Programació I - T4 - Estructures de Dades

Universitat de Barcelona
Grau en Enginyeria Informàtica

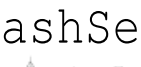
30 d'octubre de 2012

Introducció

- Hi ha problemes que requereixen tipus de dades més complexes que els existents com a tipus bàsics, per tant es necessiten definir *nous tipus de dades*: **classes** d'**objectes**
  - **Estructura de dades**: forma d'organitzar i codificar les dades.
  - **Mètodes per a manipular les dades**: conjunt d'operacions o utilitats que es poden realitzar sobre les dades emmagatzemades.
- Inicialment, es *simplifica* la **creació de nous tipus de dades** de la següent forma:
  - per a **dades homogènies**: s'usaran **taules** que permeten agrupar *N* elements del mateix tipus; s'implementaran com a objectes de la classe -ja existent- **`array`**.
  - per a **dades heterogènies**: s'usaran **tuples**; s'implementaran creant noves classes d'objectes amb un *atribut* per a cadascuna de les dades que han de contenir.

Introducció II

- És necessari un mecanisme per a referenciar de forma GLOBAL un conjunt de dades i que, alhora, permeti accedir de forma INDIVIDUAL cadascun dels elements del conjunt.
- En assignatures posteriors, per a seguir el paradigma de la **programació orientada a objectes**:
  - a les **classes noves d'objectes**: s'afegiran **mètodes** per a poder manipular les dades que contenen fàcilment per part de la resta del codi que les utilitzi. De fet, les dades dels objectes només es podran manipular a través d'aquests mètodes (**encapsulació**).
  - s'estudiaran **classes d'objectes disponibles**: per a disposar d'*estructures de dades habituals*, com ara `Collection`s (`ArrayList`, `TreeSet`, `LinkedList`...) i `Hash`s (`HashSet`, `LinkedHashSet`...).



UNIVERSITAT DE BARCELONA

Introducció a les taules

- **Concepte de taula**: estructura de dades per a **dades homogènies**, és a dir, per a contenir N elements del mateix tipus. Els elements emmagatzemats s'identifiquen a través d'un **índex de posició** dins la taula de valors.

0	1	2	3
H	o	l	a

- **`Array`**: classe d'objectes existent a la biblioteca bàsica del llenguatge Java que es poden utilitzar per a implementar el concepte de taula (existeixen més classes d'objectes que també es poden utilitzar).

Declaració i inicialització

- Malgrat `array` és de tipus referència, en ser una classe molt utilitzada, existeix una sintaxi pròpia diferent a la usada amb la resta d'objectes (és un cas semblant al d'`String`):
  - Declaració: `<tipus_elements> [] <id_variable>;`
    - `int [] a;` també s'accepta estil C `int a[];`
    - `String [] b;`
  - Inicialització: Un cop inicialitzat el nombre d'elements, el tamany de l'**array** és inalterable.

```
new <tipus_elements>[<tamany>]
  {<elem1>, ...  , <elem3>}
```

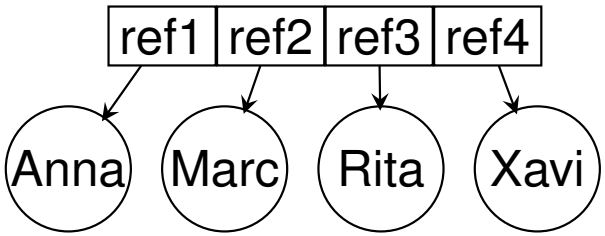
    - `int [] a = new int[10];` 10 elements
    - `int [] a = {1,2,3,s.length()};` 4 elements, índexs [0..3]
    - `String [] m ={''hola'',''adéu''.substring(0,2)};`
    - `float [] a;`
      `a = {1.1f,2.2f};` podem inicilitzar després de declarar.

Tipus emmagatzemats

- Els arrays poden contenir tipus valor o tipus referència:
  - tipus valor:

0	1	2	3
H	o	l	a

  - tipus referència:



- Exemple/exercici: què es crea a memòria amb el codi següent?
  - ```
    String [] m = new String[3];
    m[1] = ``hola'';
    m[0] = m[1];
    m[2] = m[1].substring(0,2);
    ```
    ```

Manipulació del contingut

- **Operacions bàsiques**:
  - **Assignació**: `<id_var>[<idx>] = <expressió>`
  - **Consulta**: `<id_var>[<idx>]` equival a una expressió
  - en molts llenguatges, els índexs comencen pel 0 i el límit superior es comprova automàticament.
- **Altres operacions**:
  - **Longitud**: `<id_var>.length` és longitud declarada
  - **Comparació**: `Array.equals(<array1>, <array2>)`
  - **Còpia**: `System.arraycopy(<src>, <src_ini>, <dst>, <dst_ini>, long)`
  - `fill`, `sort`, `binarySearch`...

Esquemes de programació de taules

- Podem **interpretar les taules com a seqüències d'elements**, i per tant podem adaptar els *esquemes de programació de seqüències*:
  - recorregut
  - cerca
- **Caracterització de la seqüència** dels índex d'una taula:
  - Primer() = 0
  - Següent(x) = x+1
  - QuedenElements(x) = (x $<$ #elem_taula)

Recorregut I

- Esquema de recorregut en una taula:

```
for(idx=0; idx<taula.length ;idx++){
    <sentències tractar taula[idx]>
}
```

```
import java.util.Scanner;
public class TaulaRecorregut {
    public static int TAMANY = 10;
    public static void main (String [] args) {
        int [] taula = new int[TAMANY];
        Scanner sc;
        sc = new Scanner(System.in);
        for (int idx=0; idx < TAMANY ; idx++) {
            System.out.println("Valor␣"+(idx+1)+"?␣");
            taula[idx] = sc.nextInt();
        }
        for (int idx=0; idx < taula.length ; idx++) {
            System.out.println("Element␣"+(idx+1)+"␣=␣"+taula[idx]);
        }
    }
}
```

Cerca

- Esquema de cerca per taules:

```
idx=0; trobat=false;
while ( (idx<taula.length) and !trobat ){
    if (<condició cerca sobre taula[idx]>){
        trobat=true;
    } else {
        idx++;
    }
}
```

Cerca I

```
import java.util.Scanner;
public class TaulaCerca {
  public static void main (String [] args) {
    int idx;  boolean trobat;
    Scanner sc =  new Scanner(System.in);
    String [] taula = {"Oriol", "Eulàlia", "David", "Santi"};
    System.out.println("Qui␣vols␣cercar?");
    String nom = sc.next();
    idx=0; trobat = false;
    while ( (idx < taula.length) && !trobat) {
      if (taula[idx].equals(nom)) {
        trobat = true;
      } else {
        idx++;
      }
    }
    if (trobat)  System.out.println("Trobat␣a␣la␣posició␣"+(idx+1));
    else         System.out.println("Nom␣no␣trobat");
  }
}
```

Exemple: I

Dígits continguts en els múltiples de 7: donats els múltiples de 7 inferiors a 10000, donar quants dígits 0's, quants dígits 1's, . . . , quants dígits 9 contenen.

- Identificació de la seqüència principal: múltiples de 7 inferiors a 10000
  - Primer element: m= 7
  - Següent element: m = m + 7
  - Final de seqüència: $m > 10000$
- Identificació de l'esquema: Recorregut

Exemple: I

```
public class ComptarDigitsSenseMetodes {
  public static int TAMANY = 10;
    public static void main (String [] args) {
        int [] comptador;
        int m, mult, d;
        comptador = new int[10];
        /* Identificacio de la sequencia:
           1er element: i = 0
           Seguent element:  i  = i + 1
           Final de sequencia: i >= 10
           Identificacio de l'esquema: Recorregut
        */
        for (int i=0; i<10; i++)
            comptador[i] = 0;

        m = 7;
        while (m<10000) {
            // tractarMultiple
            /* Identificacio de la sequencia:
               1er element: d = m % 0
               Seguent element: m = m / 10; d = m % 10;
               Final de sequencia: m == 0
               Identificacio de l'esquema: Recorregut
```

Exemple: II

```
        */
        System.out.println("Multiple␣" + m);
        mult = m;
        d = mult%10;
        while (mult!=0) {
            System.out.println("␣␣␣␣␣␣digit:␣"+d);
            comptador[d] = comptador[d]+1;
            mult = mult/10;
            d = mult%10;
        }

        // Seguent multiple
        m = m + 7;
    }

    // Imprimir multiples

    /* Identificacio de la sequencia:
       1er element: i = 0
       Seguent element:  i  = i + 1
       Final de sequencia: i >= 10
       Identificacio de l'esquema: Recorregut
    */
```

Exemple: III

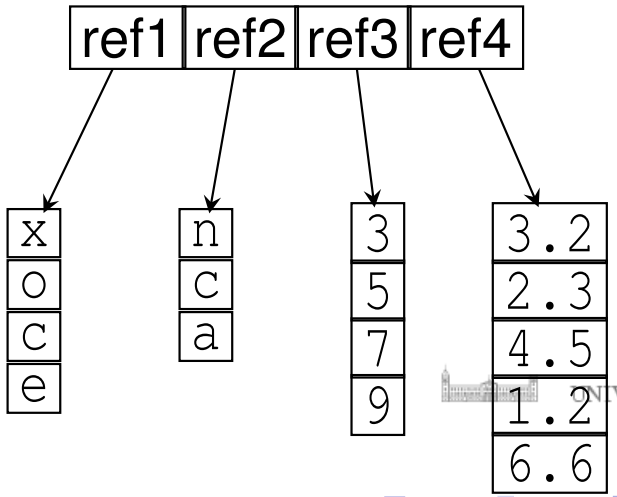
```
        for (int i=0; i<10; i=i+1) {
            System.out.println ("comptador␣"+i+":␣"+comptador[i]);
        }
    }
}
```

Introducció a les matrius

- **matriu** = taula N dimensional
- Hi ha llenguatges que les suporten directament (Java no)
- Conceptualment són homogènies

21	02	13	45
12	32	43	59
13	52	36	85
91	23	23	15

- Les podem implementar com una taula que conté altres taules (**aniuament**).
- Podrien ser heterogènies (de contingut i tamany).



Declaració i inicialització de matrius

- Declaració: `<tipus_elements> []<...>[] <id_variable>;`
  - `int [][] m;` declaració dues dimensions
- Inicialització:

  `new <tipus_elem>[<mida>]<...>[<mida_opcional>]`

  `{ {<el1>, ... , <elN>}, <...> , {<el1>, ... , <elN>} }`
  - `m = new int [3][3];` inicialització 3x3
  - `m = { {11, 21}, {12, 22} };` literals
  - `m = new int [3][];` només les "primeres" dimensions
    - `m[2] = new int [7];` posteriorment les "següents" dimensions
    - `m[2] = {3,6,4,1,8};` també com a literals

Esquemes de programació de matrius I

- Si les interpretem com a **taules aniuades**, podem adaptar els *esquemes de programació de taules*, fent **aniuament de bucles**.

```
public class MatriuRecorregut {
  public static void main (String [] args) {
    int [][] taula = { {1, 2}, {3, 4} };

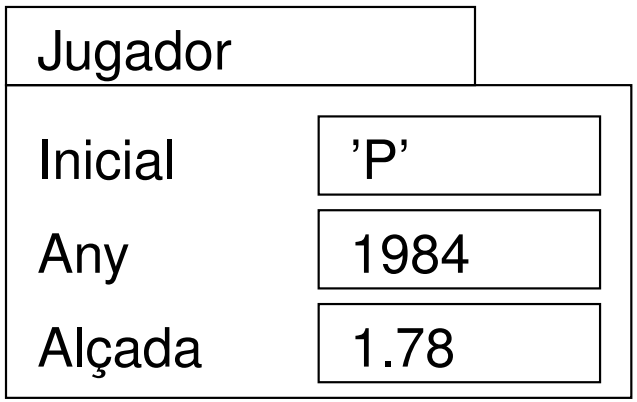
    for (int fil=0; fil < taula.length ; fil++) {

      for (int col=0; col < taula[fil].length ; col++) {

        System.out.println("Element␣("+(fil+1)+
                           ",␣"+(col+1)+
                           ")="+taula[fil][col]);
      }
    }
  }
}
```

Introducció a les tuples

- **Concepte de tupla**: estructura de dades per a **dades heterogènies**, és a dir, per a contenir N elements de diferents tipus. Els elements emmagatzemats s'identifiquen a través de l'**identificador de camp**.



- En Java: Inicialment, les implementarem creant noves **classes d'objectes** amb un *atribut públic* per a cadascuna de les dades que han de contenir. A mig termini, els **atributs** seran *privats* i crearem **mètodes d'objecte** per a poder manipular-los.

Declaració i inicialització

- **Declaració**: inicialment declararem classes amb atributs públics per a poder accedir des de fora d'elles. Posteriorment incorporarem mètodes per a manipular els atributs privats.
  - **Classe**:

```
class <id_classe> {
    public <id_tipus1> <id_atribut1>;
    <...>
    public <id_tipusN> <id_atributN>;
}
```

  - **Variable**:

```
<id_classe> <id_variable>;
```

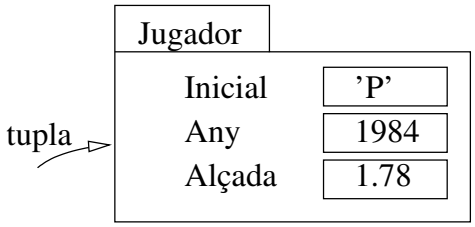
- **Inicialització**: Instanciació de la classe en un objecte amb uns valors propis.
  - **Objecte**: (de tipus referència)

```
<id_variable> = new <id_classe>();
```

Manipulació del contingut

- **Operacions bàsiques**:
  - amb *atributs públics*: inicialment
    - **Assignació**: `<id_var>.<id_atribut> = <expressió>`
    - **Consulta**: `<id_var>.<id_atribut>` equival a una expressió
  - amb *atributs privats*: a mig termini els atributs seran privats i la única forma de manipular-los serà cridant a mètodes d'objecte (**encapsulació**).
    - **Assignació**: `<id_var>.set<id_atribut>(<valor>)`
    - **Consulta**: `<id_var>.get<id_atribut>()` equival a una expressió
- **Altres operacions**:
  - **Comparació**: s'haurà de fer atribut a atribut o crear un mètode d'objecte.
  - **Còpia**: s'haurà de fer atribut a atribut o crear un mètode d'objecte.

Operacions bàsiques amb atributs públics I




```
import java.util.Scanner;

class Jugador {
  public char  inicial;
  public int   any;
  public float alcada;
}

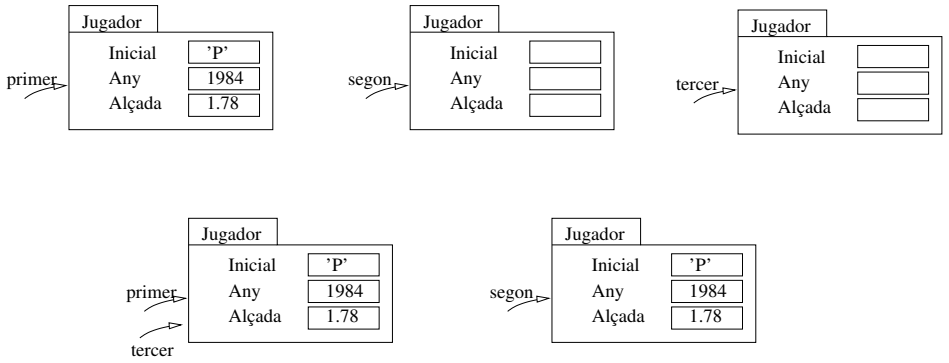
public class TuplaPublicsBasic {
  public static void main (String [] args) {
    Jugador tupla = new Jugador();
    String aux;
    Scanner sc;
    sc = new Scanner(System.in);
    System.out.println("Inicial?"); aux = sc.next();
    tupla.inicial = aux.charAt(0);
```

Operacions bàsiques amb atributs públics II

```
        System.out.println("Any?");
        tupla.any = sc.nextInt();
        System.out.println("Alçada?");
        tupla.alcada = sc.nextInt();

        System.out.println("Inicial=" + tupla.inicial);
        System.out.println("Any="     + tupla.any);
        System.out.println("Alçada="  + tupla.alcada);
    }
}
```

Altres operacions amb atributs públics I




```
class Jugador {
  public char  inicial;
  public int   any;
  public float alcada;
}
public class TuplaPublicsAltres{
  public static void main (String [] args) {
    Jugador primer, segon, tercer;
    primer=new Jugador(); segon=new Jugador(); tercer=new Jugador();

    primer.inicial = 'a';
    primer.any     = 1980;
    primer.alcada  = 181.4f;
```

Altres operacions amb atributs públics II

```
        segon.inicial  = primer.inicial;
        segon.any      = primer.any;
        segon.alcada   = primer.alcada;
        tercer = primer;

        System.out.println("segon␣vs.␣primer:" + (segon  == primer));
        System.out.println("tercer␣vs.primer:" + (tercer == primer));
    }
}
```

Problema: Cal guardar tot el conjunt? I

- Donada una seqüència de 10 polígons, on cada polígon té un caràcter identificador, el nombre de costats i el color donat en tres canals enters (R, G, B), calcular el nombre de polígons que són quadrats.

1. Identificació de la seqüència: Seqüència de 10 conjunts d'enters entrada per teclat:
   $i = 1;$,
   $i = i + 1;$
   `FinalSeq()=` $(i >= 10)$
2. Identificació de l'esquema: Recorregut. No fa falta guardar el polígon en lloc.

```
import java.util.Scanner;

class Poligon {
    public char inicial;
    public int ncostats;
}

public class Poligon1 {

    public static final int NUM_POLIGONS=10;
```

Problema: Cal guardar tot el conjunt? II

```
public static void main(String [] args) {
    Scanner sc;
    String aux;
    int color;
    int i;
    int numQuadrats;
    Poligon pol;

    numQuadrats = 0;
    i = 1;

    sc = new Scanner(System.in);
    pol = new Poligon();

    System.out.println("Costats?"); pol.ncostats = sc.nextInt();
    System.out.println("Inicial?"); aux = sc.next();
    pol.inicial = aux.charAt(0);
    System.out.println("R?"); color = sc.nextInt();
    System.out.println("G?"); color = sc.nextInt();
    System.out.println("B?"); color = sc.nextInt();

    while (i < NUM_POLIGONS) {
        if (pol.ncostats == 4) {
```

Problema: Cal guardar tot el conjunt? III

```
                numQuadrats = numQuadrats + 1;
            }
            // obtencio del seguent element

            System.out.println("Costats?"); pol.ncostats = sc.nextInt(
            System.out.println("Inicial?"); aux = sc.next();
            pol.inicial = aux.charAt(0);
            System.out.println("R?"); color = sc.nextInt();
            System.out.println("G?"); color = sc.nextInt();
            System.out.println("B?"); color = sc.nextInt();

            i = i + 1;
        }
        // tractament del darrer element
        if (pol.ncostats == 4) {
            numQuadrats = numQuadrats + 1;
        }
        System.out.println("El nombre de quadrats es " + numQuadrats);
    }
}
```

Problema: Cal guardar tot el conjunt? I

- Donada una seqüència de 10 polígons, on cada polígon té un caràcter identificador, el nombre de costats i el color donat en tres canals enters (R, G, B), calcular el nombre de polígons que són quadrats i al final llistar tota la informació d'aquells que són quadrats.

1. Identificació de la seqüència: Seqüència d'enters entrada per teclat:
   `ObtePoligon(p);` $i = 1;$,
   `ObtePoligon(p);` $i = i + 1;$
   `FinalSeq()=` $(i >= 10)$
2. Identificació de l'esquema: Recorregut. No fa falta guardar tots els polígons en lloc. Només cal guardar els quadrats.

```
import java.util.Scanner;

class Poligon {
    public int  ncostats;
    public char inicial;
    public int  R;
    public int  G;
    public int  B;
}
```

Problema: Cal guardar tot el conjunt? II

```
// Nomes guardara els quadrats
class TaulaPoligons {
    public int        numPoligons;
    public Poligon[] taula;
}
public class Poligon2 {

    public static final int NUM_POLIGONS=10;

    public static void main(String [] args) {

        Poligon       p;
        TaulaPoligons  tp;
        int           i;
        int           numQuadrats;
        String        aux;
        Scanner       sc;


        sc = new Scanner(System.in);

        tp = new TaulaPoligons();
        tp.taula = new Poligon[NUM_POLIGONS];
```

Problema: Cal guardar tot el conjunt? III

```
tp.numPoligons = 0;

numQuadrats = 0;

p = new Poligon();
System.out.println("Costats?"); p.ncostats = sc.nextInt();
System.out.println("Inicial?"); aux = sc.next();
p.inicial = aux.charAt(0);
System.out.println("R?");       p.R = sc.nextInt();
System.out.println("G?");       p.G = sc.nextInt();
System.out.println("B?");       p.B = sc.nextInt();

i = 1;
while (i < 2) {
    if (p.ncostats == 4) {
        tp.taula[tp.numPoligons] = new Poligon();
        tp.taula[tp.numPoligons].ncostats = p.ncostats;
        tp.taula[tp.numPoligons].inicial = p.inicial;
        tp.taula[tp.numPoligons].R = p.R;
        tp.taula[tp.numPoligons].G = p.G;
        tp.taula[tp.numPoligons].B = p.B;

        tp.numPoligons = tp.numPoligons + 1;
```

Problema: Cal guardar tot el conjunt? IV

```
            }
            System.out.println("Costats?"); p.ncostats = sc.nextInt();
            System.out.println("Inicial?"); aux = sc.next();
            p.inicial = aux.charAt(0);
            System.out.println("R?");   p.R = sc.nextInt();
            System.out.println("G?");   p.G = sc.nextInt();
            System.out.println("B?");   p.B = sc.nextInt();
            i = i + 1;

        }
        if (p.ncostats == 4) {
            tp.taula[tp.numPoligons] = new Poligon();
            tp.taula[tp.numPoligons].ncostats = p.ncostats;
            tp.taula[tp.numPoligons].inicial = p.inicial;
            tp.taula[tp.numPoligons].R = p.R;
            tp.taula[tp.numPoligons].G = p.G;
            tp.taula[tp.numPoligons].B = p.B;

            tp.numPoligons = tp.numPoligons + 1;
        }
        System.out.println("El nombre de quadrats es " + tp.numPoligons);
        for (i=0; i<tp.numPoligons; i = i+1) {
            System.out.println("Costats : " + tp.taula[i].ncostats);
```

Problema: Cal guardar tot el conjunt? V

```
                System.out.println("Inicial : " + tp.taula[i].inicial);
                System.out.println("R : " + tp.taula[i].R);
                System.out.println("G : " + tp.taula[i].G);
                System.out.println("B : " + tp.taula[i].B);
            }
        }
    }
```

Separabilitat de les classes: introducció a l'encapsulament I

- Cada classe ha d'anar dins d'un fitxer que s'anomena igual que la classe interna.
- S'ha de compilar el fitxer que conté el programa principal (main).
- La variable d'entorn CLASSPATH ha de contenir el directori on estan els diferents fitxers.

```
import java.util.Scanner;

public class MenuPoligons {

    public static final int NUM_POLIGONS=10;

    public static void main(String [] args) {

        Poligon      p;
        TaulaPoligons  tp;
        int          i;
        int          numQuadrats;
        String       aux;
        Scanner      sc;


        sc = new Scanner(System.in);

        tp = new TaulaPoligons();
        tp.taula = new Poligon[NUM_POLIGONS];
```

Separabilitat de les classes: introducció a l'encapsulament II

```
tp.numPoligons = 0;

numQuadrats = 0;

p = new Poligon();
System.out.println("Costats?"); p.ncostats = sc.nextInt();
System.out.println("Inicial?"); aux = sc.next();
p.inicial = aux.charAt(0);
System.out.println("R?");       p.R = sc.nextInt();
System.out.println("G?");       p.G = sc.nextInt();
System.out.println("B?");       p.B = sc.nextInt();

i = 1;
while (i < 2) {
    if (p.ncostats == 4) {
        tp.taula[tp.numPoligons] = new Poligon();
        tp.taula[tp.numPoligons].ncostats = p.ncostats;
        tp.taula[tp.numPoligons].inicial = p.inicial;
        tp.taula[tp.numPoligons].R = p.R;
        tp.taula[tp.numPoligons].G = p.G;
        tp.taula[tp.numPoligons].B = p.B;

        tp.numPoligons = tp.numPoligons + 1;
    }
    System.out.println("Costats?"); p.ncostats = sc.nextInt();
    System.out.println("Inicial?"); aux = sc.next();
    p.inicial = aux.charAt(0);
    System.out.println("R?");   p.R = sc.nextInt();
    System.out.println("G?");   p.G = sc.nextInt();
    System.out.println("B?");   p.B = sc.nextInt();
    i = i + 1;
```

Separabilitat de les classes: introducció a l'encapsulament III

```
        }
        if (p.ncostats == 4) {
            tp.taula[tp.numPoligons] = new Poligon();
            tp.taula[tp.numPoligons].ncostats = p.ncostats;
            tp.taula[tp.numPoligons].inicial = p.inicial;
            tp.taula[tp.numPoligons].R = p.R;
            tp.taula[tp.numPoligons].G = p.G;
            tp.taula[tp.numPoligons].B = p.B;

            tp.numPoligons = tp.numPoligons + 1;
        }
        System.out.println("El nombre de quadrats es " + tp.numPoligons);
        for (i=0; i<tp.numPoligons; i = i+1) {
            System.out.println("Costats : " + tp.taula[i].ncostats);
            System.out.println("Inicial : " + tp.taula[i].inicial);
            System.out.println("R : " + tp.taula[i].R);
            System.out.println("G : " + tp.taula[i].G);
            System.out.println("B : " + tp.taula[i].B);
        }
    }
}
```

- Classe TaulaPoligons: dins del fitxer TaulaPoligons.java

```
public class TaulaPoligons {
    public int      numPoligons;
    public Poligon[] taula;
}
```

- Classe Poligon: dins del fitxer Poligon.java

```
public class Poligon {
    public int  ncostats;
    public char inicial;
    public int  R;
    public int  G;
    public int  B;
}
```

Problemes de taules i tuples I

1. Feu un programa que demani dos llistats de nombres enters i compari si estrictament són iguals, és a dir, els mateixos nombres en les mateixes posicions. [sense utilitzar Arrays.equals]
2. Demanar a l'usuari quin nombre de jugadors té un equip, i a continuació l'alçada de cadascun d'ells. En acabar li retornem la mitjana, la moda i la desviació estàndard.
3. Feu un programa que demani dos llistats de nombres enters i compari si contenen el mateix conjunt de nombres, és a dir, tenen els mateixos nombres encara que sigui en posicions diferents.

Problemes de taules i tuples II

4. Dissenyeu un programa per a controla el vostre cartró en un bingo. Primer us demanarà tots els nombres que hi ha al vostre cartró, per files i columnes. En acabat, us anirà preguntant quina bola ha sortit. En el moment que li indiqueu, us dirà les seves coordenades (fila, columna) si està al cartró, o simplement que no la teniu. El programa acabarà quan indiqueu que ha sortit la bola -1.
5. Implementeu un programa que pregunti el llistat de noms dels jugadors d'un equip esportiu (*equip*). Després d'emmagatzemar-los, l'usuari entrarà la llista de suplents (*suplents*) i el nom del jugador titular darrere el qual s'ha d'inserir la llista de jugadors suplents. Al final, s'imprimirà per pantalla la

Problemes de taules i tuples III

llista *final* que inclourà tots els jugadors. [feu l'exercici amb 3 arrays, i després només amb 2]

6. Dissenyeu un programa que demani una llista de noms d'alumnes junt amb la seva alçada. Després l'usuari entrarà una alçada mínima i una alçada màxima i el programa llistarà tots els alumnes que estan dins del rang d'alçades.
7. Feu un programa que demani una fitxa bibliogràfica (autor, any de publicació i pes en Kg) a l'usuari i després la mostri per pantalla.
8. Dissenya un programa que demani les dades d'una pel·lícula (títol, any i durada) i després crei un objecte de tipus pel·lícula.
9. Crea un programa que demani la informació de dues bicicletes (model, pes, si té suspensió i preu) i després indiqui si són iguals i quina té major preu per kilogram.

Problemes de taules i tuples IV

10. Implementa un programa que permeti entrar les dades dels jugadors d'un equip (nom, alçada i pes) i a continuació indiqui quin és el nom del jugador més alt i quina és la mitjana de pes.
11. Codifica un programa que demani les dades acadèmiques d'un alumne (nom, edat i les notes de sis controls). En acabat les mostrarà per pantalla i indicarà quina és la seva nota mitjana.
12. Feu un programa que permeti emmagatzemar l'horari acadèmic dels cinc dies laborables, amb sessions d'una hora de 8 del matí a 2 del migdia. Per a cada sessió volem emmagatzemar quina matèria es dóna, quin professor/a la imparteix i si és de teoria o laboratori. En acabat el programa demanarà el nom d'un professor/a i mostrarà totes les sessions que realitza.