JAVA

Cloner en Java

Prof. Youssef Dahani

Cloner en Java

- Dans la programmation orientée objet, il arrive que l'on doive cloner un objet. Le "clonage" d'un objet pourrait se définir comme la création d'une copie par valeur de cet objet.

L'interface Cloneable

Pour pouvoir être clonée, une classe doit implémenter l'interface Cloneable. Celle-ci indique que la classe peut réécrire la méthode clone() héritée de la classe Object afin que ses instances puissent procéder à une copie de ses attributs vers une nouvelle instance (copie de l'objet par valeur). Par convention, les classes implémentant l'interface Cloneable doivent réécrire la méthode Object.clone() (qui est protected) avec une visibilité public.

Notez que l'interface Cloneable ne contient PAS la méthode clone(). Par conséquent, il est impossible de cloner un objet si sa classe ne fait qu'implémenter cette interface. Même si la méthode clone() est appelée par réflexion, son succès n'est pas garanti. Une classe implémentant l'interface Cloneable doit nécessairement réécrire la méthode clone() pour pouvoir être clonée ; à l'inverse, une classe réécrivant la méthode clone() doit implémenter l'interface Cloneable sous peine de se retrouver avec une CloneNotSupportedException.

La méthode clone()

Comme nous l'avons vu, une classe implémentant l'interface Cloneable doit réécrire la méthode clone(). La méthode clone() doit retourner une copie de l'objet que l'on veut cloner. Cette copie dépend de la classe de l'objet, cependant elle doit respecter des conditions (dont certaines par convention)

La méthode clone()

- Pour un objet "clonable" x :
  - l'expression x.clone() != x doit renvoyer true ;
  - l'expression x.clone().getClass() == x.getClass() doit renvoyer true (par convention) ;
  - l'expression x.clone().equals(x) doit renvoyer true (par convention).
  - Par convention, l'objet retourné doit être obtenu par un appel à la méthode super.clone() .
  - Si une classe et toutes ses classes parentes (exceptée la classe Object) respectent cette convention, alors l'expression x.clone().getClass() == x.getClass() renvoie bien true.
  - Par convention, l'objet retourné doit être indépendant de l'objet cloné, c'est à dire que tous les attributs non immuables devront être eux aussi clonés.
  - Donc, pour résumer, les points importants de la réécriture de la méthode clone() sont :
    - récupérer l'objet à renvoyer en appelant la méthode super.clone(),
    - cloner les attributs non immuables afin de passer d'une copie de surface à une copie en profondeur de l'objet.

Le Clonage : Exemple

```
public class Patronyme implements Cloneable {

        private String prenom;  private String nom;

         public Patronyme(String prenom, String nom) {
                 this.prenom = prenom;
                this.nom = nom;
         }
         public Object clone() {
                 Object o = null;
                  try {
                          // On récupère l'instance à renvoyer par l'appel de la
                          // méthode super.clone()
                          o = super.clone();   }

                 catch(CloneNotSupportedException cnse) {
                           // Ne devrait jamais arriver car nous implémentons
                          // l'interface Cloneable
                          cnse.printStackTrace(System.err);
                           }
                 // on renvoie le clone   return o;  }
        }
```

Le Clonage : Exemple

```
public class Personne implements Cloneable {
    private Patronyme patronyme;  private int age;

    public Personne(Patronyme patronyme, int age)
    {   this.patronyme = patronyme;     this.age = age;  }

     public Object clone() {     Personne personne = null;
     try {      // On récupère l'instance à renvoyer par l'appel de la         // méthode super.clone()
    personne = (Personne) super.clone();
    } catch(CloneNotSupportedException cnse) {
     // Ne devrait jamais arriver car nous implémentons
     // l'interface Cloneable        cnse.printStackTrace(System.err);
       }
      // On clone l'attribut de type Patronyme qui n'est pas immuable.
      personne.patronyme = (Patronyme) patronyme.clone();          // on renvoie le clone      return personne;
    } }
```

Tester