut (Data d) Check out de tots els clients tals que la seua data d'eixida es la Data d donada, tornant el preu total a pagar o 0 si no hi ha cap client amb aquesta data d'eixida.
double	checkOut (int i) Check out del client que ocupa l'habitacio i (sent i un numero d'habitacio valid), tornant el preu a pagar o 0 si l'habitacio no estava ocupada.
Client	getClient (int i) Torna el Client que ocupa l'habitacio i (sent i un numero d'habitacio valid) o null si l'habitacio esta lliure.
int	getClientsHistoric () Torna el numero de clients en l'historic.
int	getLliures () Torna el numero d'habitacions lliures.
int	getPC () Torna el numero d'habitacions ocupades per clients en regim de pensio completa.
boolean	hiHaLliures () Torna true si hi ha habitacions lliures i torna false en cas contrari.
int[]	pensioCompleta () Torna un array amb els numeros d'habitacions ocupades pels clients en regim de pensio completa.
int	primeraLliure () Torna el numero de la primera habitacio lliure (la de numero menor) si hi ha habitacions lliures o torna un -1 si no hi ha.
java.lang.String	toString () Torna un String que descriu l'Hostal, es a dir, quins clients ocupen quines habitacions i quines habitacions estan lliures.

```
private void cercar(String nif)
private void afegirHistoric(Client c)
```

Esquemes combinats de cerca i recorregut

- Hi ha problemes que requereixen combinar ambdues estratègies:
 - Donat un array de String, determinar per a cada String de l'array la primera repetició. El resultat ha de ser un String en el qual en cada línia estiga la dada i les dues posicions en les quals apareix.



Esquemes combinats de cerca i recorregut

Blue:exercicisT7 [utilsArrays]

- Fixa't en el mètode `duplicatsLlista` de la classe `RecorregutOCerca` del paquet `utilsArrays` que, donat un array de `String`, determina per a cada `String` de l'array la primera repetició.
- Executa el `main` que prova el mètode `duplicatsLlista`.

→ Recorregut de l'array en què per a cada element es realitza, a la vegada, la cerca d'una component igual a ell des de la seua posició en endavant.

```
105 public static String duplicatsLlista(String[] a) {
106     String res = "";
107     for (int i = 0; i < a.length - 1; i++) {
108         int j = i + 1;
109         while (j < a.length && !a[i].equals(a[j])) { j++; }
110         if (j < a.length) {
111             res += a[i] + " duplicat en: " + i + " i " + j + "\n";
112         }
113     }
114     return res;
115 }
```

Més problemes d'accés directe

- L'array permet l'accés en temps constant a qualsevol component, coneguda la seua posició.
 - En l'array $a[0..999]$, $a[0]$ té el mateix cost temporal que $a[999]$.
- Això permet implementar alguns algorismes de forma molt eficient. Exemples:
 - Cerca binària o dicotòmica (es veurà en PRG)
 - Permet la cerca eficient sobre un array ordenat.
 - Càlcul de la moda d'un multiconjunt de naturals
 - El membre del multiconjunt que més vegades es repeteix.

Moda d'un multiconjunt de naturals

- La moda d'un multiconjunt és l'element d'aquest que més vegades apareix.
- Estratègia en dues fases:
 - Fase 1: Recorregut seqüencial ascendent i array auxiliar de comptadors (**freqüència**) on **freqüència[i]**: nº de vegades que apareix **i** a l'array.
 - Fase 2: La moda s'obté calculant el màxim de l'array **freqüència**.
- Exemple per a $C = \{5, 7, 1, 7, 8, 7, 3, 0, 7, 8, 5, 0\}$ amb $n = 8$ (nº més gran).



Moda d'un multiconjunt de naturals

BlueJ:exercicisT7 [utilsArrays]

- Fixa't en el mètode `modaDe0aN` de la classe `Recorregut` del paquet `utilsArrays` que, donat un array que conté elements enters en el rang $[0..n]$, calcula la moda (el valor que més es repeteix).
- Prova el mètode en el *Code Pad* de BlueJ.

```
public static int modaDe0aN(int[] a, int n) {  
    // es construeix un array entre 0 i n  
    int[] frequencia = new int[n + 1];  
    // recorregut d'a i obtencio de frequencies  
    for (int i = 0; i < a.length; i++) { frequencia[a[i]]++; }  
    // la moda es el maxim de l'array frequencia  
    int moda = 0;  
    for (int i = 1; i < frequencia.length; i++) {  
        if (frequencia[i] > frequencia[moda]) { moda = i; }  
    }  
    return moda;  
}
```

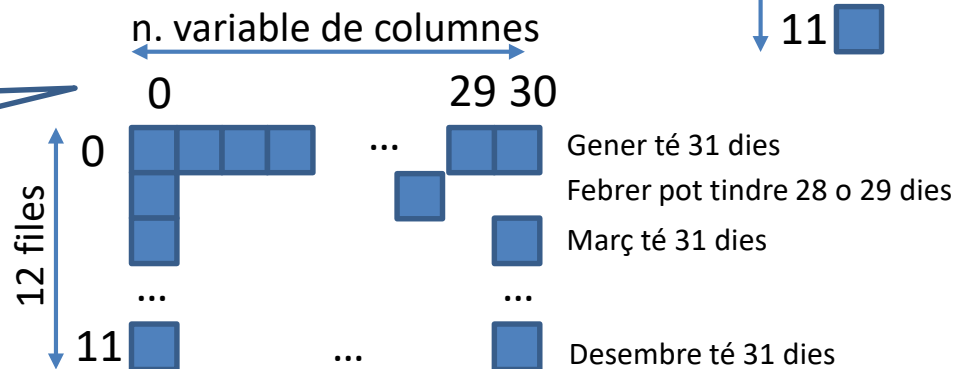
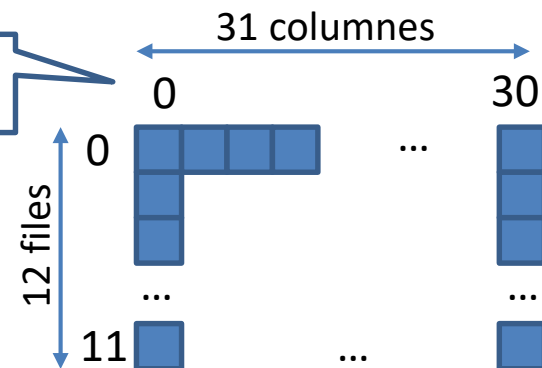
```
int[] a = {5, 7, 1, 7, 8, 7, 3, 0, 7, 8, 5, 0};  
int moda = Recorregut.modaDe0aN(a, 8);  
moda  
7 (int)
```

Arrays multidimensionals: Justificació



- Exemple d'aplicació: Mesures diàries de temperatura mitjana en una zona geogràfica.
 - Solucions possibles:
 - Array temp de 366 elements (per si l'any és bixest) que emmagatzema de forma correlativa les dades de temperatura per a cada dia.
 - 1 de gener \rightarrow temp[0] 3 de febrer \rightarrow temp[33] 2 de juliol \rightarrow temp[183]
 - Representació matricial de dimensions 12x31
 - Simplifica els càlculs de la posició real de cada dia en l'estructura de dades.
 - Representació mitjançant un array multidimensional amb 12 files i grandària variable de columnes:
 - Aquesta representació redueix el consum de memòria
- Sempre és convenient ajustar la mida de l'array a les dades que es vagen a emmagatzemar per reduir el consum de memòria.
- Es tracta d'un array de 12 components on cada component és al seu torn un array amb un nombre variable de components. Això és un exemple d'**array multidimensional** (array d'arrays).

Aquesta representació desaprofita memòria



Arrays multidimensionals: Definició, declaració i ús

- **Definició:** Un **array multidimensional** és un array els components del qual són també arrays.
- La **dimensió** d'un array és el nombre de definicions niuades que inclou. La dimensió defineix el nombre d'índexs que s'utilitzaran per accedir als elements primitius de l'array.
- Cada niuament és un array d'una dimensió menor.
- **Valors:** Es representen com una successió d'elements entre claus i separats per comes. Per exemple: `{{1, 2}, {1, 2, 3}, {1}}` és un array bidimensional d'enters amb tres arrays unidimensionals.
- **Operador:** L'operador d'accés `[]` ha de repetir-se per a cada dimensió.

`nomVble[índex1][índex2]...[índexN]`

índexk és una expressió que s'avalua a un índex vàlid per l'array `nomVble` en la dimensió **k**

- **Declaració i creació:**
 - Declaració: han d'aparèixer tants parells de claudàtors com la dimensió de l'array.
`tipus[][]...[] nomVble;`
 - Inicialització: han d'aparèixer tants parells de claudàtors com la dimensió de l'array i, almenys, la primera dimensió ha de tenir una mida específica. Les altres dimensions poden estar buides.

`nomVble = new tipus[expressió1]...[expressióN];`

- Conjuntament:

`tipus[][]...[] nomVble = new tipus[expressió1]...[expressióN];`

Arrays multidimensionals: Definició, declaració i ús

BlueJ:exercicisT7 [matrius]

Projecte Edita Eines Vegeu Ajuda

Nova classe... → Compila

UtilsMatrius <go up

a0: int[]

```
int[][] a = {{1, 2}, {1, 2, 3}, {1}};
```

a

<object reference> (int[][])

a[0]

<object reference> (int[])

a[0][1]

2 (int)

a0 : int[]

a : int[][]

int length	3
[0]	→
[1]	→
[2]	→

Inspect Get

Show static fields Close

[0] : int[]

int length	2
[0]	1
[1]	2

Inspect Get

Show static fields Close

[1] : int[]

int length	3
[0]	1
[1]	2
[2]	3

Inspect Get

Show static fields Close

[2] : int[]

int length	1
[0]	1

Inspect Get

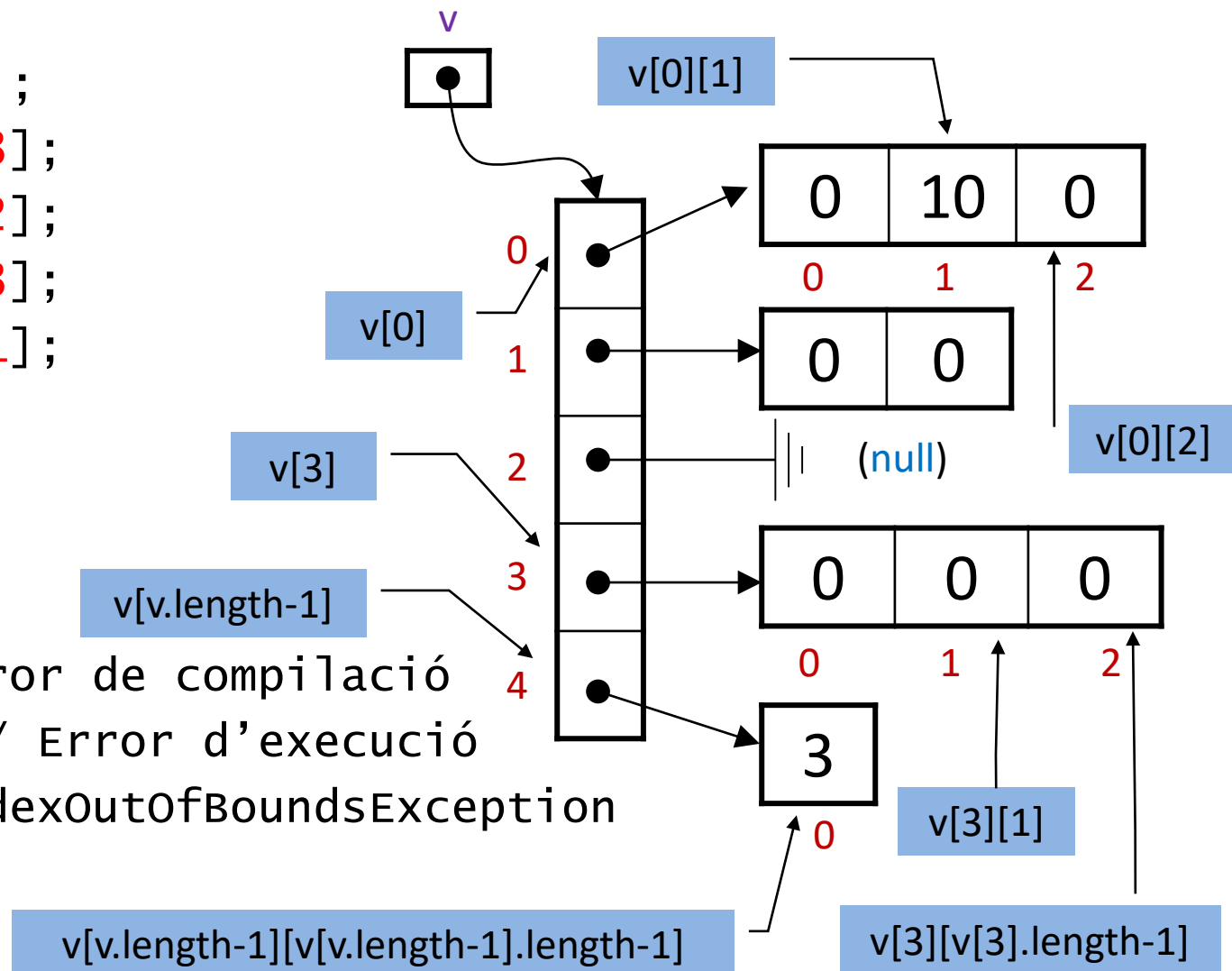
Show static fields Close

Arrays bidimensionals

```
int[][] v;  
v = new int[5][];  
v[0] = new int[3];  
v[1] = new int[2];  
v[3] = new int[3];  
v[4] = new int[1];  
v[0][1] = 10;  
v[4][0] = 3;
```

Errors:

```
v[2] = -3; //Error de compilació  
v[4][1] = -3; // Error d'execució  
// ArrayIndexOutOfBoundsException
```



Arrays bidimensionals: Matrius

- Les **matrius** són un cas particular d'arrays bidimensionals on tots els arrays de la segona dimensió tenen la mateixa mida.
- Es poden definir amb una única instrucció **new**.
- Exemple: matriu amb dues files i tres columnes:

```
double[][] matriu = new double[2][3];
```

- D'un altra forma:

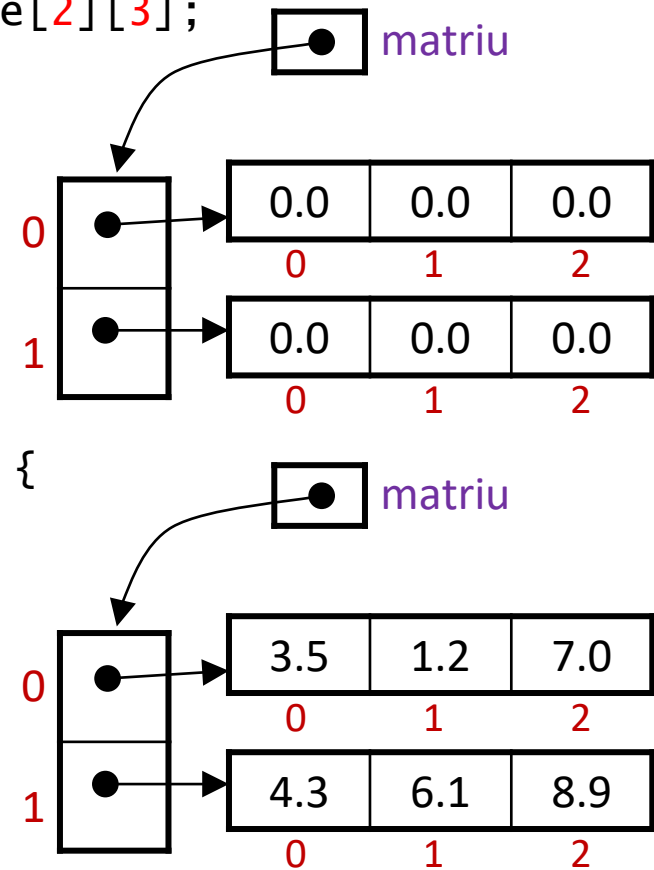
```
double[][] matriu = new double[2][];
```

```
matriu[0] = new double[3];
```

```
matriu[1] = new double[3];
```

```
for (int i = 0; i < matriu.length; i++) {  
    for (int j = 0; j < matriu[i].length; j++) {  
        matriu[i][j] = teclat.nextDouble();  
    }  
}
```

```
3.5  
1.2  
7  
4.3  
6.1  
8.9
```



Arrays de dimensió N

- Per a dimensions majors que 2 es segueixen les mateixes regles. Per exemple :
- Tres dimensions:

```
int[][][] tres = {{{1}}, {{1, 2}, {3, 4}}, {{1}, {2}}};  
int[][] dos = {{5, 4}, {6, 7}};  
tres[1] = dos;  
tres[1][0][1] = 6;
```

- Quatre dimensions:

```
int[][][][] quatre = new int[10][2][][];  
for (int i = 0; i < quatre.length; i++) {  
    for (int j = 0; j < quatre[i].length; j++) {  
        quatre[i][j] = new int[3][3];  
    }  
}
```

De forma equivalent:

```
int[][][][] quatre = new int[10][2][3][3];
```


Arrays bidimensionals

Exemple: Temperatura



- Declaració d'una matriu de 12x31 per emmagatzemar les mesures diàries de temperatura per a cada mes.

```
double[][] temp = new double[12][31];  
temp[3][6] = 50.1; // Temp. del dia 7 d'abril
```

- Declaració d'un array multidimensional de 12 files i nombre variable de columnes.

```
double[][] temp = new double[12][];  
// temp.length és igual a 12  
temp[0] = new double[31]; // Índexs del 0 al 30  
temp[1] = new double[29]; // Índexs del 0 al 28  
// ...  
temp[2][4] = 78.0; // Temp. del dia 5 de març
```

Arrays bidimensionals

Exemple: Temperatura



- Alternativa d'implementació amb definició prèvia del nombre de dies.

```
final int[] NUM_DIES = {31,28,31,30,31,30,31,31,30,31,30,31}  
// NUM_DIES[i] = n° dies del mes, 0 <= i <= 11  
  
double[][] temp = new double[12][];  
// temp[i] representa el mes, 0 <= i <= 11  
  
for (int i = 0; i < temp.length; i++) {  
    temp[i] = new double[NUM_DIES[i]];  
    // el n° elements de temp[i] = NUM_DIES[i], 0 <= i <= 11  
}
```

- `temp[i][j]` representa la temperatura mitjana del dia (`j+1`) del mes (`i+1`).
 - `temp[3][14]` és la temperatura mitjana del 15 d'abril.

Arrays bidimensionals

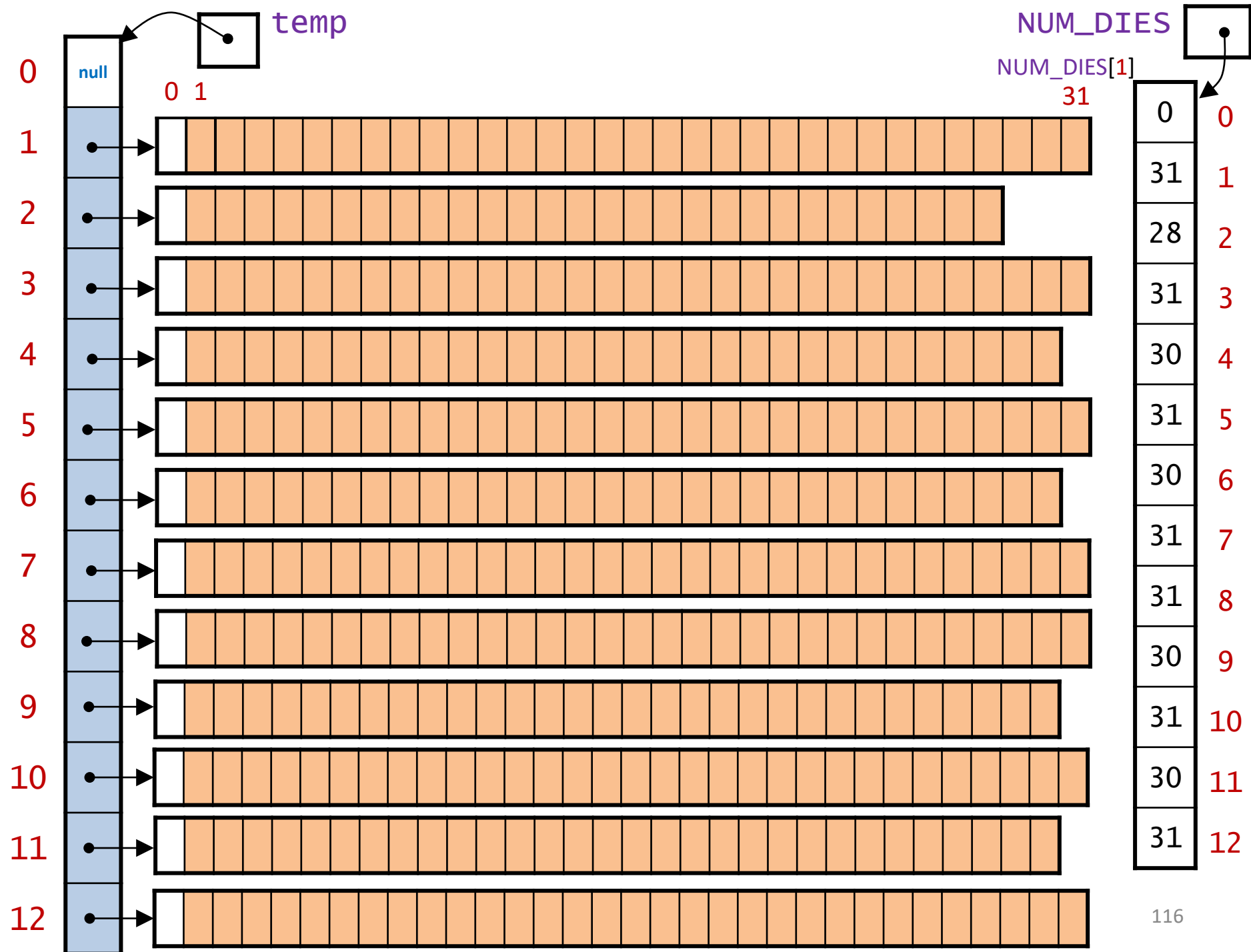
Exemple: Temperatura



- Alternativa d'implementació usant numeració des d'1.

```
final int[] NUM_DIES = {0,31,28,31,30,31,30,31,31,30,31,30,31};  
// NUM_DIES[0] = 0 i NUM_DIES[i] = n° dies del mes i, 1<=i<=12  
  
double[][] temp = new double[13][];  
// temp[0] = null i temp[i] representa el mes i, 1<=i<=12  
  
for (int i = 1; i < temp.length; i++) {  
    temp[i] = new double[NUM_DIES[i] + 1];  
    // el n° d'elements de temp[i] = NUM_DIES[i] + 1, 1<=i<=12  
}
```

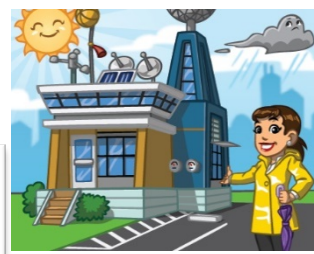
- Ara, `temp[i][j]` representa la temperatura mitjana del dia `j` del mes `i`.
 - `temp[4][15]` és la temperatura mitjana del 15 d'abril.



Recorregut en arrays bidimensionals

BlueJ:exemplesT7

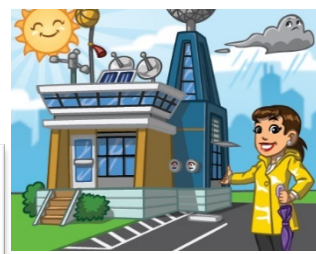
- En la classe **TemperaturaAnual** del projecte BlueJ **exemplesT7**:
- **mostrarTemp** mostra per pantalla la temperatura enregistrada cada dia de l'any.
- **mostrarPrimeraMiniDia** mostra la primera mesura **mínima** i **quan** es va produir.



```
45 public void mostrarTemp() {
46     for (int i = 1; i < temp.length; i++) {
47         System.out.println("Mes: " + i);
48         for (int j = 1; j < temp[i].length; j++) {
49             System.out.print("Dia: " + j);
50             System.out.println(" Temp: " + temp[i][j]);
51         }
52     }
53 }
```

```
62 public void mostrarPrimeraMiniDia() {
63     double tempMin = Double.MAX_VALUE;
64     int mesMin = 0, diaMin = 0;
65     for (int i = 1; i < temp.length; i++) {
66         for (int j = 1; j < temp[i].length; j++) {
67             if (temp[i][j] < tempMin) {
68                 tempMin = temp[i][j];
69                 mesMin = i;
70                 diaMin = j;
71             }
72         }
73     }
74     System.out.print("Mes: " + mesMin + " Dia: " + diaMin);
75     System.out.println(" Temp: " + tempMin);
76 }
```

Cerca en arrays bidimensionals

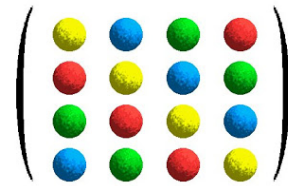


BlueJ:exemplesT7

- En la classe `TemperaturaAnual` del projecte BlueJ `exemplesT7`, el mètode `mostrarPrimerDia40` mostra per pantalla la primera data en la que es van enregistrar 40°.

```
82 public void mostrarPrimerDia40() {
83     int i = 1, j = 0;
84     boolean trobat = false;
85     while (i < temp.length && !trobat) {
86         j = 1;
87         while (j < temp[i].length && !trobat) {
88             trobat = (temp[i][j] == 40);
89             j++;
90         }
91         i++;
92     }
93     if (trobat) {
94         System.out.print("40° mesurat el dia " + (j - 1));
95         System.out.println(" del mes " + (i - 1));
96     }
97     else { System.out.println("Cap dia amb 40°"); }
98 }
```

Exemple: Suma de matrius



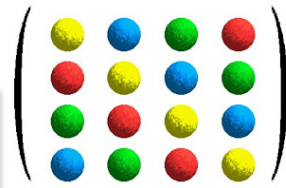
Blue:exercicisT7 [matrius]

- En la classe `UtilsMatrius` del paquet `matrius`, el mètode `sumaMat` torna la matriu resultant de sumar les matrius `a` i `b` d'igual dimensió.

a	0	1	2	3		b	0	1	2	3		c	0	1	2	3
0	a ₀₀	a ₀₁	a ₀₂	a ₀₃		0	b ₀₀	b ₀₁	b ₀₂	b ₀₃		0	a ₀₀ +b ₀₀	a ₀₁ +b ₀₁	a ₀₂ +b ₀₂	a ₀₃ +b ₀₃
1	a ₁₀	a ₁₁	a ₁₂	a ₁₃	+	1	b ₁₀	b ₁₁	b ₁₂	b ₁₃	=	1	a ₁₀ +b ₁₀	a ₁₁ +b ₁₁	a ₁₂ +b ₁₂	a ₁₃ +b ₁₃
2	a ₂₀	a ₂₁	a ₂₂	a ₂₃		2	b ₂₀	b ₂₁	b ₂₂	b ₂₃		2	a ₂₀ +b ₂₀	a ₂₁ +b ₂₁	a ₂₂ +b ₂₂	a ₂₃ +b ₂₃
3 x 4						3 x 4						3 x 4				
m x n						m x n						m x n				

```
/** a i b tenen la mateixa dimensió m x n, m>0, n>0 */
public static int[][] sumaMat(int[][] a, int[][] b) {
    int[][] c = new int[a.length][a[0].length];
    for (int i = 0; i < c.length; i++) {
        for (int j = 0; j < c[i].length; j++) {
            c[i][j] = a[i][j] + b[i][j];
        }
    }
    return c;
}
```

Exemple: Producte de matrius



- En la classe `UtilsMatrius` del paquet `matrius`, el mètode `producteMat` torna la matriu resultant de multiplicar les matrius `a` i `b` de dimensions `m x n` i `n x p`, respectivament.

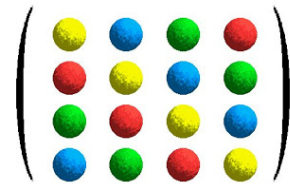
$$\begin{array}{c}
 \begin{array}{c} a \\ 0 \\ 1 \end{array} \begin{array}{cc} 0 & 1 & 2 \end{array} \\
 \begin{array}{|c|c|c|} \hline a_{00} & a_{01} & a_{02} \\ \hline a_{10} & a_{11} & a_{12} \\ \hline \end{array} \\
 \begin{array}{c} 2 \times 3 \\ m \times n \end{array}
 \end{array}
 \times
 \begin{array}{c}
 \begin{array}{c} 0 \\ 1 \\ 2 \end{array} \begin{array}{cc} b_{00} & b_{01} \\ b_{10} & b_{11} \\ b_{20} & b_{21} \end{array} \\
 \begin{array}{c} 3 \times 2 \\ n \times p \end{array}
 \end{array}
 =
 \begin{array}{c}
 \begin{array}{cc} 0 & 1 \end{array} \\
 \begin{array}{|c|c|} \hline a_{00}*b_{00}+a_{01}*b_{10}+a_{02}*b_{20} & a_{00}*b_{01}+a_{01}*b_{11}+a_{02}*b_{21} \\ \hline a_{10}*b_{00}+a_{11}*b_{10}+a_{12}*b_{20} & a_{10}*b_{01}+a_{11}*b_{11}+a_{12}*b_{21} \\ \hline \end{array} \\
 \begin{array}{c} 2 \times 2 \\ m \times p \end{array}
 \end{array}$$

```

/** a és de dimensió m x n i b de dimensió n x p, m>0, n>0, p>0 */
public static int[][] producteMat(int[][] a, int[][] b) {
    int[][] c = new int[a.length][b[0].length];
    for (int i = 0; i < c.length; i++) {
        for (int j = 0; j < c[i].length; j++) {
            c[i][j] = 0;
            for (int k = 0; k < a[i].length; k++) {
                c[i][j] = c[i][j] + (a[i][k] * b[k][j]);
            }
        }
    }
    return c;
}

```


Exercici: Producte de matriu per vector



BlueJ:exercisT7 [matrius]

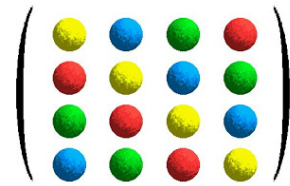
- En la classe `UtilMatrius` del paquet `matrius`, completa el mètode `matPerVec` per tal que torne el producte d'una matriu `a` per un vector `v`. La dimensió d'`a` és `m` x `n` i la de `v` és `n`, `m` > 0, `n` > 0.

$$\begin{array}{c} \text{a} \end{array} \begin{array}{c} 0 \quad 1 \quad 2 \quad 3 \\ \begin{array}{|c|c|c|c|} \hline 0 & a_{00} & a_{01} & a_{02} & a_{03} \\ \hline 1 & a_{10} & a_{11} & a_{12} & a_{13} \\ \hline 2 & a_{20} & a_{21} & a_{22} & a_{23} \\ \hline \end{array} \end{array} \begin{array}{c} \text{v} \\ \begin{array}{|c|} \hline 0 \quad b_0 \\ \hline 1 \quad b_1 \\ \hline 2 \quad b_2 \\ \hline 3 \quad b_3 \\ \hline \end{array} \end{array} = \begin{array}{c} \text{c} \\ \begin{array}{|c|} \hline 0 \quad a_{00} * b_0 + a_{01} * b_1 + a_{02} * b_2 + a_{03} * b_3 \\ \hline 1 \quad a_{10} * b_0 + a_{11} * b_1 + a_{12} * b_2 + a_{13} * b_3 \\ \hline 2 \quad a_{20} * b_0 + a_{21} * b_1 + a_{22} * b_2 + a_{23} * b_3 \\ \hline \end{array} \end{array}$$

3×4
 $m \times n$

$3 = m$

Exercici: Transposada d'una matriu



BlueJ:exercisT7 [matrius]

- En la classe `UtilsMatrius` del paquet `matrius`, completa el mètode `transpMat` per tal que torne la transposada d'una matriu `a` donada. La dimensió d'a és $m \times n$, $m > 0$, $n > 0$.

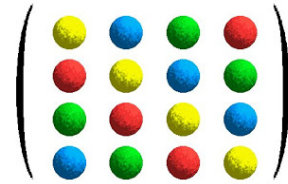
3 x 4
m x n

a	0	1	2	3
0	a_{00}	a_{01}	a_{02}	a_{03}
1	a_{10}	a_{11}	a_{12}	a_{13}
2	a_{20}	a_{21}	a_{22}	a_{23}

at	0	1	2
0	a_{00}	a_{10}	a_{20}
1	a_{01}	a_{11}	a_{21}
2	a_{02}	a_{12}	a_{22}
3	a_{03}	a_{13}	a_{23}

4 x 3
n x m

Exercici: Transposada d'una matriu quadrada



BlueJ:exercisT7 [matrius]

- En la classe `UtilsMatrius` del paquet `matrius`, completa el mètode `transpMatQuad` per tal que transpose la matriu quadrada `a` donada. La dimensió d'`a` és `n x n`, `n > 0`.

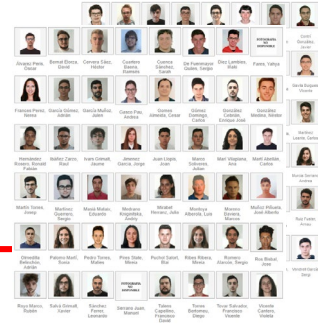
4 x 4
n x n

a	0	1	2	3
0	a_{00}	a_{01}	a_{02}	a_{03}
1	a_{10}	a_{11}	a_{12}	a_{13}
2	a_{20}	a_{21}	a_{22}	a_{23}
3	a_{30}	a_{31}	a_{32}	a_{33}

4 x 4
n x n

a^t	0	1	2	3
0	a_{00}	a_{10}	a_{20}	a_{30}
1	a_{01}	a_{11}	a_{21}	a_{31}
2	a_{02}	a_{12}	a_{22}	a_{32}
3	a_{03}	a_{13}	a_{23}	a_{33}

Exemple: grups de primer curs

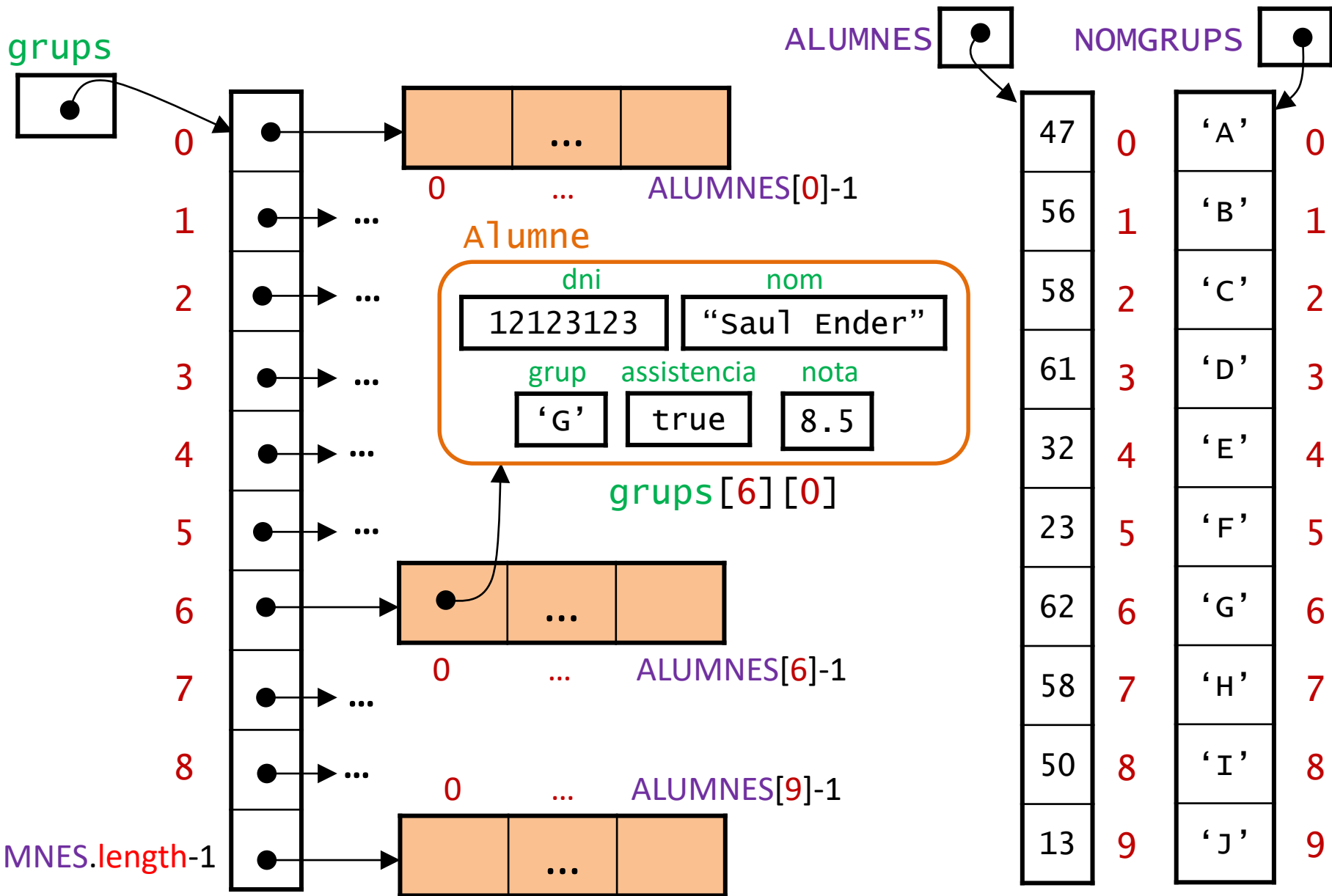


BlueJ:exercicisT7 [grupsPrimerCurs]

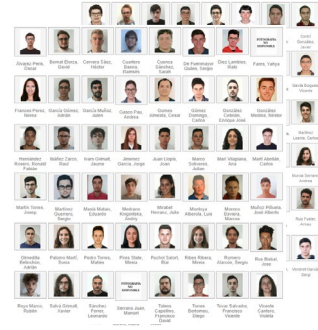
```
public class PrimerCurs {
    final int[] ALUMNES = {47,56,58,61,32,23,62,58,50,13};
    final char[] NOMGRUPS = {'A','B','C','D','E','F','G','H','I','J'};
    private Alumne[][] grups;

    public PrimerCurs(Scanner tec) {
        grups = new Alumne[ALUMNES.length][];
        for (int i = 0; i < grups.length; i++) {
            grups[i] = new Alumne[ALUMNES[i]];
            for (int j = 0; j < grups[i].length; j++) {
                grups[i][j] = new Alumne(tec);
                grups[i][j].setGrup(NOMGRUPS[i]);
            }
        }
    }
    ...
}
```

```
public class Alumne {
    private long dni;
    private double nota;
    private String nom;
    private boolean assistència;
    private char grup;
    ...
}
```

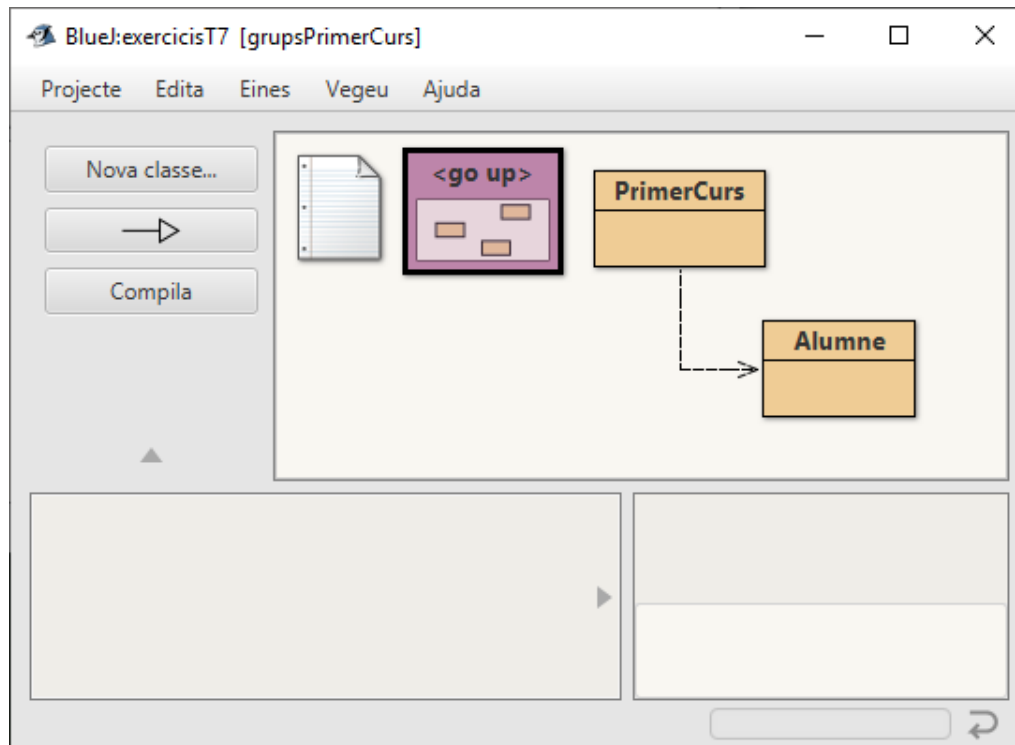


Exemple: grups de primer curs



Blue!exercicisT7 [grupsPrimerCurs]

- Per poder provar els mètodes de la classe **PrimerCurs** del paquet **grupsPrimerCurs**, s'ha definit un constructor que llig les dades d'un fitxer de text (**alumnes.txt**), evitant d'aquesta manera tenir que teclejar-los en cada prova.
- Comenta les declaracions de les constants **ALUMNES** i **NOMGRUPS** i descomenta les que hi ha definides a continuació per a 3 grups de 6, 5 i 7 alumnes, respectivament.
- Crea un objecte **PrimerCurs** amb aquest constructor, passant-li com a paràmetre el nom del fitxer ("grupsPrimerCurs/alumnes.txt"). Inspecciona'l i comprova que els alumnes se corresponen amb els del fitxer.

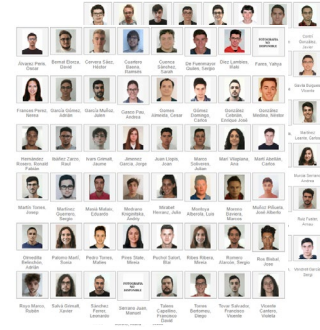


```
alumnes.txt
1  A
2  123456 9,7 Pau Gasol
3  234567 6,3 Sergio Rodriguez
4  345678 8,5 Pau Ribas
5  456789 10 Victor Luengo
6  567899 8,0 Fernando Romay
7  678999 10 Guillem Vives
8  B
9  987654 8,3 JuanMa Iturriaga
10 876543 10 Sergio Llull
11 765432 9,5 Felipe Reyes
12 654321 9,8 Rudy Fernandez
13 543211 6,7 Pablo Aguilar
14 C
15 111111 7,0 Victor Claver
16 222222 7,5 Nikola Mirotic
17 333333 8,6 Fernando San Emeterio
18 444444 10 Marc Gasol
19 555555 6,5 Sergi Ibaka
20 666666 7,5 Willy Hernangomez
21 777777 9,0 Juan Carlos Navarro
22
```

Exemple: grups de primer curs

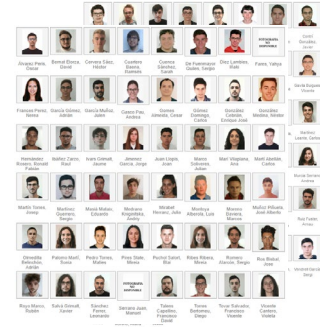
Blue:exercicisT7 [grupsPrimerCurs]

- En la classe **PrimerCurs** del paquet **grupsPrimerCurs**, el mètode **l·listarAlumnes** torna un llistat amb la informació de tots els alumnes d'un grup donat.



```
73 public String l·listarAlumnes(char g) {
74     int i = 0;
75     while (i < NOMGRUPS.length && NOMGRUPS[i] != g) { i++; }
76     String res = "";
77     if (i < NOMGRUPS.length) {
78         for (int j = 0; j < grups[i].length; j++) {
79             res += grups[i][j] + "\n";
80         }
81     }
82     else { res = g + "no és un grup vàlid"; }
83     return res;
84 }
```

Exemple: grups de primer curs



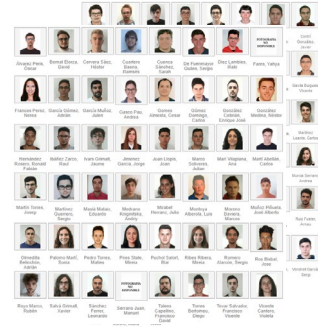
BlueI:exercicisT7 [grupsPrimerCurs]

- En la classe **PrimerCurs** del paquet **grupsPrimerCurs**, el mètode **notaMinima** torna el darrer alumne amb la nota mínima de tot primer.

```
88 public Alumne notaMinima() {
89     double notaMin = Double.MAX_VALUE;
90     Alumne aMin = null;
91     for (int i = 0; i < grups.length; i++) {
92         for (int j = 0; j < grups[i].length; j++) {
93             if (grups[i][j].getNota() <= notaMin) {
94                 notaMin = grups[i][j].getNota();
95                 aMin = grups[i][j];
96             }
97         }
98     }
99     return aMin;
100 }
```


Exercici: grups de primer curs

- En la classe **PrimerCurs** del paquet **grupsPrimerCurs**, completa els següents mètodes:
- mitjanaGrup**, que torna un array amb la nota mitjana de cada grup.
- matrHonorGrup**, que torna un llistat amb el primer alumne amb Matrícula d'Honor (nota 10) de cada grup.
- Crea un objecte **PrimerCurs** amb la informació del fitxer `alumnes.txt`, i prova els mètodes que has implementat.



BlueJ:exercisT7 [grupsPrimerCurs]

Projecte Edita Eines Vegeu Ajuda

Nova classe...

→

Compila

primerCu1: PrimerCurs

System.out.println(primerCu1.matrHonorGrup());

primerCu1 : PrimerCurs

BlueJ: BlueJ: Finestra de terminal - exercisT7-Sol

Opcions

```
Nom: Victor Luengo      Dni: 456789
Grup: A  Nota: 10.0  Assistència: sí

Nom: Sergio Llull       Dni: 876543
Grup: B  Nota: 10.0  Assistència: sí

Nom: Marc Gasol Dni: 444444
Grup: C  Nota: 10.0  Assistència: sí
```

BlueJ: Resultat del mètode

```
// Torna un array amb la nota mitjana de cada grup.
// @return double[], array resultat.
// Estratègia: recorregut iteratiu ascendent.
double[] mitjanaGrup()
```

primerCu1.mitjanaGrup() retornat:

double[]

Inspecciona

Obté

Tanca

: double[]

int length	3
[0]	8.75
[1]	8.8600000000000001
[2]	8.014285714285714

Inspect

Get

Show static fields

Close