Elements de Programació
Tema3 - Disseny de composicions iteratives: Esquemes de recorregut i cerca

Universitat de Barcelona
Enginyeria en Informàtica de Sistemes

9 d'octubre de 2012

UNIVERSITAT DE BARCELONA

Disseny de composicions 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. Identificació de la seqüència implicada en la iteració
  2. Identificació de l'esquema de programació a aplicar (recorregut o cerca)
  3. Aplicació de l'esquema
  4. Comprovació de casos extrems (tractament del darrer element, tractament de la seqüència buida)

Identificació de la seqüència

- **Seqüència**: conjunt finit d'elements. Es pot caracteritzar per:
  - 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>: 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

Exemples:

- [1..100]: Primer()=1, Següent(x)=$x+1$, FinalSeq(x)=$(x > 100)$
- ['a','e','i','o','u']: Primer()='a',
  Següent(x)= `switch(x){case 'a':seg='e';break;<...>}`,
  FinalSeq(x)= `! (x=='a'||<...>||x=='u')`
- usuari entra comandes fins que indica "*sortir*":
  Primer()= `sc.nextLine();`
  Següent(x)= `sc.nextLine();`
  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;
    /* Caracterització de la seqüència: Seqüència de Strings
                                          entrats pel teclat
       Primer()=  sc.nextLine();
       Següent(x)= sc.nextLine();
       Final de seqüència (x)= (x.equals("sortir"))
    */
    Scanner sc;
    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();
    }
  }
}
```

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.

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

- 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));
    }

  }
}
```

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

Esquema de programació: 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 elements 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

- 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: Quan s'entra la clau secreta s'atura el while: esquema de cerca: condició de cerca :
     $(clau == CLAU\_SECRETA)$

Exemple de resolució II

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

        System.out.println("Clau secreta?");
        clau = sc.nextInt();
        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 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: Quan s'entra la clau secreta es para:
     esquema de cerca: condició de cerca :
     $(clau == CLAU\_SECRETA)$

Exemple de resolució II

```
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);

    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();
      }
    }
```

Exemple de resolució III

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

Esquema de cerca amb tractament

- Un esquema de cerca on s'ha de realitzar un tractament als elements de la seqüència, a mesura que es comprova la condició de cerca.
- **Esquema 3:**

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

Exemple de resolució 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 i comptar el nombre d'intents que s'han realitzat.
  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: Quan s'entra la clau secreta es para:
     esquema de cerca: condició de cerca :
     $(clau == CLAU\_SECRETA)$

Exemple de resolució II

```
import java.util.Scanner;
public class Cerca3 {

  public static final int CLAU_SECRETA=725;

  public static void main (String [] args) {
    int     clau;     // clau entrada per l'usuari
    int     intents;  // nombre d'intens de l'usuari
    boolean atura;    // indica si s'ha encertat la clau
    Scanner sc;
    sc = new Scanner(System.in);

    intents = 0;
    System.out.println("Clau secreta?");
    clau = sc.nextInt();
    atura = false;
    while ( (clau != 0) && !atura) {
        intents = intents + 1;
        if ( clau == CLAU_SECRETA ) {
            atura = true;
        } else {
          System.out.println("Clau secreta?");
          clau = sc.nextInt();
```

Exemple de resolució III

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

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) {
      /* Sequencia: Primer()=1; Següent(base)= base+1,
                    FinalSeq(base)= (base>10)
         Esquema: Recorregut
      */
      for (int base = 1; (base <= 10) ; base++) {
          System.out.println("Taula del "+base);

          /* Sequencia: Primer()=0; Següent(num)= num+1,
             FinalSeq(num)= (num>10)
             Esquema: Recorregut
          */
          for (int num = 0; (num <= 10) ; num++) {
              System.out.println(base+" x "+num+" = "+(base*num));
          }
      }
  }
}
```

Problemes I

1. Dissenyar un programar que donat un enter pel teclat, *n*, compti el nombre de dígits que el composen.
2. Feu un programa que mostri per pantalla els múltiples de 13 positius inferiors a 100.
3. Feu un programa que mostri per pantalla el 100 primers múltiples de 13 positius.
4. Donada una seqüència d'enters pel teclat acabada en 0, feu un programa que esbrini si tots són positius.
5. Donada una seqüència d'enters positius pel teclat acabada en -1, feu un programa que trobi el màxim.
6. Dissenyeu un programa que demani tres nombres a l'usuari, i tot seguit mostri la llista dels múltiples del 3r nombre entrat que hi ha en el rang comprés entre el 1r i el 2n nombre que ha donat l'usuari.
7. Creeu un programa que llegeixi un nombre enter entrat per l'usuari i el descompongui en els seus factors primers.
8. 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 programar 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
                     comptar el nombre de digits que te */
        int d;    // variable auxiliar de control de la seqüència
        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

```
        /* Identificació de la seqüència principal: enters sucessius o
             1er element: d = n
             Seguent element: d = d/10
             Fi de seqüència: d == 0
           Identificació de l'esquema de tractament: 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 progr
      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 seqüència principal: sequencia d'enters e
                                                  per teclat acabada e
           1er element:       x = sc.nextInt();
           Seguent element:   x = sc.nextInt();
           Fi de seqüència:   x == -1
         Identificació de l'esquema de tractament: Recorregut
      */
```

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

```
        sc = new Scanner(System.in);
        suma = 0;           // Inicialitzacions
        comptador = 0;
        System.out.println("Entra␣enters␣entre␣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, cadascun d'ells entre 1 i 100, entrats per teclat acabada en -1, calcular la seva mitjana.
2. 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 progr
      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 seqüència principal: sequencia d'enters e
                                                  per teclat acabada en -1
          1er element:      x = llegirEnter entre 1 i 100;
          Seguent element:  x = llegirEnter entre 1 i 100;
```

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

```
    Fi de seqüència:  x == -1
   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
 /* Identificacio de la seqüencia: Enters entrats fora de l'inte
      1er element:      x = sc.nextInt();
      Seguent element:  x = sc.nextInt();
      Fi de seqüència:  x == -1
    Identificació de l'esquema de tractament: Cerca: 1<=x && x<1
  */

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);
        }
    }
}
```