Conception Object

OCP : *Open Close Principle*

Alain Plantec

UBO

13 novembre 2012

Problème de l'évolution du code

- Logiciel vivant $\rightarrow$ revue continuelle de la mise en œuvre.
- Une modification à la fois.
- Une modification implique :
  - revue complète et tests ;
  - réinstallation chez les utilisateurs,
  - remise en cause du code des utilisateurs (ajout de paramètre, de variable d'instance...)
- $\rightarrow$ Opérations couteuses et risquées.

Ouverture-Fermeture

Le principe d'*Ouverture-Fermeture* (ou **OCP** pour *Open-Closed principle*) a été bien décrit par Bertrand Meyer.

Ennoncé

Un module doit parvenir à être ouvert en extension et fermé en modification.

- **Ouvert en extension** : on peut enrichir la structure de données ou ajouter des services.
- **Fermé en modification** : dès qu'un module est utilisé par d'autre, les services existants doivent être stables.

Exemple

```
class Printer {
   String device;
   Printer (String target) {
      this.target = target
   }
   void printDoc (Document d) {
      if (device.equals("printer")) {
         printOnPrinter (d);
      } else {
         printOnScreen (d);
      }
   }
   void printOnPrinter (Document d) { ... }
   void printOnScreen (Document d) { ... }
}
```

Autre exemple

```
class Agenda {
    ArrayList<Elem> elements;
    ...
    ArrayList<Elem> recByDate(String d) { ... }
    ArrayList<Elem> recByName(String d) { ... }
    ...
}
```

Problème (Java) : on veut ajouter une recherche avec des expression régulières sur le nom ou une autre sur la description ...

Autre exemple

```
class Item {
}
class ItemA extends Item {
   void doitForA ();
}
class ItemB extends Item {
   void doitForB ();
}

class ItemList {
   ArrayList<Item> items;
   doit () {
      for (int i = 0; i < items.size(); i++) {
         Item curr = items.get(i);
         if (curr instanceof ItemA) {
```

Comment respecter l'OCP ?

Techniques de base de l'objet :

- Rendre privés tous les attributs.
- Seuls sont visibles (public), les méthodes qui implémentent les services.
- Pas de globales.
- Rendre local la gestion des variables (énumérés...), des flags
- Méthodes abstraites pour coder ce qui est susceptible de changer.
- Ne pas interroger le type, pas de RTTI (proscrire **instanceof** ou **isKindOf :**)

Comment respecter l'OCP ?

Changer la conception :

- Utiliser des classes abstraites.
- Privilégier l'association à l'héritage.
- Appliquer des patrons de conception : Visiteur, Stratégie ...

Le terrain de chasse : dans le code, les alternatives, les flags, les variables globales