LE PATTERN SINGLETON

Ludovic Liétard

Le pattern SINGLETON

- Rôle : *créateur*
- Problème : garantir qu'une classe n'a qu'une seule instance et fournit un point d'accès à cette instance.

Le pattern SINGLETON

- Solution :
  - L'instance est une variable de classe (`static`).
  - Une seule méthode est accessible depuis l'extérieur (`public`). Elle donne l'accès à l'instance.
  - Obligation de passer par cette méthode (encapsulation de la variable de classe : `private`).

Le pattern SINGLETON

- Solution (suite):
  - Lorsque cette méthode d'accès est utilisée : si l'instance de classe n'existe pas encore, elle est créée.
  - Impossibilité de créer cette instance autrement

    (=> constructeur `private`)

Exemple : Gérer un compteur

- Utilisation du compteur :

```
public class Princip {
        public static void main(String[] args) {

                System.out.println(Compteur.obtenirValeur());
                Compteur.incremente();
                Compteur.incremente();
                System.out.println(Compteur.obtenirValeur());


        }

}
```

Déclaration du compteur :

```
public class Compteur {
        private static Compteur clui=null;
        private static Integer valeur = new Integer(0);
        private Compteur(){}

       public static int obtenirValeur(){
          if (clui = = null) {
                  clui=new Compteur();
          }
          return clui.valeur.intValue();
        }

      public static void incremente(){
          int a;
          a = Compteur.celui.obtenirValeur();
         a++;
        celui.valeur=new Integer(a);
      }
}
```

Le pattern SINGLETON

- Mais un risque subsiste en cas de plusieurs threads
  - Pour accéder à l'objet unique
  - Pour manipuler l'objet unique

Le pattern SINGLETON

- Exemple :
  - Deux threads en parallèle, avec les exécutions suivantes de la méthode incrémente (au départ, le compteur vaut 0)

Thread 1

```
int a;
1) a = Compteur.obtenirValeur();
3)  a++;
5) valeur=new Integer(a);
```

Thread 2

```
int a;
2) a = Compteur.obtenirValeur();
4) a++;
6) valeur=new Integer(a);
```

- Quelle valeur du compteur obtient-on ?
- Quelle valeur aurions nous du obtenir ?

Le pattern SINGLETON

- On obtient une valeur de 1
- Nous aurions du obtenir la valeur 2
- Il faut donc synchroniser les méthodes

Déclaration du compteur :

```
public class Compteur {
        private static Integer valeur = null;
        private Compteur(){
}

public static synchronized int obtenirValeur(){
        if (valeur = = null) {
                valeur = new Integer(0);
         }
        return valeur.intValue();
}

public static synchronized void incremente(){
        int a;
        a = Compteur.obtenirValeur();
        a++;
        valeur=new Integer(a);
} }
```