Conception Object

Le décorateur

Alain Plantec

UBO

15 octobre 2014

Un exercice

Énoncé

On veut développer un logiciel pour créer des cartes de visite. Les cartes peuvent être décorées :

- couleur de fond
- bords colorés
- double bordure
- ...

Le calcul du cout est en fonction des options de décoration.

Quelle solution pour ce problème ?

Une première solution

Une classe avec autant de flags que de possibilités :

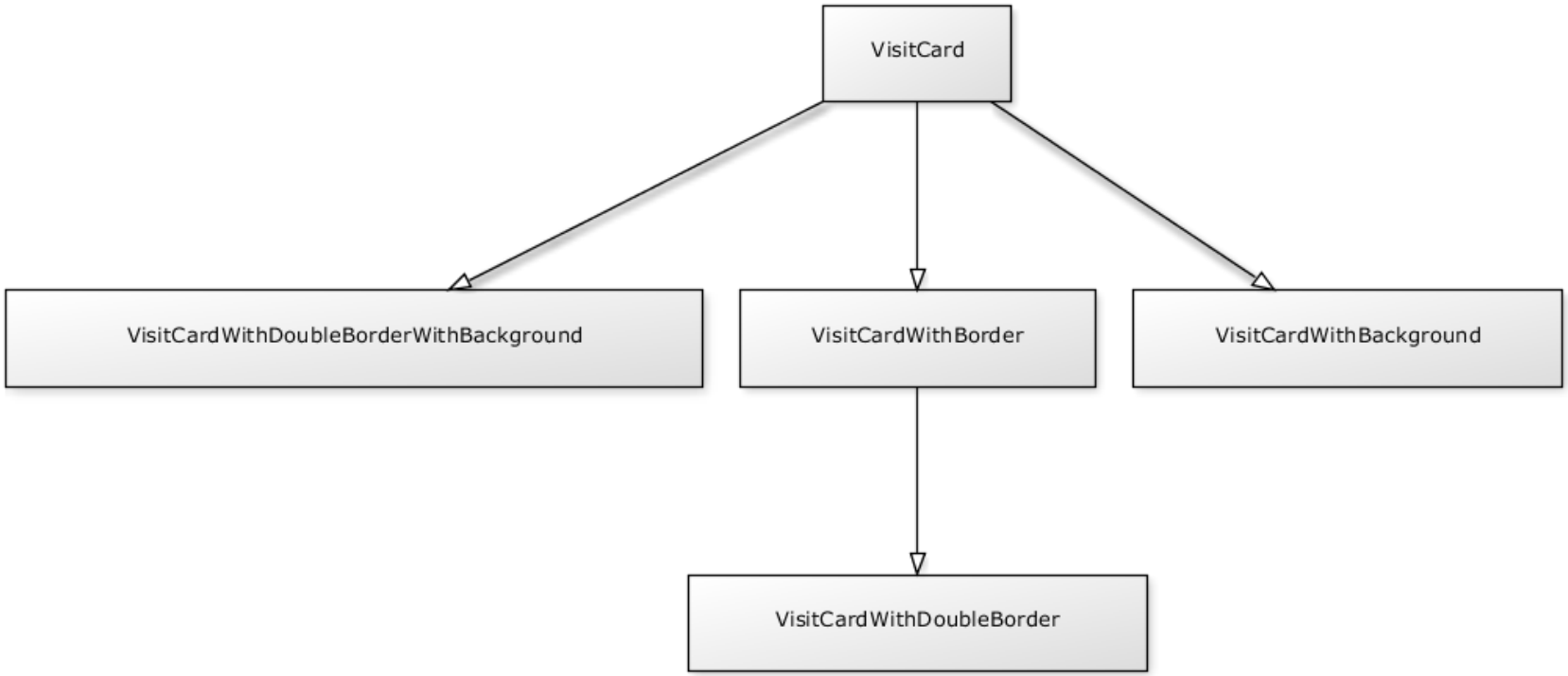
```
Object subclass : #VisitCard
    instanceVariableNames : 'borderColor withDoubleBorder backgroundColor'
    classVariableNames : ''
    poolDictionaries : ''
    category : 'CO-DIP'
```

Pourquoi cette solution n'est elle pas satisfaisante ?

- pour ajouter une option de dessin, il faut modifier la classe et les méthodes qui dessinent et qui calculent le cout
- $\rightarrow$ on ne respecte pas l'OCP, la conception ne peut pas évoluer facilement

Une seconde solution

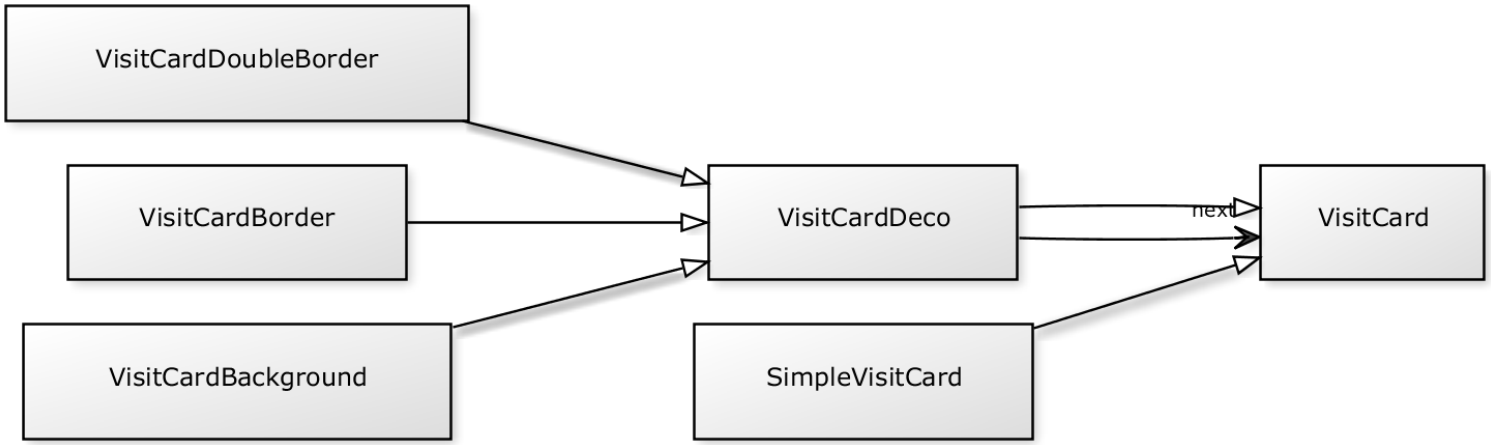
Des sous-classes ?



Pourquoi cette solution n'est elle pas satisfaisante ?

- explosion du nombre des sous-classes
- le code est dupliqué

Solution du décorateur



Utilisation :

```
v  := visitCardBackground with : (visitCardBorder with : SimpleVisitCard new).
v cost.
v annotation.
```

- intérêt : on peut ajouter de nouvelles décorations.
- mais comment cela marche t-il ?

Fonctionnement du décorateur

```
SimpleVisitCard>>cost
    ^ cost
SimpleVisitCard>>annotation
    ^ 'Carte de visite'

VisitCardDeco>>cost
    ^ self next cost
VisitCardDeco>>annotation
    ^ self next annotation

VisitCardBorder>>cost
    ^ super cost + 0.5
VisitCardBorder>>annotation
    ^ super annotation, 'avec Bordure'
```

On a une chaine de décorateurs, chaque décorateur cumul son propre coût à celui du suivant.

Exemple : Stream en Java

Extrait de
http://stackoverflow.com/questions/6366385/decorator-pattern-for-io

```
//First open an inputstream of it :
FileInputStream fis = new FileInputStream("/objects.gz");

//We want speeeed, so let's buffer it in memory :
BufferedInputStream bis = new BufferedInputStream(fis);

//The file is gzipped, so we need to ungzip it :
GzipInputStream gis = new GzipInputStream(bis);

//We need to unserialize those Java objects :
ObjectInputStream ois = new ObjectInputStream(gis);

//Now we can finally use it :
SomeObject someObject = (SomeObject) ois.readObject();
```

Exercice

Comment décorer du texte en html avec le décorateur ?

```
<b>Texte en gras</b>
<i>Texte en italique</i>
<b><i>Texte en gras et italique</i></b>
<big>Texte plus grand </big>
<big><b><i>Texte plus grand en gras et italique</i></b></big>
```

Programmez un modèle de texte riche simplifié et des tests pour mettre en gras, italique, gras-italique, etc

Exercice

On veut simuler une machine à café :

- La machine affiche les boissons disponibles, café, thé, chocolat, avec ou sans sucre, avec ou sans lait, plus ou moins fort....
- Si une personne insère des pièces de monnaie, il peut choisir certaines boissons (en fonction de leur prix).
- Au final, l'utilisateur choisi sa boisson pour de bon, récupère sa monnaie et sa boisson.

Programmer le simulateur. Vous le validez à l'aide des tests unitaires et des tests fonctionnels (sans interface utilisateur)