Sessió 2 – Pràctica 2

Consideracions
Aclariments

Objectius

- **Implementar una estructura de dades lineal: La cua.**
  - **Genèrica usant la notació <E>**
  - **Encadenada vol dir que la gestió de l'espai es fa utilitzant "nodes" que es poden encadenar entre ells. Usar la tècnica enllaçada**

La interfície Cua<E>

Defineix les operacions típiques de les cues:

- **encuar** un objecte al final de la cua
- **desencuar** el primer objecte (cap) de la cua
- **cuaBuida** preguntar si és buida
- **consultaCua** obtenir el cap sense desencuar
- **buidar** deixa la cua com si s'acabés de crear

La interfície Cua<E>

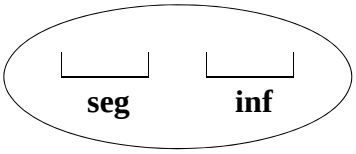
Els mètodes **desencuar** i **consultaCua** han de llençar una excepció en el cas de ser:

La cua buida

La classe Node<E>

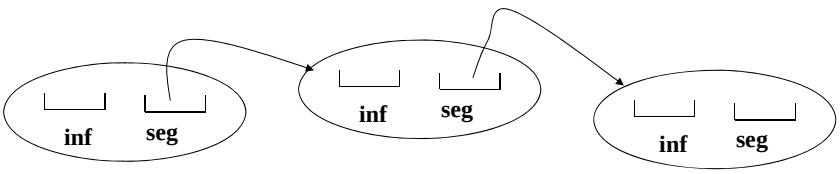
Els objectes de la classe Node tenen dos atributs:

- **inf** referencia un objecte tipus E qualsevol
- **seg** referencia un objecte de la classe Node



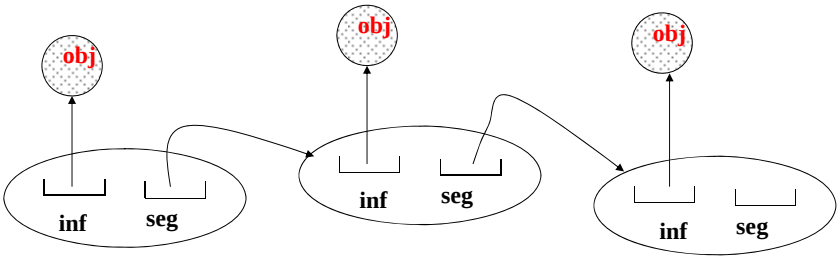
La classe Node<E>

Els objectes de la classe Node es poden enllaçar entre ells per formar una seqüència enllaçada.



La classe Node<E>

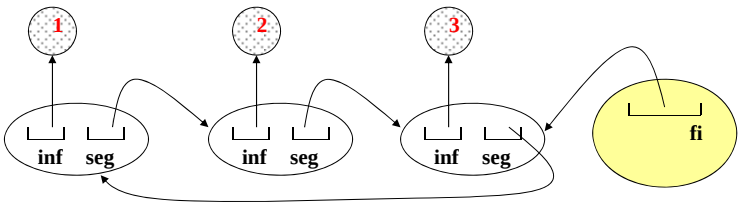
Cada Node pot referenciar un objecte



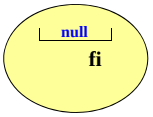
La classe CuaEnll<E>

- Implementa totes les operacions definides per la interfície Cua<E>, i algunes més.
- Utilitza una representació basada en una seqüència enllaçada **sense node capçalera, simple i circular.**
- La seqüència està formada per objectes de la classe Node.
- Es manté una única referència al darrer element de la cua i aquest té una referència (seg) al primer element.

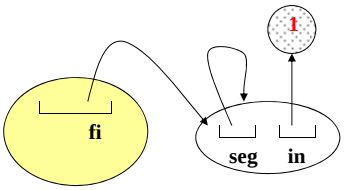
La classe CuaEnll<E>: representació



La classe CuaEnll<E>: representació

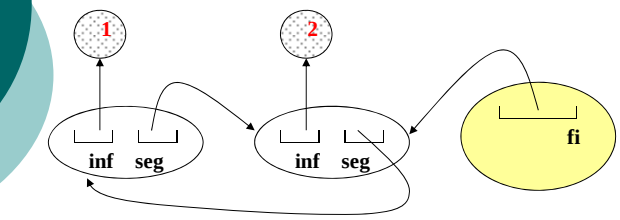


Una cua buida

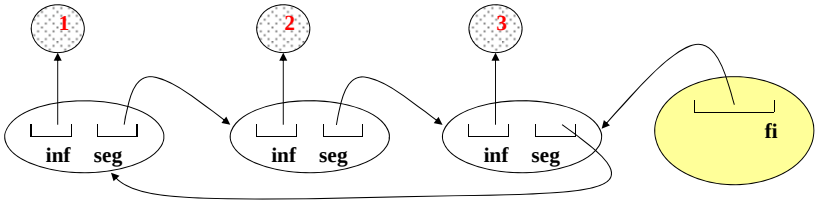


Una cua amb un sol element

La classe Cua CuaEnll<E> : representació



Una cua amb dos elements

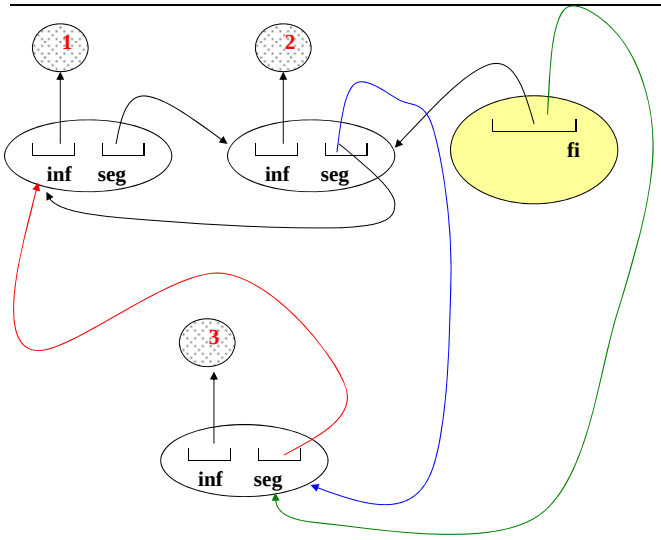


Una cua amb tres elements

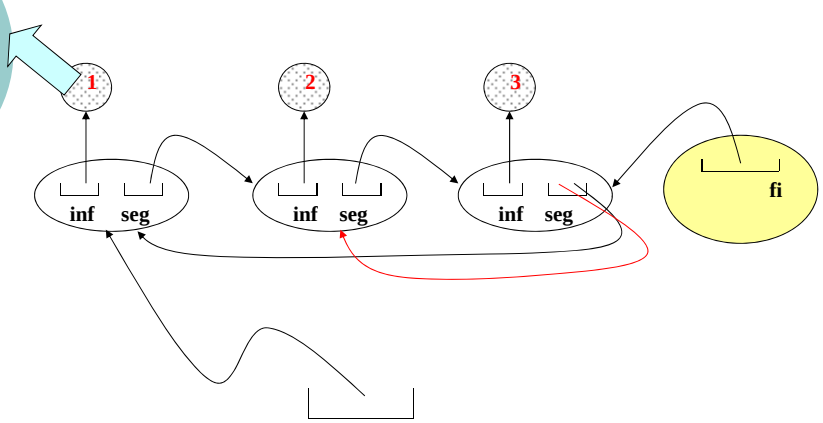
La classe Cua CuaEnll<E>: Atributs

- L'enunciat de la pràctica indica que només cal usar un atribut per referenciar la seqüència enllaçada.
- La classe `Node<E>` cal que sigui **privada**, declarada dins del cos de la classe CuaEnll.

La classe Cua CuaEnll<E> : encuar

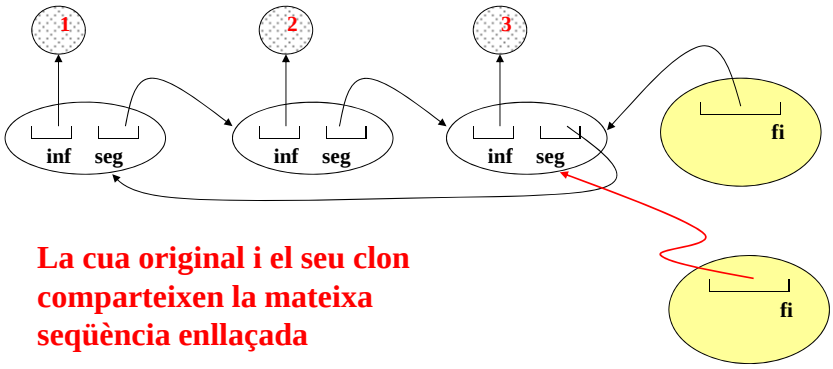


La classe Cua CuaEnll<E> : desencuar



La classe Cua CuaEnll<E> : clonació

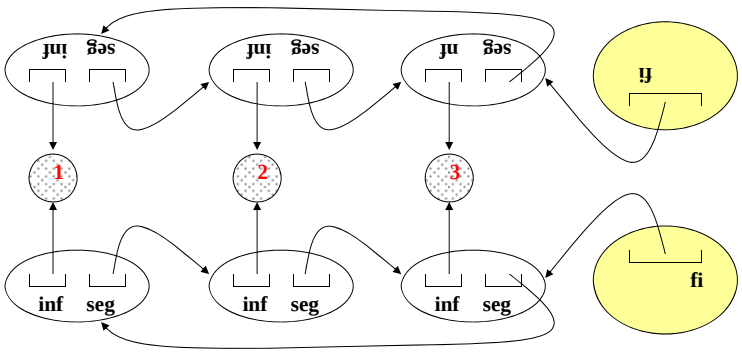
- Si ens limitem a invocar (super).clone i a convertir el resultat en una Cua passa quelcom com això:



La cua original i el seu clon comparteixen la mateixa seqüència enllaçada

La classe Cua CuaEnll<E> : clonació

- El mètode clone ha d'aconseguir que la cua clònica tingui la seva pròpia seqüència enllaçada:



Redefinició mètode clone()

- Ha de seguir els següents passos:
  - La classe ha d'indicar que implementa la interfície Cloneable
  - La classe ha de redefinir el mètode de clone() de la classe Object.
  - Sintaxi:

```
public Object clone () throws
            CloneNotSupportedException
```

Redefinició mètode clone()

```
public Object clone () throws
            CloneNotSupportedException{
      try{
            ??? = (????)super.clone();
            // duplicació en profunditat
      }
      catch (CloneNotSupportedException e){
            //creació i llançament
            // exception de la classe
            // indicada
      }
      // retorn objecte clonat
}
```

La classe Cua CuaEnll<E> : equals

- Dues cues són iguals si les seves seqüències enllaçades fan referència a objectes iguals (equals) i en el mateix ordre.
- La verificació de la igualtat és una **CERCA**. És cerca un parell d'objectes que no són iguals (equals).

La classe Cua CuaEnll<E> : toString

- Per generar un String que "mostri" els elements de la cua, res millor que un **RECORREGUT** de la seqüència enllaçada.

La classe Cua CuaEnll<E>
Mètode quantsElements()

```
public int quantsElements () {
//calcula la cardinalitat del magatzem
// la implementació ha de ser recursiva

}
```


```
private int quantsElements(???){
    //sentències – crida recursiva
}
```