LES PATRONS DE CONCEPTION

(DESIGN PATTERNS)

Ludovic Liétard

1

Introduction

- Les design patterns ont été proposés par Erich Gamma, Richard Helm, Ralph Johnsson et John Vlissides en 1995
- Le but est de proposer un catalogue de modèles (= pattern = patrons) pour apporter des solutions à des problèmes de conception (= design = conception) orientées objet
- Termes équivalents : design pattern, patron de conception, modèle de conception

2

Introduction

- « Chaque patron décrit un problème récurrent dans notre environnement, puis le cœur de la solution qui sera réutilisable indéfiniment quelle que soit la mise en pratique choisie »

⇒ un ensemble de patrons est disponible

⇒ chaque patron identifie un problème et apporte une solution prédéfinie

3

Introduction

- Un patron de conception est défini par :

Nom : chaque pattern à un nom qui l'identifie de manière unique

Problème : le problème qu'il s'efforce de résoudre

Solution : la solution apportée

Conséquences : les effets résultants de la mise en oeuvre

4

Introduction

Un Rôle :

***Créateur* : pour créer des objets**

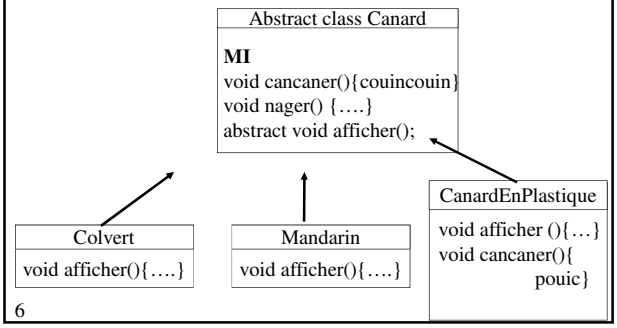
***Structurel* : pour s'occuper de la composition des classes et des objets**

***Comportemental* : pour répartir l'interaction des classes et des objets, se répartir les responsabilités**

5

Interêt des patrons de conception (sur un exemple)

- Considérons l'application suivante :



6

Interêt des patrons de conception (sur un exemple)

- Utilisons le principe suivant :
  - Extraire les parties variables et les encapsuler pour les modifier plus tard (ou les augmenter) sans affecter celles qui ne varient pas
- Ce concept constitue la base de presque tous les designs patterns. Tous les patterns fournissent un moyen de permettre à une partie du système de varier indépendamment de toutes les autres

7

Interêt des patrons de conception (sur un exemple)

- Qu'est ce qui varie ?
  - Les comportements cancaner()
- Cancaner()
  - Coin Coin (tous les canards sauf le plastique)
  - Pouic Pouic (le canard en plastique)

8

Interêt des patrons de conception (sur un exemple)

- Le pattern Stratégie définit des classes pour implémenter ces comportement
- Désormais, les comportements de Canard résideront dans une classe distincte – une classe qui implémente une interface comportementale particulière.
- Ainsi, les classes de canard n'auront besoin de connaître aucun détail de l'implémentation de leur propre comportement

9

Interêt des patrons de conception (sur un exemple)

- Un canard va maintenant déléguer ses comportements au lieu d'utiliser la méthode cancaner()
  - une variable d'instance est rajoutée à la classe Canard (comportementCancan)
  - elles contiennent le bon comportement du canard (elles sont instanciées à la création du canard)
  - la méthode effectuerCancan() utilise cette VI

10

Interêt des patrons de conception (sur un exemple)

```
abstract class Canard {
        ComportementCancan monCancan;
        public abstract void afficher();
        public void effectuerCancan(){
                monCancan.cancaner();
        }
        public void nager(){
                System.out.println("je nage");
}       }
```

11

Interêt des patrons de conception (sur un exemple)

```
interfaceComportementCancan{
    void cancaner(); }

Class CoinCoin implements ComportementCancan{
    void cancaner(){
        System.out.println ("Coin Coin..");     }
}

Class PouicPouic implements ComportementCancan{
    void cancaner(){
        System.out.println ("Pouic Pouic.. ");     }
}
```

12

Interêt des patrons de conception (sur un exemple)

```
class Colvert extends Canard {

    public Colvert(){
        ComportementCancan = new CouinCouin();
    }
    public void afficher(){
        System.out.println("Je suis un Colvert");
    }
}
```

13

Interêt des patrons de conception (sur un exemple)

```
class CanardPlastique extends Canard {

    public CanardPlastique(){
        ComportementCancan = new PouicPouic();
    }
    public void afficher(){
        System.out.println("Canard en plastique");
    }
}
```

14

Interêt des patrons de conception (sur un exemple)

- Il est également possible de modifier dynamiquement le comportement d'un canard (il suffit de modifier ses VI).
- Conclusion : La relation « possède un » peut être préférable à « est un ».
- Nous venons de mettre en œuvre le **pattern Stratégie**

15