Programació I
Tema 5 - Disseny de composicions iteratives: Esquemes de recorregut i cerca

UNIVERSITAT DE BARCELONA

Grau en Enginyeria Informàtica
Facultat de Matemàtiques i Informàtica
Curs 18-19

5.1. Introducció al disseny d'estructures iteratives

- La construcció més complicada d'usar és la composició iterativa.
- Cal un mètode per dissenyar iteracions de manera que:
  - El disseny aconseguit sigui correcte.
  - Sigui un mètode universal i senzill.
- Passos a realitzar:
  1. Caracterització de la **seqüència** implicada en la iteració.
  2. Identificació de **l'esquema** de programació a aplicar: recorregut o cerca.
  3. Aplicació de l'esquema (segons un model d'esquema).
  4. Comprovació de casos extrems (tractament del darrer element, tractament de la seqüència buida).

5.2. Concepte de seqüència

- **Seqüència**: conjunt finit d'elements.
- La **caracterització d'una seqüència** es fa indicant:
  1. Identificació de la seqüència: és una descripció que indica quins elements formen part de la seqüència.
  2. Primer() $\rightarrow S_0$: primer element de la seqüència.
  3. Següent(S_i) $\rightarrow S_{i+1}$: regla per a calcular el següent element.
  4. FinalSeq(S_i) $\rightarrow$<booleà>: regla o propietat per saber si podem continuar avançant en la seqüència. Identifica el primer element que NO és de la seqüència.
     - Es coneixen el nombre d'elements de la seqüència.
     - El primer element que no és de la seqüència (o sentinella) es pot identificar.

Exemple 1

En aquest exemple, quina és la següència?

- **Caracterització d'una seqüència**:
  1. Identificació de la seqüència: és una descripció que indica quins elements formen part de la seqüència.
  2. Primer() $\rightarrow S_0$: primer element
  3. Següent(S_i) $\rightarrow S_{i+1}$: regla per a calcular el següent element
  4. FinalSeq(S_i) $\rightarrow$<booleà> Identifica el primer element que NO és de la seqüència.
- **Exemple 1: caracterització de la seqüência** $[1 \ldots 100]$**:**
  1. Identificació de la seqüència:
  2. Primer(): .....
  3. Següent(x): .....
  4. FinalSeq(x): .....

Exemple 1

En aquest exemple, quina és la següència?

- **Caracterització d'una seqüència**:
  1. Identificació de la seqüència: és una descripció que indica quins elements formen part de la seqüència.
  2. Primer() $\rightarrow S_0$: primer element
  3. Següent(S_i) $\rightarrow S_{i+1}$: regla per a calcular el següent element
  4. FinalSeq(S_i) $\rightarrow$<booleà> Identifica el primer element que NO és de la seqüència.
- **Exemple 1: caracterització de la seqüência** $[1...100]$**:**
  1. Identificació de la seqüència: enters des de l'1 al 100
  2. Primer(): 1
  3. Següent(x): $x+1$
  4. FinalSeq(x): $(x > 100)$

Exemple 2

En aquest exemple, quina és la següència?

- **Caracterització d'una seqüència**:
  1. Identificació de la seqüència: és una descripció que indica quins elements formen part de la seqüència.
  2. Primer() $\rightarrow S_0$: primer element
  3. Següent(S_i) $\rightarrow S_{i+1}$: regla per a calcular el següent element
  4. FinalSeq(S_i) $\rightarrow$<booleà> Identifica el primer element que NO és de la seqüència:
- **Exemple 2: caracterització de la seqüència** ['a','e','i','o','u']
  1. Identificació de la seqüència:
  2. Primer(): .....
  3. Següent(x): .....
  4. FinalSeq(x): .....

Exemple 2

En aquest exemple, quina és la següència?

- **Caracterització d'una seqüència**:
  - Identificació de la seqüència: és una descripció que indica quins elements formen part de la seqüència.
  - Primer() $\rightarrow S_0$: primer element
  - Següent(S_i) $\rightarrow S_{i+1}$: regla per a calcular el següent element
  - FinalSeq(S_i) $\rightarrow$<booleà> Identifica el primer element que NO és de la seqüència.
- **Exemple 2: caracterització de la seqüència ['a','e','i','o','u']**
  - Identificació de la seqüència: vocals
  - Primer(): 'a'
  - Següent(x): `switch(x){case 'a': seg = 'e'; break; case 'e': seg = 'i'; break;<...>}`,
  - FinalSeq(x): `!(x =='a' || x =='e' || x =='i' || x =='o' || x=='u' )`

Exemple 3

En aquest exemple, quina és la següència?

- **Caracterització d'una seqüència**:
  1. Identificació de la seqüència: és una descripció que indica quins elements formen part de la seqüència.
  2. Primer() $\rightarrow S_0$: primer element
  3. Següent(S_i) $\rightarrow S_{i+1}$: regla per a calcular el següent element
  4. FinalSeq(S_i) $\rightarrow$<booleà> Identifica el primer element que NO és de la seqüència.
- **Exemple 3: usuari entra comandes fins que indica "*sortir*"**:
  1. Identificació de la seqüència:
  2. Primer(): .....
  3. Següent(x): .....
  4. FinalSeq(x): .....

Exemple 3

En aquest exemple, quina és la següència?

- **Caracterització d'una seqüència**:
  1. Identificació de la seqüència: és una descripció que indica quins elements formen part de la seqüència.
  2. Primer() $\rightarrow S_0$: primer element
  3. Següent(S_i) $\rightarrow S_{i+1}$: regla per a calcular el següent element
  4. FinalSeq(S_i) $\rightarrow$<booleà> Identifica el primer element que NO és de la seqüència.
- **Exemple 3: usuari entra comandes fins que indica "*sortir*"**:
  1. Identificació de la seqüència: comandes entrats per teclat
  2. Primer(): `sc.nextLine();`
  3. Següent(x): `sc.nextLine();`
  4. FinalSeq(x): `x.equals("sortir")`

Exemples: L'usuari entra comandes fins que indica "*sortir*" I

```
import java.util.Scanner;
public class Sortir {

  public static void main (String [] args) {
    String cmd;
    Scanner sc;
    /* Identificació de la seqüència: Seqüència de
       Strings/Comandes entrats pel teclat.
       Primer()=  sc.nextLine();
       Següent(x)= sc.nextLine();
       FinalSeq(x)= (x.equals("sortir"))
    */
    sc = new Scanner(System.in);
    System.out.println("Comanda?");
    cmd = sc.nextLine();
    while (!cmd.equals("sortir")) {
      System.out.println("Processant comanda "+cmd);
      System.out.println("Comanda?");
      cmd = sc.nextLine();
    }
  }
}
```

5.3. Esquemes de programació de seqüències

- **Esquema de programació**: patró de solució que es pot aplicar per a resoldre un ventall de problemes concrets.
- **Esquema de programació de seqüències**:
  - de **recorregut**: per a recórrer tots els elements de la seqüència.
  - de **cerca**: per a recórrer la seqüència fins a trobar un determinat element o fins a que es compleixi una certa condició booleana.

5.3.1. Esquema de Recorregut

- **Objectiu:** Donada una seqüència d'elements es vol accedir a TOTS ells per a aplicar-los un cert tractament.
- **Esquema:**

```
<inicialitzacions>
elem=Primer();
while ( !FinalSeq(elem) ) {
    <tractar_element>
    elem=Següent(elem);
}
<finalitzacions>
```

Esquema de programació: Recorregut

L'esquema de recorregut garanteix que:

- Cada element de la seqüència només es tracta una vegada.
- Sempre es tracta l'element actual primer i després s'avança al següent element de la seqüència.
- No es tracta cap element que no sigui de la seqüència.
- A cada iteració decreix la part dreta de la seqüència.
- En acabar, tots els elements de la seqüència s'han tractat (la part dreta de la seqüència és buida).

Exemple de resolució I

- Exemple: Calcular la mitjana dels quadrats dels 100 primers naturals.
  1. **Identificació de la seqüència**: Seqüència d'enters [1..100]. Primer()=1, Següent(x)=$x+1$, FinalSeq(x)=$(x > 100)$
  2. **Identificació de l'esquema**: Recorregut

```
public class MitjanaQ {
  public static void main (String[] args) {
      int   x;
      int   suma;
      float mitjana;
      suma = 0;                              // Inicialitzacions
      x = 1;                                 // Primer Element
      while (x <= 100) {                     // while (!FinalSeq)
          suma = suma + x*x;                 // Tractament
          x = x + 1;                         // Seguent Element
      }
      mitjana = (float)(suma) / 100.0f; // Finalitzacions
      System.out.println ("La mitjana és " + mitjana);
  }
}
```

Recorreguts de seqüències amb el nombre d'elements conegut

- Els problemes que es poden reduir a seqüències tals que es controla el seu final amb el nombre d'elements i que, a més a més, s'ha d'aplicar l'esquema de recorregut, es poden solucionar amb la instrucció especialitzada `for`.
- La variable que controla el nombre d'elements de la seqüència s'anomenada **comptador**.

Recorreguts de seqüències amb nombre d'elements conegut

- Coneixem el nombre d'elements. Doncs, l'aplicació directa de l'esquema de recorregut és:

```
 <inicialitzacions>
idx=<inici>;
while (idx <= <final>) {
    <tractar_element>
    idx=Incrementa(idx);
}
 <finalitzacions>
```

```
public class IterativaMentreComptador {
  public static void main (String[] args) {
    int num;
    num = 0;
    while (num <= 10) {
      System.out.println("5 x " + num + " = " + (5 * num));
      num = num + 1;
    }
  }
}
```

Per/for I

- Quan el nombre d'elements és conegut i s'aplica un recorregut, és aconsellable utilitzar l'estructura algorísmica `for`:

```
 <inicialitzacions>
for(idx=<inici>; idx<=<final> ;idx=Incrementa(idx)){
    <tractar_element>
}
 <finalitzacions>
```

- Si cal fer el bucle en ordre decreixent: la condició seria l'oposada i en lloc d'incrementar, decrementariem.
- En Java, podem declarar la variable comptador a interior del bucle. Aquesta variable només es pot utilitzar dins del `for`. A fora del `for` no es coneix aquesta variable.

Per/for II

```
public class IterativaPer {

  public static void main (String[] args) {

    for (int num = 0; num <= 10; num++) {
      System.out.println("5 x " + num + " = " + (5 * num));
    }

  }
}
```

5.3.2. Esquema de cerca

- **Objectiu:** Donada una seqüència d'elements es vol accedir a un element que verifica o compleix una determinada propietat (o condició booleana).
- **Esquema 1:**

```
<inicialitzacions>
condicio_cerca = false
elem=Primer();
while ( !FinalSeq(elem) && !condició_cerca ) {
    elem=Següent(elem);
}
<condicio_cerca = !FinalSeq(elem)>
<finalitzacions>
```

5.3.2. Esquema de Cerca

- L'esquema de cerca garanteix que:
  - Cada element de la seqüència només es tracta una vegada.
  - Sempre es tracta l'element actual primer i després s'avança al següent element de la seqüència.
  - No es tracta cap element que no sigui de la seqüència.
  - A cada iteració, tot element de la part de l'esquerra de la seqüència no cumpleix la propietat de cerca.
  - En acabar, s'ha comprovat que la condició de cerca no es cumpleix per a tots els elements de la seqüència o bé s'ha trobat el primer element de la seqüència que cumpleix la propietat de cerca.
  - S'ha de garantir que la propietat de cerca es pot avaluar sobre el sentinella de la seqüència.

Exemple de resolució I

En aquest problema, quina és la seqüència i quin és l'esquema

- Donada una seqüència d'enters introduïts pel teclat acabada en 0, esbrinar si algun d'aquests enters és la *CLAU_SECRETA* d'accés a un compte.
  1. Caracterització de la seqüència
     - Identificació de la seqüència: .....
     - Primer(): .....
     - Següent(x): .....
     - FinalSeq(x): .....
  2. Identificació de l'esquema (recorregut o cerca, en cas de cerca, indicar condició de cerca):

     .....

Exemple de resolució (Esquema de cerca 1) I

```
import java.util.Scanner;
public class Cerca {
  public static final int CLAU_SECRETA=725;

  public static void main (String[] args) {
    int clau;     // clau entrada per l'usuari
    Scanner sc;
    /*
       Identificacio de la sequencia: sequencia d'enters (claus)
                                          entrats per teclat,
                                          acabada en 0.
       Primer:       clau = sc.nextInt();
       Seguent():    clau = sc.nextInt();
       FinalSeq():   clau == 0
       Esquema:      Cerca. Condicio: clau == CLAU_SECRETA
    */

    sc = new Scanner(System.in);

    System.out.println("Clau␣secreta?");
    clau = sc.nextInt();
```

Exemple de resolució (Esquema de cerca 1) II

```
    while ( (clau != 0) && (clau!=CLAU_SECRETA)) {
      System.out.println("Clau␣secreta?");
      clau = sc.nextInt();
    }
    if ( clau!=0 ) {
      System.out.println("Acces␣obert");
    }
    else {
      System.out.println("No␣ho␣has␣encertat!");
    }
  }
}
```

Esquema de cerca

- Un esquema alternatiu que permet no avaluar el sentinella de la seqüència
- **Esquema 2:**

```
<inicialitzacions>
elem = Primer();
trobat = false;
while ( !FinalSeq(elem) && !trobat) {
    if (condició_cerca) {
        trobat = true;
    } else {
        elem=Següent(elem);
    }
}
<finalitzacions>
```

Exemple de resolució (Esquema de cerca 2) I

- Exemple: Donada una seqüència d'enters pel teclat acabada en 0, esbrinar si algun d'aquests enters és la *CLAU_SECRETA* d'accés a un compte.
  1. **Identificació de la seqüència**: Seqüència d'enters entrada per teclat:
     `Primer()=sc.nextInt();`,
     `Següent(clau)=sc.nextInt();`,
     `FinalSeq(clau)`=$(clau == 0)$
  2. **Identificació de l'esquema**: cerca : condició de cerca :
     $(clau == CLAU\_SECRETA)$

```
import java.util.Scanner;
public class Cerca2 {
  public static final int CLAU_SECRETA=725;
  public static void main (String[] args) {
    int clau;        // clau entrada per l'usuari
    boolean atura;   // indica si s'ha encertat la clau

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

Exemple de resolució (Esquema de cerca 2) II

```
    System.out.println("Clau␣secreta?");
    clau = sc.nextInt();
    atura = false;
    while ( (clau != 0) && !atura) {
      if ( clau == CLAU_SECRETA ) {
        atura = true;
      } else {
          System.out.println("Clau␣secreta?");
          clau = sc.nextInt();
      }
    }



    if ( atura ) {
        System.out.println("Acces␣obert");
    } else {
        System.out.println("No␣ho␣has␣encertat!");
    }
  }
}
```

Bucles aniuats I

- Podem incloure una sentència iterativa dins del bloc de sentències d'una altra sentència iterativa.
- En aquests casos s'analitzen les composicions iteratives per separat (es poden tenir recorreguts dins de recorreguts, cerques dins de recorreguts, etc.)
- Es realitza el mateix procés d'identificar la seqüència i identificar l'esquema en cada bucle implicat en la solució.
- Exemple: Llistar per pantalla les taules de multiplicar de 1 al 10.

Bucles aniuats I

```
/* Llistat de les taules de multiplicar del 1 al 10 */
public class IterativaPerAniuat {
  public static void main (String[] args) {
      /* Identificació de la sequencia: enters 1 al 10
         Primer: 1
         Seguent(base): base + 1
         FinalSeq(base): base > 10
         Identificacio de l'esquema: Recorregut
      */
      for (int base = 1; (base <= 10) ; base++) {
          System.out.println("Taula del " + base);
          /* Identificacio de la sequencia: enters del 0 al 10
             Primer: 0
             Seguent(num):  num + 1
             FinalSeq(num): num > 10
             Identificació de l'esquema: Recorregut */
          for (int num = 0; (num <= 10) ; num++) {
              System.out.println(base + " x " + num + " = " + (base *
                  num));
          }
      }
  }
}
```

Bucles aniuats II

Problemes I

1. Dissenyar un programa que donat un enter pel teclat, *n*, compti el nombre de dígits que el composen.
2. Donada una seqüència d'enters positius entrats per teclat acabada en -1, calcular la seva mitjana.
3. Donada una seqüència d'enters positius entrats per teclat acabada en -1, calcular la mitjana dels nombres de la seqüència que són entre 1 i 100. Suposem, llavors, que en aquesta seqüència pot haver-hi enters fora de l'interval esmentat.
4. Feu un programa que mostri per pantalla els múltiples de 13 positius inferiors a 1000.
5. Feu un programa que mostri per pantalla el 1000 primers múltiples de 13 positius.
6. Donada una seqüència d'enters pel teclat acabada en 0, feu un programa que esbrini si tots són negatius.
7. Donada una seqüència d'enters positius pel teclat acabada en -1, feu un programa que trobi el màxim. Com modificaries aquest programa per a que trobi el mínim també?
8. Creeu un programa que llegeixi un nombre enter entrat per l'usuari i el descompongui en els seus factors primers.
9. Donada la seqüència dels múltiples de 7 menors de 10000, comptar el nombre de vegades que surt el dígit 2.

ComptaDigits I

1. Dissenyar un programa que donat un enter pel teclat, *n*, compti el nombre de dígits que el composen.

```
import java.util.Scanner;

public class ComptaDigits {

  public static void main(String[] args) {

  int n;     // variable entera entrada per l'usuari, de la qual
             // es vol comptar el nombre de digits que te
  int d;     // variable auxiliar de control de la sequencia
  int suma; // comptador de digits
  Scanner sc;

  sc = new Scanner(System.in);

  System.out.println("A quin enter li vols comptar els digits");
  n = sc.nextInt();
```

ComptaDigits II

```
  /* Identif. de la seq: enters sucessius obtinguts
                         en extreure digits.
     Primer: d = n
     Seguent(d): d = d/10
     FinalSeq(d): d == 0
     Identificació de l'esquema: Recorregut
   */

    suma = 0;
    d = n;
    while (d != 0) {
      suma = suma + 1;
      d = d / 10;
    }
    System.out.println("El nombre de digits de "+n+" es "+suma);
  }
}
```

Mitjana versió 0 (sense comprovació del rang dels enters entrats) I

1. Donada una seqüència d'enters entrats per teclat acabada en -1, calcular la seva mitjana.

```
import java.util.Scanner;
public class Mitjanav0 {
  public static void main (String[] args) {
      int    x;          // x: enter llegit. Si es -1 acaba el programa
      int    comptador; // comptador: nombre d'enters llegits
      int    suma;       // suma: sumes parcials acumulades
      float mitjana;    // mitjana: mitjana dels enters entrats
      Scanner sc;
      /*
         Identif. de la seq: sequencia d'enters entrats
                                          per teclat acabada en -1
           Primer:       x = sc.nextInt();
           Seguent():    x = sc.nextInt();
           FinalSeq(x):   x == -1
           Identificació de l'esquema: Recorregut
      */
      sc = new Scanner(System.in);
```

Mitjana versió 0 (sense comprovació del rang dels enters entrats) II

```
        suma = 0;           // Inicialitzacions
        comptador = 0;
        System.out.println("Entra␣enters␣positius␣(o␣-1␣per␣acabar)");

        x = sc.nextInt(); // Primer element

        /* Tractament de la seqüencia buida */
        if ( x == -1) {
            System.out.println("La␣sequencia␣es␣buida");
        } else {
            while ( x != -1 ) {
                suma = suma + x;              // Tractament
                comptador = comptador + 1;
                x = sc.nextInt();             // Seguent element
            }
            mitjana = (float)(suma) / (float)(comptador);
            System.out.println ("La␣mitjana␣es␣" + mitjana);
        }
    }
}
```

Mitjana versió 1 (amb comprovació del rang dels enters entrats) I

1. Donada una seqüència d'enters acabada en -1, calcular la mitjana dels nombres de la seqüència que són entre 1 i 100.
2. Pot haver-hi enters fora de l'interval, llavors, fa falta una identificació addicional per a obtenir els enters correctes

```
import java.util.Scanner;
public class Mitjana {
  public static void main (String[] args) {
    int    x;  // x: enter llegit. Si es -1 acaba el programa
    int    comptador; // comptador: nombre d'enters llegits
    int    suma;      // suma: sumes parcials acumulades
    float  mitjana;   // mitjana: mitjana dels enters entrats
    Scanner sc;
    /* Identificació de la seq: sequencia d'enters
                                entrats per teclat,
                                acabada en -1.
       Primer:        x = llegirEnter entre 1 i 100;
       Seguent():     x = llegirEnter entre 1 i 100;
       FinalSeq(x):   x == -1
```

Mitjana versió 1 (amb comprovació del rang dels enters entrats) II

```
  Identificació de l'esquema de tractament: Recorregut
*/
sc = new Scanner(System.in);
suma = 0;
comptador = 0;
System.out.println("Entra enters entre 1 i 100 (o -1 per acabar)")
    ;

// Obtencio del Primer element
/* Identif. de la seqüencia: Enters entrats fora de l'interval
   Primer:        x = sc.nextInt();
   Seguent(x):    x = sc.nextInt();
   FinalSeq(x):   x == -1
   Identif. de l'esquema: Cerca. Condicio cerca: 1 <= x && x <=
       100
*/

 x = sc.nextInt();
 while (x != -1 && !(1 <= x && x <= 100)) {
     x = sc.nextInt();
 }
```

Mitjana versió 1 (amb comprovació del rang dels enters entrats) III

```
        /* Tractament de sequencia buida */
        if ( x == -1) {
             System.out.println("La␣sequencia␣es␣buida");
        }
        else {
           while ( x != -1 ) {
             // Tractament
             suma = suma + x;
             comptador = comptador + 1;

             // Obtencio del seguent element
             x = sc.nextInt();
             while (x != -1 && !(1 <= x && x <= 100)) {
               x = sc.nextInt();
             }

         }
         mitjana = (float)(suma) / (float)(comptador);
         System.out.println ("La␣mitjana␣es␣" + mitjana);
    }
}
```

Mitjana versió 1 (amb comprovació del rang dels enters entrats) IV

```
}
```