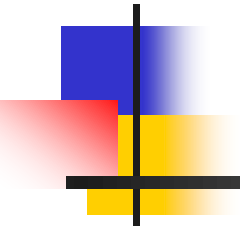
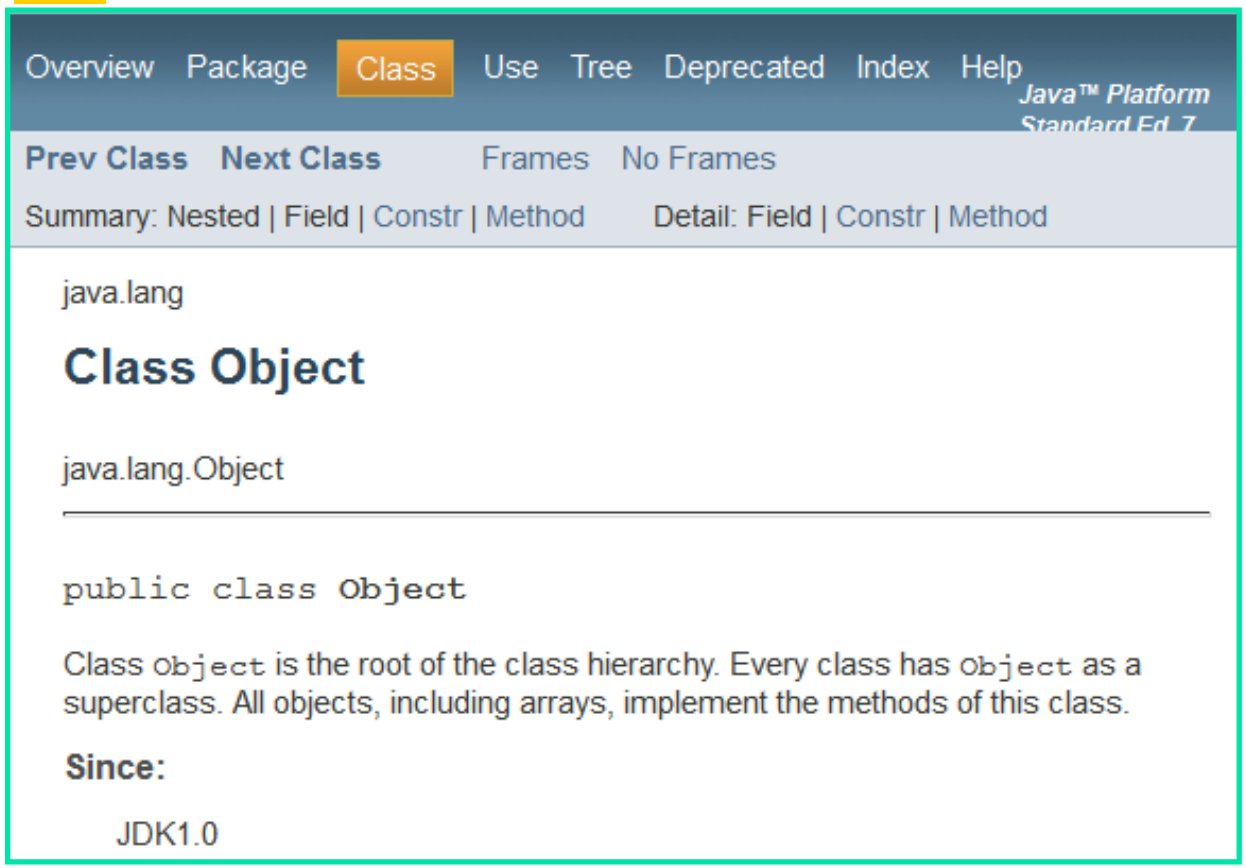
Classe Object



La superclasse de totes les classes

La classe Object



Overview Package Class Use Tree Deprecated Index Help

Java™ Platform Standard Ed. 7

Prev Class Next Class Frames No Frames

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

java.lang

Class Object

java.lang.Object

```
public class Object
```

Class `Object` is the root of the class hierarchy. Every class has `Object` as a superclass. All objects, including arrays, implement the methods of this class.

Since:

JDK1.0

Java.lang.Object

Api docs

http://docs.oracle.com/javase/7/docs/api/java/lang/Object.html

La classe Object

Qualsevol classe hereta tots els mètodes de la classe **`Object`** atès que tota classe és, en última instància, una subclasse de **`Object`**

Alguns dels mètodes d'**`Object`** sovint es **redefineixen** atès que el seu comportament no s'adapta a les necessitats de les subclasses.

La classe Object

Overview Package Class Use Tree Deprecated Index Help Java™ Platform Standard Ed. 7

Prev Class Next Class Frames No Frames

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

java.lang

Class String

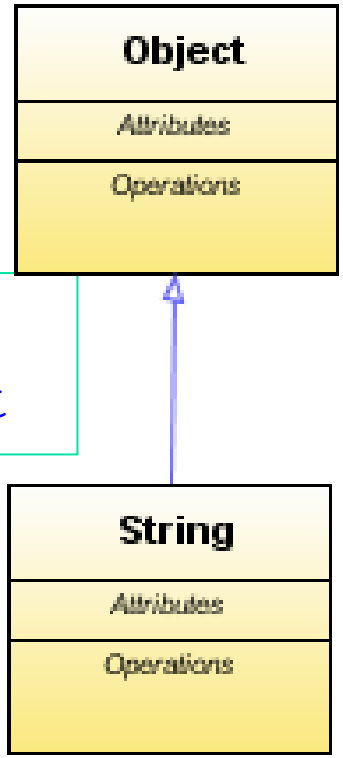
java.lang.Object
  java.lang.String

All Implemented Interfaces:

Serializable, CharSequence, Comparable<String>

```
public final class String
extends Object
implements Serializable, Comparable<String>, CharSequence
```

`String` és una subclasse d'`Object`



Java.lang.String

Api docs

La classe Object

Mètodes habitualment redefinits

`toString`: proporciona una representació en forma de String de l'objecte. Sovint es redefineix.

`System.out.print` i `System.out.println` **invoquen** aquest mètode per obtenir l'String que escriuen quan el seu paràmetre és un objecte.

La classe Object

Mètodes habitualment redefinits

`equals`: aquest mètode respon a la pregunta de si this i el paràmetre es poden considerar iguals.

Object implementa aquest mètode utilitzant la igualtat == (dos objectes es poden considerar iguals si són el mateix objecte). Aquest mètode es redefineix sempre que **es vulgui tenir una definició menys restrictiva de la igualtat**.

La classe Object

Mètodes habitualment redefinits

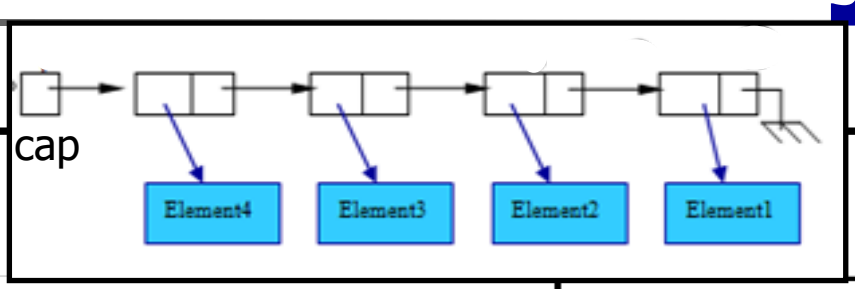
`clone`: aquest mètode s'utilitza per fer un duplicat d'un objecte.

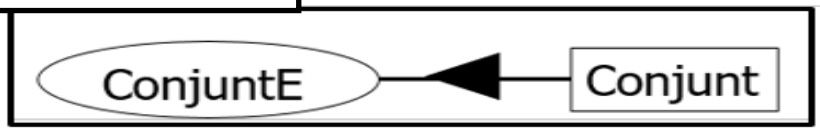
El funcionament per defecte és crear el nou objecte i copiar en cadascun dels seus atributs el mateix que té l'objecte del qui es copia.

Si aquests són referències no és duplica el referenciat.

La classe Conjunt

Exemple:





La classe Conjunt

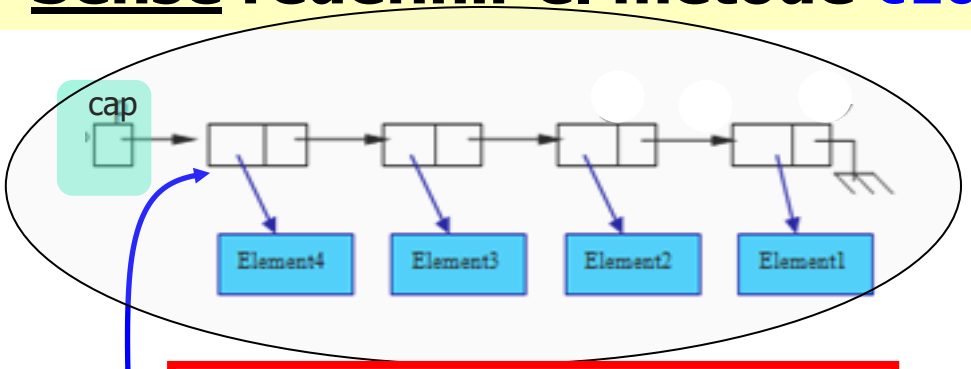
```
public class Conjunt implements ConjuntE{
    private class node{
        private Object ele;
        private node seg;
        public node(int e , node s){
                ele=e; seg=s;
        }
        public node(int e){
                this(e, null);
        }
    }
    private node cap;
```

Classe privada

Atribut

Exemple: clonem un Conjunt

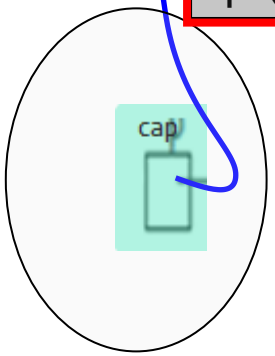
Sense redefinir el mètode `clone()`



Conjunt m

```
Conjunt m= new Conjunt();
m.inserir(Element1);
.....
Conjunt q;
q=(Conjunt)m.clone();
```

Crea un conjunt i realitza una assignació atribut a atribut

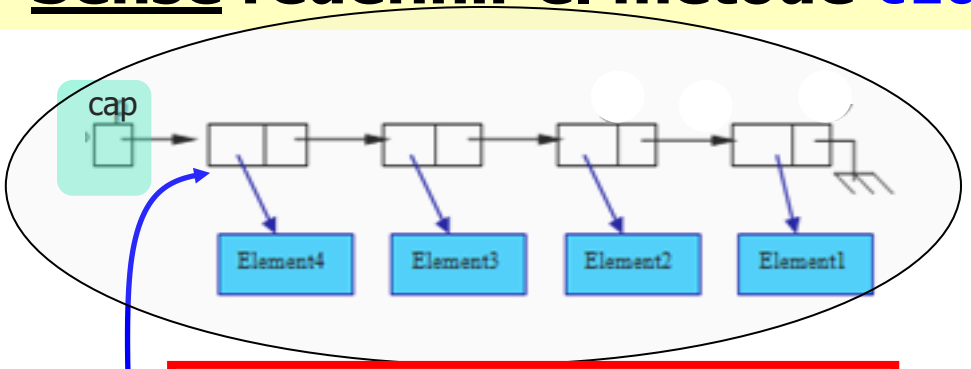


Conjunt q

```
protected Object clone()
                  throws CloneNotSupportedException
```

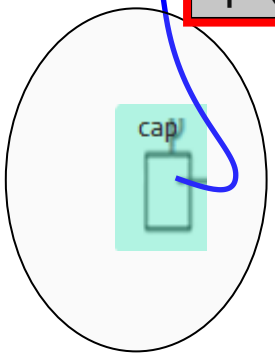
Exemple: clonem un Conjunt

Sense redefinir el mètode `clone()`



Conjunt m

```
Conjunt m= new Conjunt();
m.inserir(Element1);
.....
Conjunt q;
q=(Conjunt)m.clone();
```

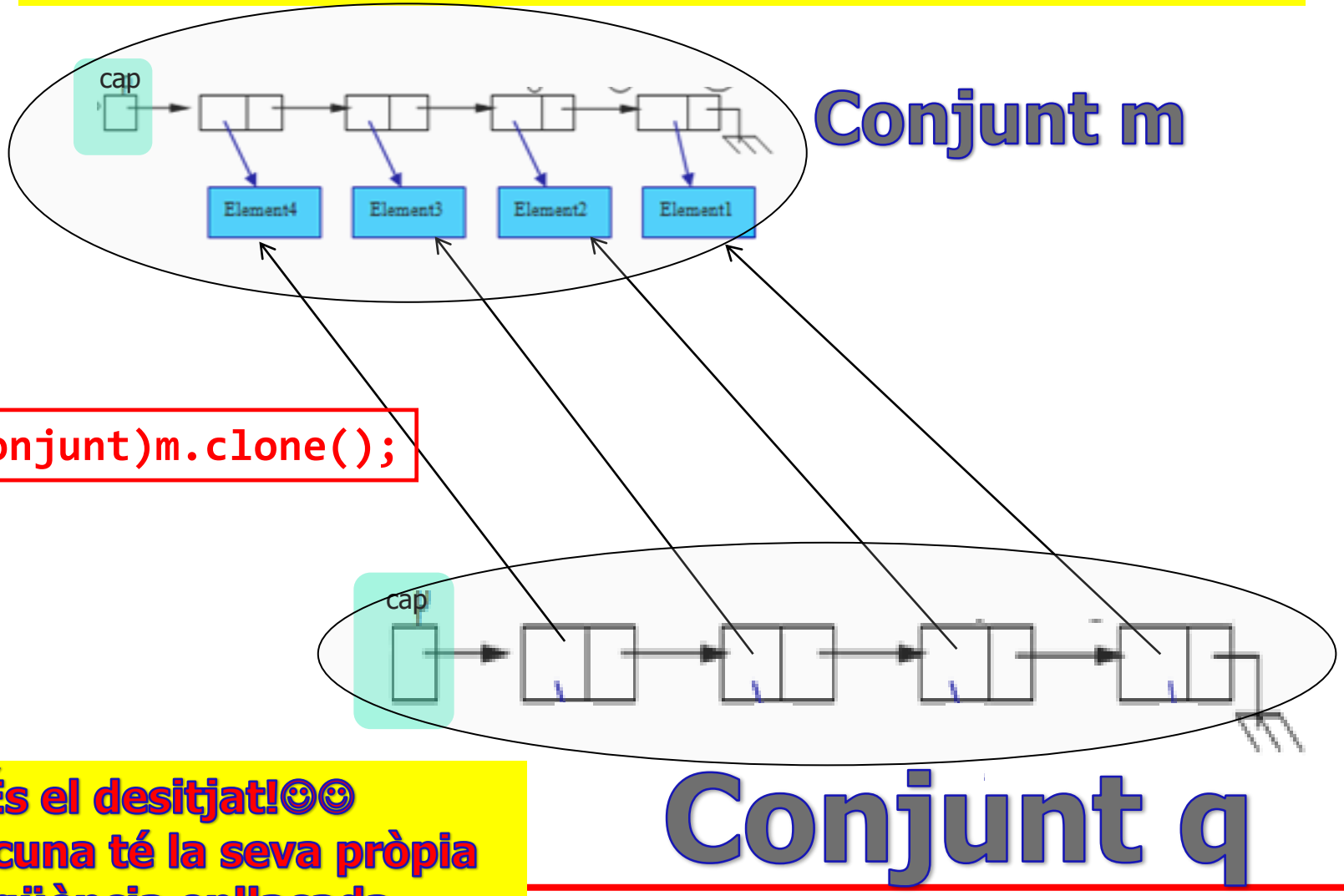


Conjunt q

Observeu que comparteixen la mateixa seqüència enllaçada. Cadascuna treballarà amb una referència diferent però el conjunt està "compartit"☹☹

Exemple: clonem un Conjunt

Amb redefinició del mètode clone(). VOLEM



És el desitjat!☺☺
Cadascuna té la seva pròpia seqüència enllaçada

La classe Object

protected Object **clone**()

Creates and returns a copy of this object.

java.lang

Interface Cloneable

All Known Subinterfaces:

AclEntry, Attribute, AttributedCharacterIterator, Attributes, Name

---

`public interface` **`Cloneable`**

A class implements the `Cloneable` interface to indicate to the `Object.clone()` method

Interface Cloneable

- Redefinir el mètode clone() amb l'objectiu de clonar un objecte.
- **Cal fer:**
  - 1.- Implementar la interfície Cloneable.
  - 2.- Redefinir el mètode clone() a la classe.

Estructura a seguir

```
public class NomClasse implements Cloneable{                    1
    public Object clone(){                                       2
        Object clon = null;
        try{
            clon = super.clone();
            ……..
        }
        catch(CloneNotSupportedException e){
            ……
        }
        return clon;
    }
```

Sentències que fan el duplicat "real"

Clonat d'un Conjunt

- **Solució 1:**
  - **Implementació iterativa.**
  - **Recorregut afegint nodes al final.**

- **Solució 2:**
  - **Fent els nodes clonables.**
  - **Implementació recursiva.**

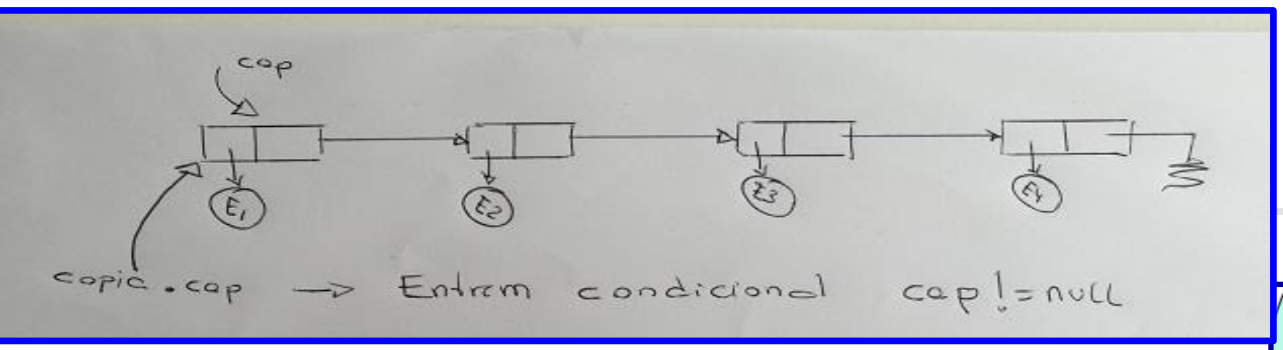
```
class node {
   Object inf;
   node seg;
   public node(Object o){this(o, null);}
   public node(Object o, node n){inf=o; seg=n;}
}
```

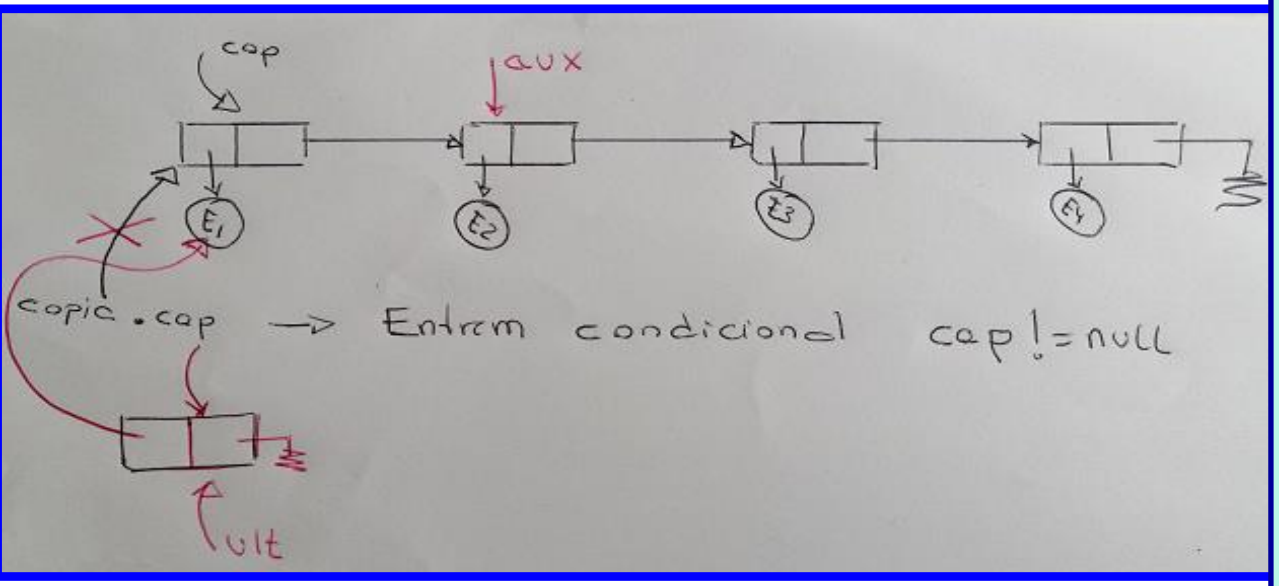
Clonat d'un Conjunt - **Solució 1**

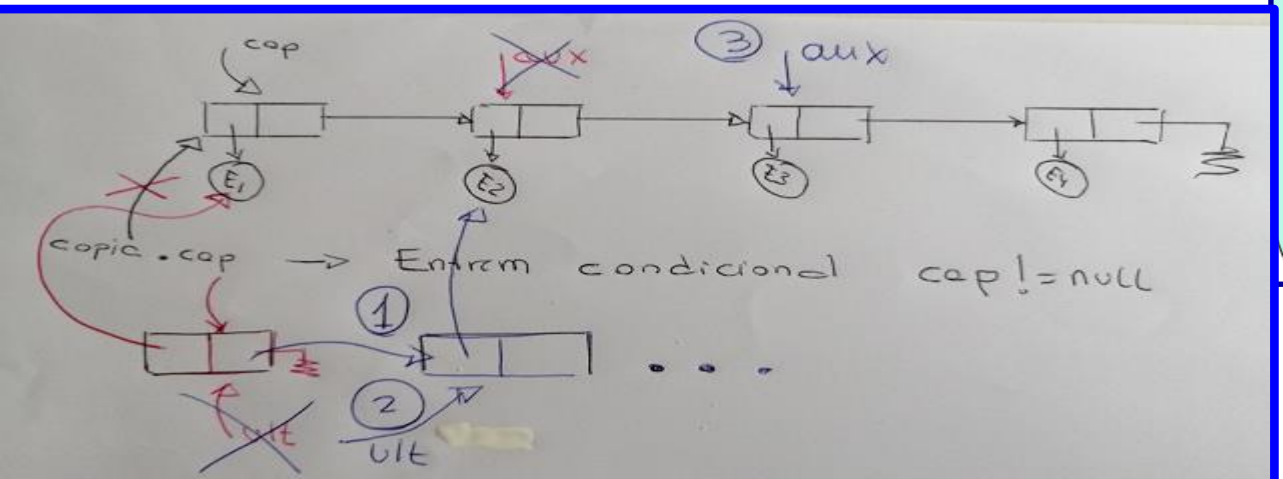
Referència al primer node de la seq enl

Crear una seqüència enllaçada sense capçalera afegint pel final

```
public class Conjunt implements ConjuntE,Cloneable{
   private node cap;
   public Object clone(){
        Conjunt copia=null;
        try{   copia=(Conjunt)(super.clone());
               if (cap !=null){ //com a mínim un element
                     copia.cap=new node(cap.inf);
                     node aux=cap.seg, ult=copia.cap;
                     while (aux !=null){
                           ult.seg=new node(aux.inf);
                           ult=ult.seg; aux=aux.seg;
                     }
               }
        }catch(CloneNotSupportedException e){return null;}
        return copia;
    }
   ...
} // fi classe
```







Solució 1

```
if (cap !=null){ //com a mínim un element
        copia.cap=new node(cap.inf);
        node aux=cap.seg, ult=copia.cap;
        while (aux !=null){
                ult.seg=new node(aux.inf);
                ult=ult.seg; aux=aux.seg;
        }
}
```

Clonat d'un Conjunt - **Solució 2**

Referència al primer node de la seqüència enllaçada

```
public class Conjunt implements ConjuntE, Cloneable{
    private node cap;
    public Object clone(){
        Conjunt copia=null;
        try{
                copia=(Conjunt)(super.clone());
                 if (cap!=null) //el conjunt no está buit
                      copia.cap=(node)(cap.clone());
        }
        catch(CloneNotSupportedException e ){return null;}
        return copia;
   }
   …
} // fi classe
```

Comportament per defecte!

Clonat d'un Conjunt - **Solució 2**

- **Fent els nodes clonables.**

```
class node implements Cloneable {
   Object inf; node seg;
   public node(Object o){this(o, null);}
   public node(Object o, node n){inf=o; seg=n;}
   public Object clone(){
        node copia=null;
        try{   copia=(node)(super.clone());
               if (seg !=null)
                     copia.seg=(node)(seg.clone());
        } // fi try
        catch (CloneNotSupportedException e){
                              return null;}
        return copia;
   } // fi mètode
} // fi classe
```