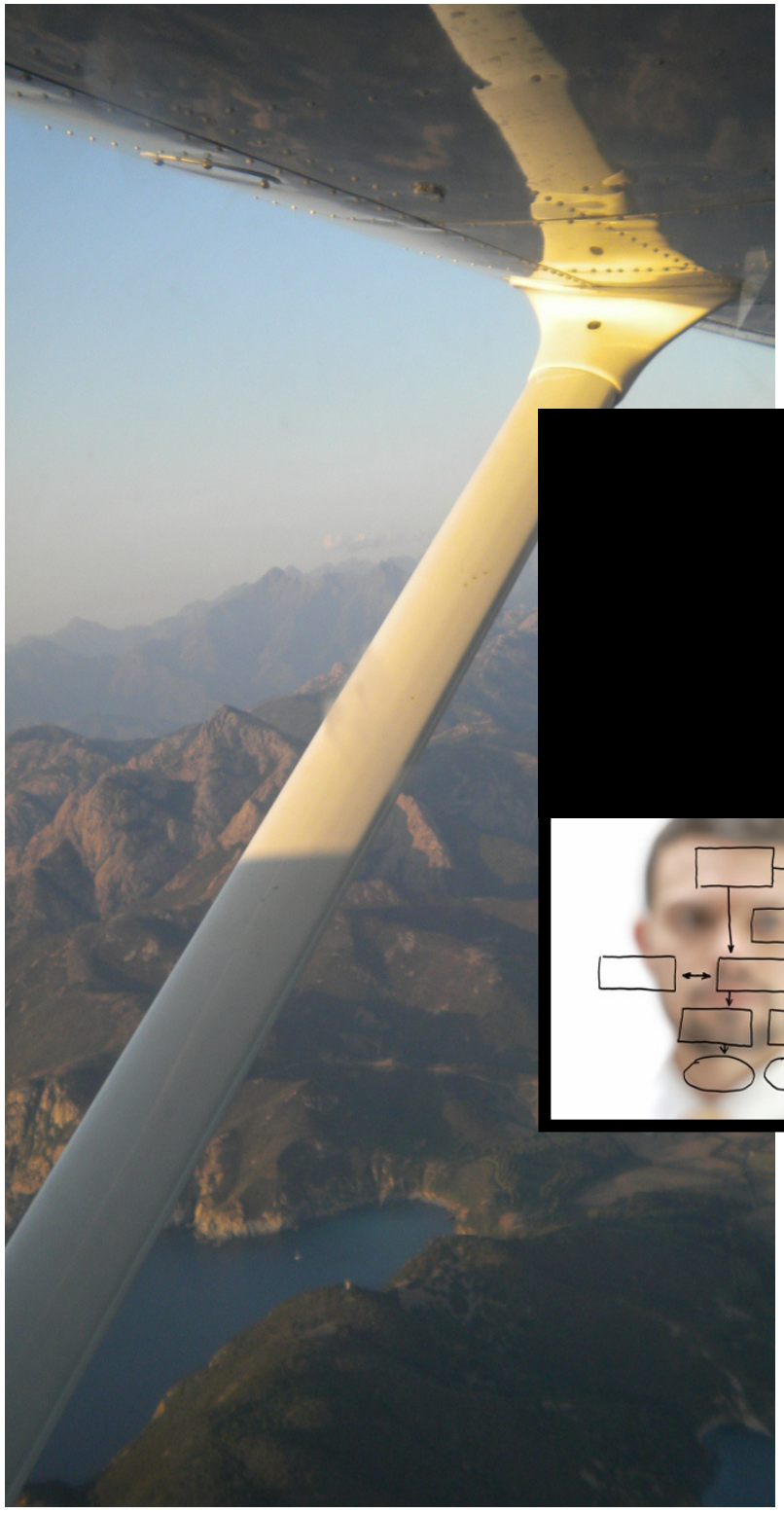
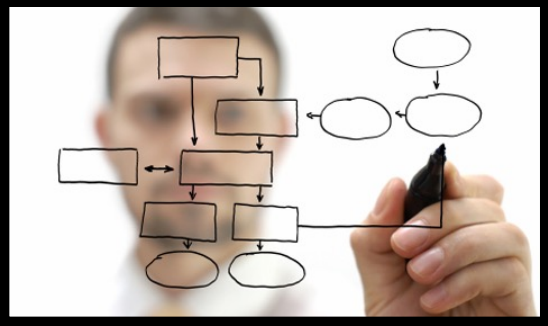
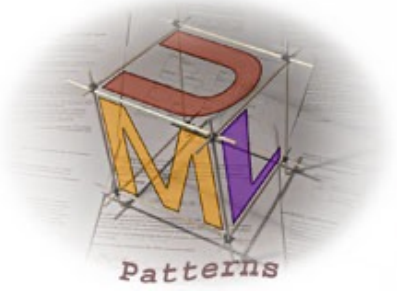


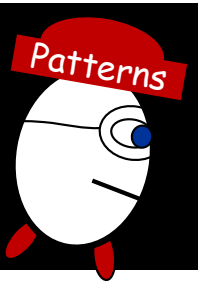
Université de Corse
2025-2026
MASTER DFS-DE 1ère année

Cours PATTERNS

CH 2.3 - Patterns GOF Structurels





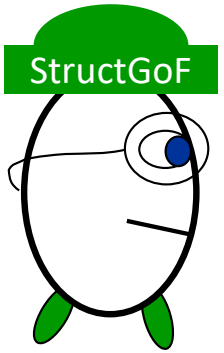


CH 2 – PATTERNS GangOfFour

Introduction

1 – Patterns créationnels

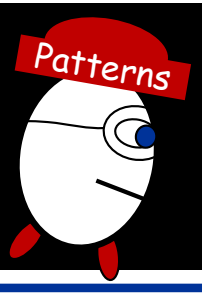
2 - Patterns structurels



Adaptateur (*Adapter*)
Décorateur (*Decorator*)
Composite (*Composite*)
Procuration (*Proxy*)
Façade (*Facade*)

Pont (*Bridge*)
Pois Mouche (*Flyweight*)

3 - Patterns comportementaux



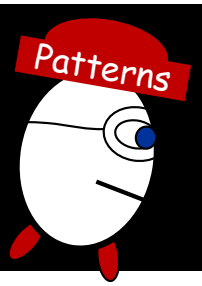
3 – Patterns Structurels
Adaptateur «adapter»

Problème

Comment faire collaborer des classes ayant des interfaces incompatibles?

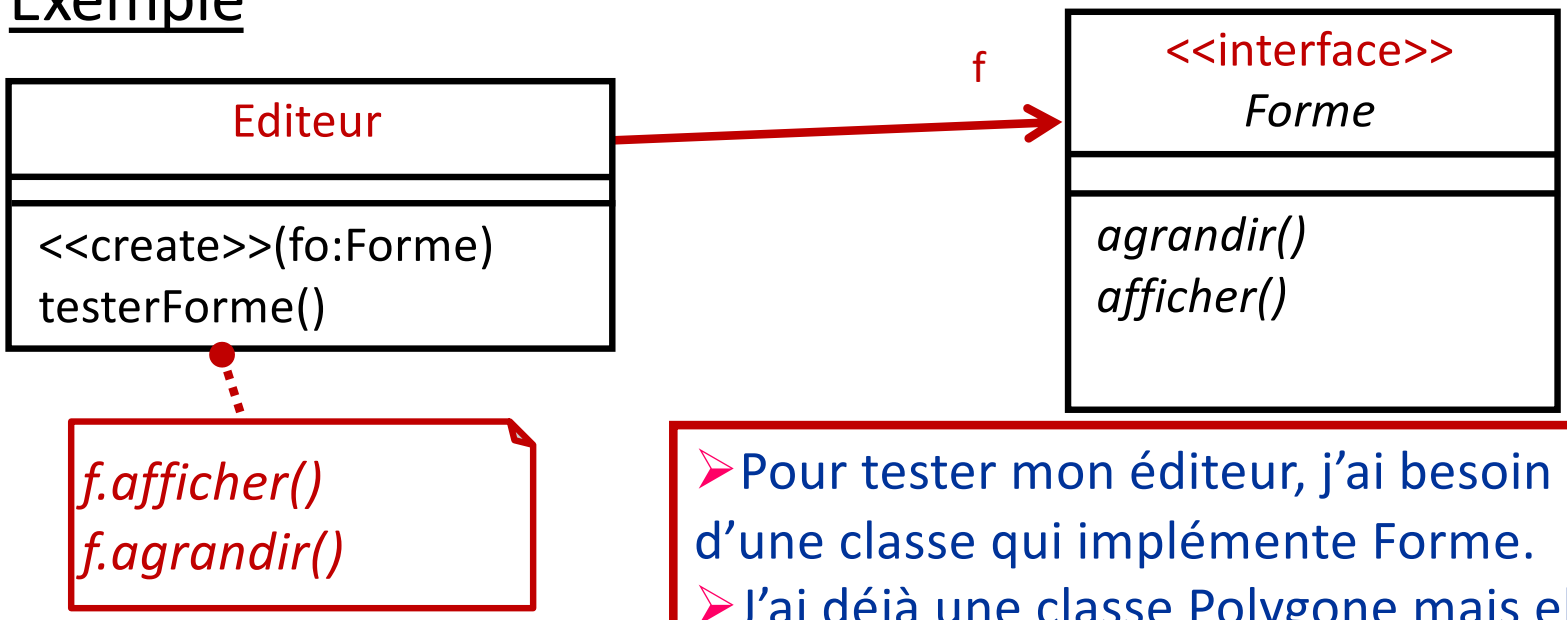
Solution

➢Convertir l'interface d'origine en une autre interface via un objet adaptateur intermédiaire.

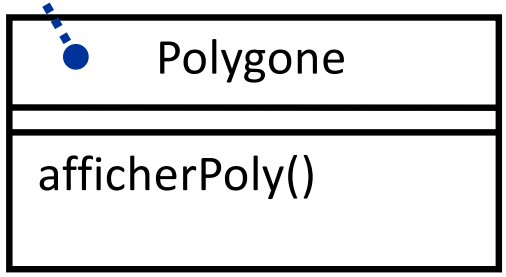


Adapter

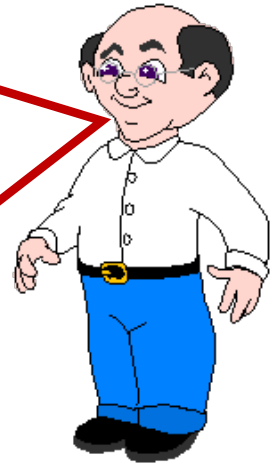
Exemple

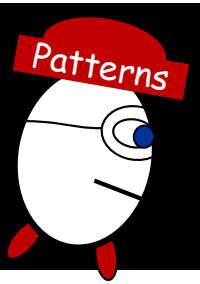


Classe Existante

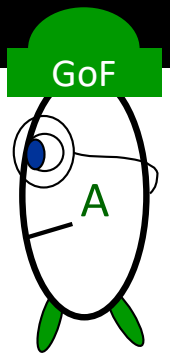


- Pour tester mon éditeur, j'ai besoin d'une classe qui implémente Forme.
- J'ai déjà une classe Polygone mais elle n'implémente pas Forme.
- Je voudrais l'utiliser mais **sans modifier ma classe Editeur et sans modifier Polygone**.
- Comment Faire?????

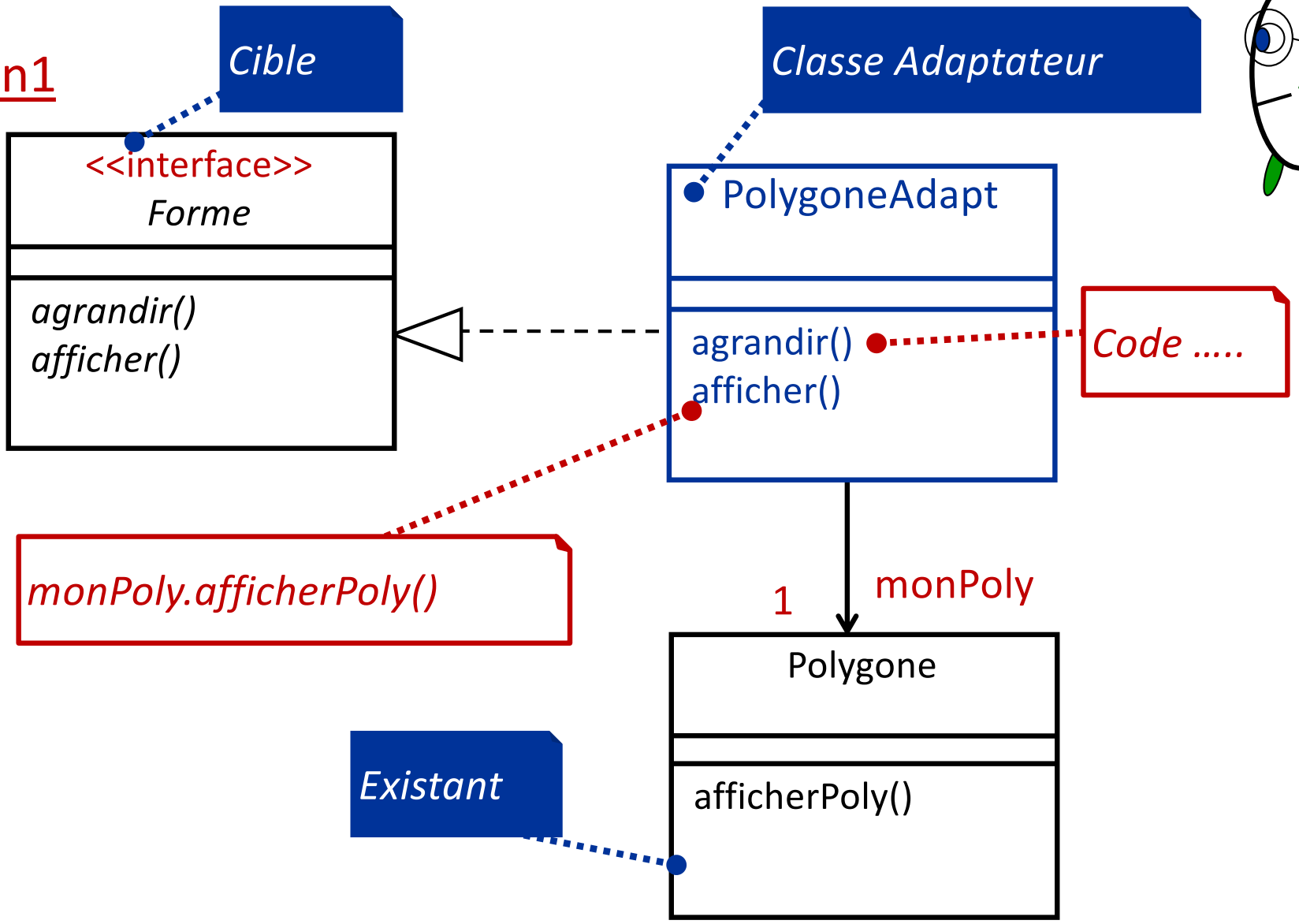




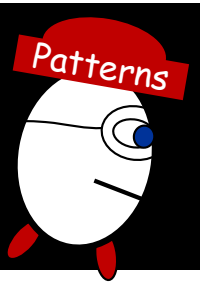
Adaptateur d'objet

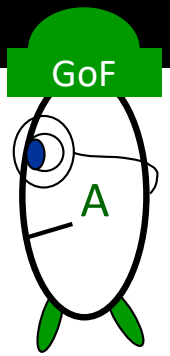


Solution1

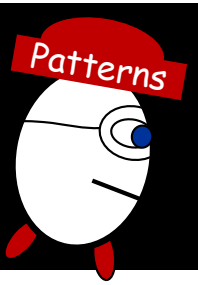


Adaptateur de classe

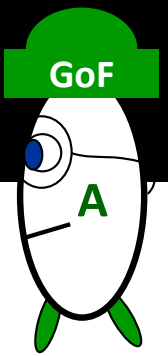




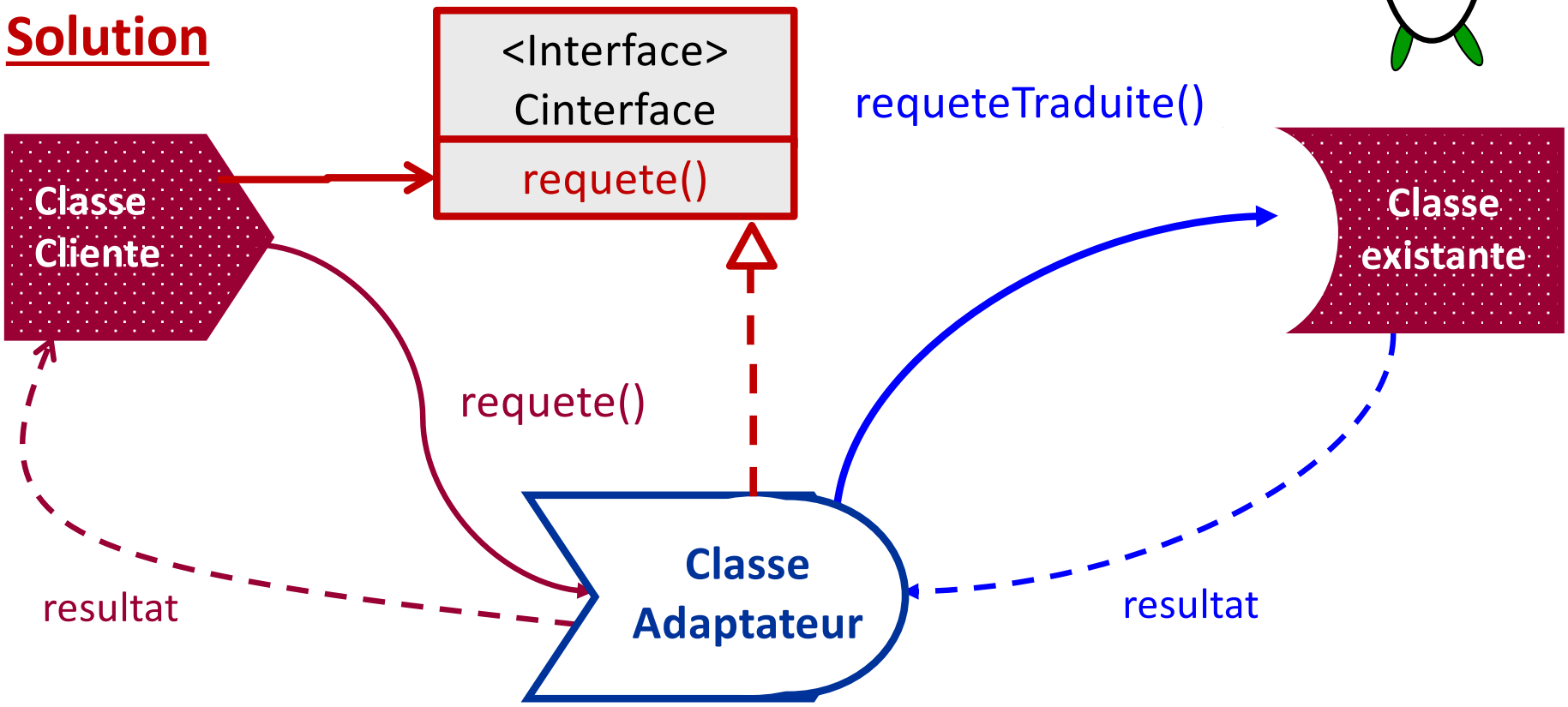
Solution2



Adapter

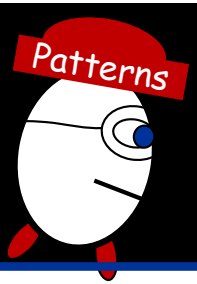


Solution



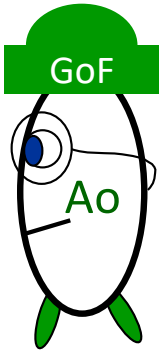
- Implémente requete() (**interface cible**)
- Contient une instance de la classe existante

ou est une instance d'une sous-classe de la classe existante

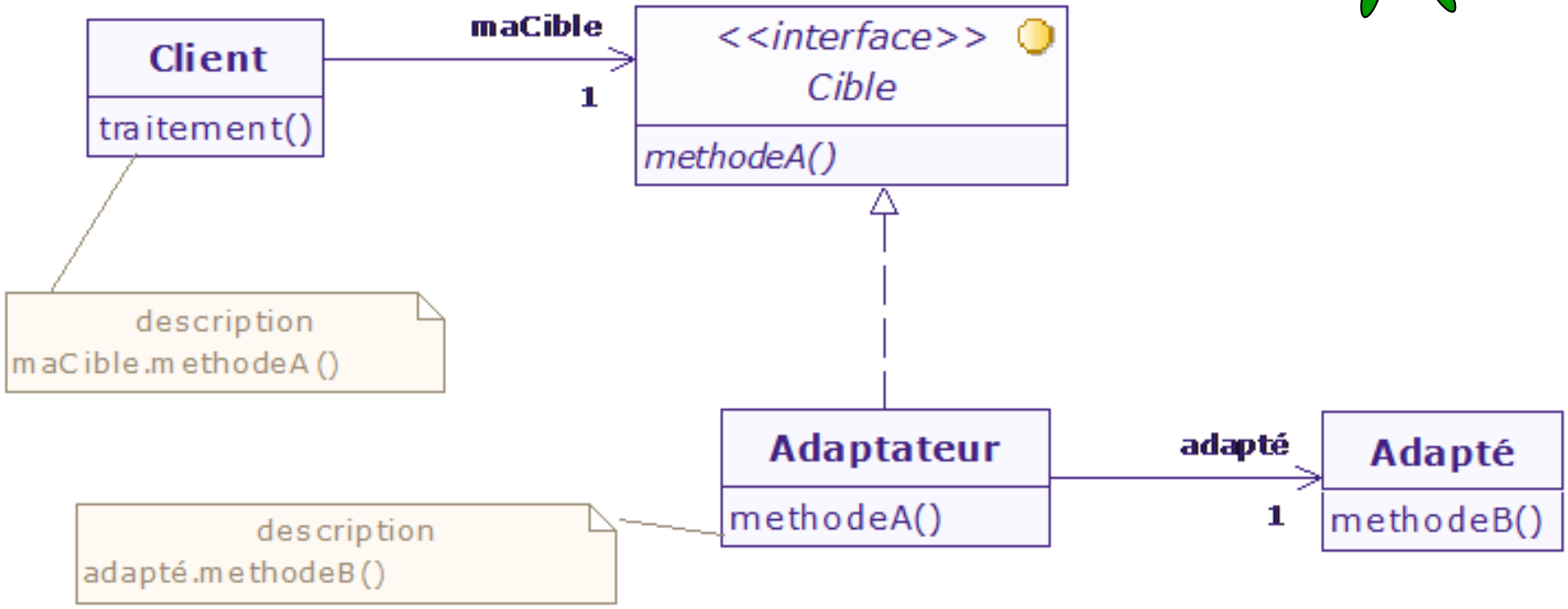


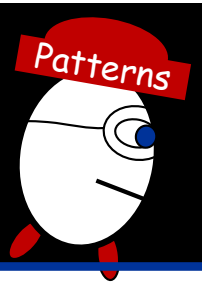
Adapter

Solution (cas général)

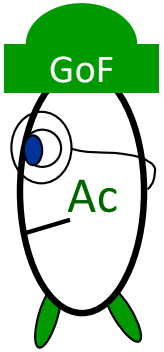


Version 1 : Adaptateur d'objet



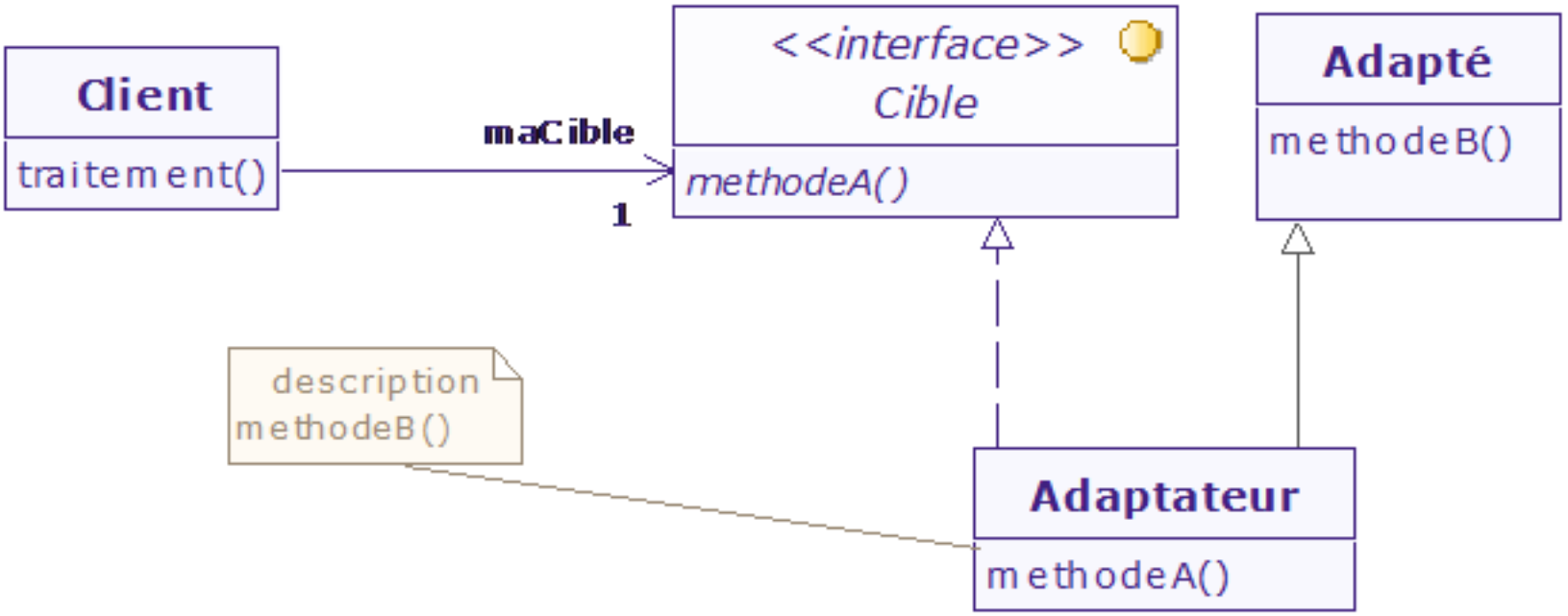


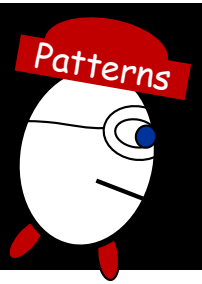
Adapter



Solution (cas général)

Version 2 : Adaptateur de classe

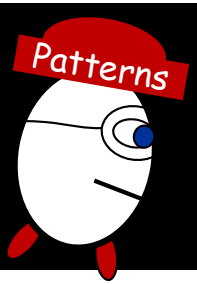




Adapter

Conséquences

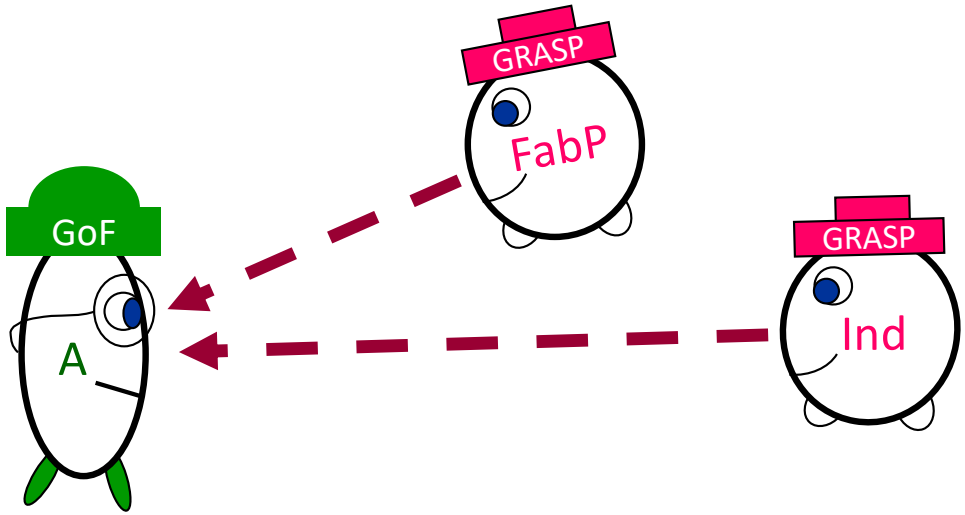
Adaptateur de Classe	Adaptateur d'objet
Utilise l'héritage	Utilise la Composition
Offre la possibilité à la classe Adaptateur de redéfinir des méthodes de Adapté	La redéfinition de méthodes de Adapté nécessite de sous-classer Adapté et de faire référence à la sous-classe
N'introduit qu'un seul objet (instance d'adaptateur)	Nécessite Deux objets (adaptateur+adapté)



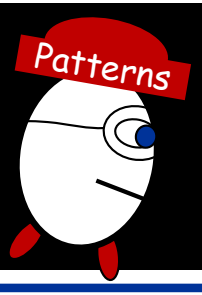
Adapter

Conséquences

Le pattern Adaptateur est une sorte d'indirection et une fabrication pure.



Retour Plan

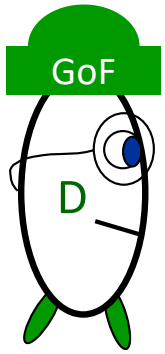


Décorateur *(Decorator)*

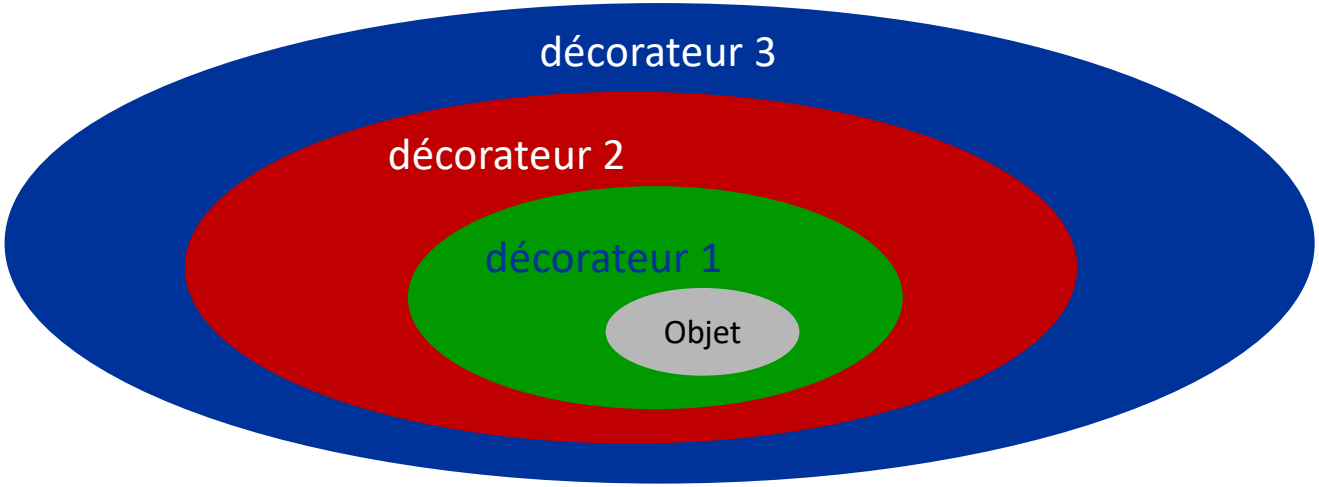
Problème

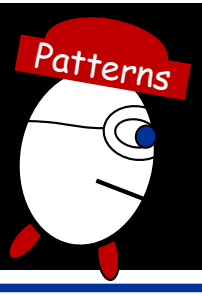
Comment attacher dynamiquement des responsabilités supplémentaires à un objet?

Solution



Envelopper l'objet dans un objet enveloppe (le décorateur) chargé d'assumer la nouvelle responsabilité.

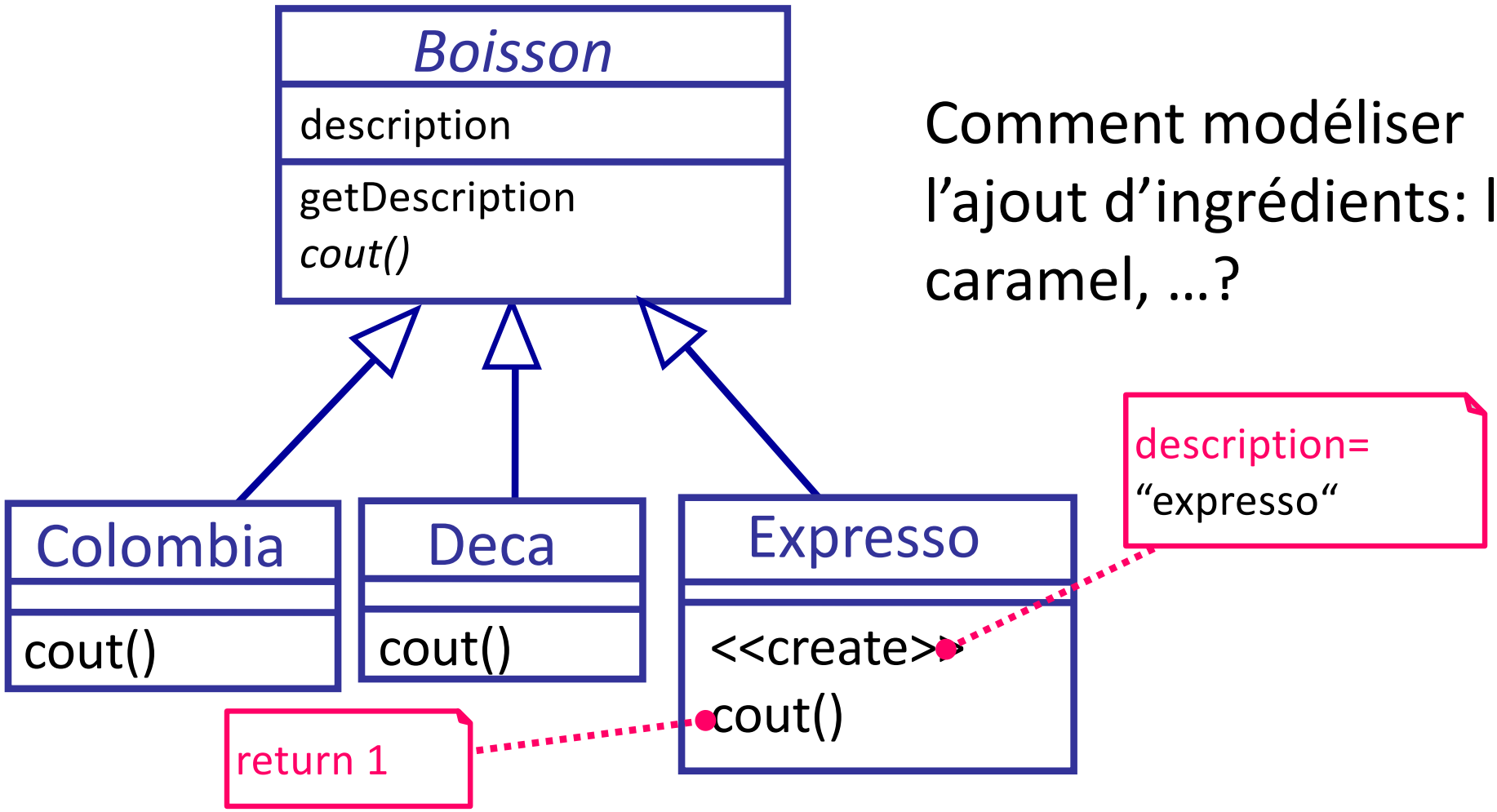




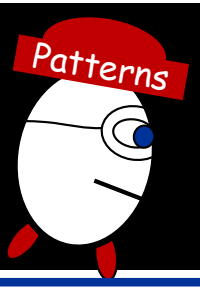
Décorateur *(Decorator)*

Exemple

StarbuzzCoffee *(cf. :Head First Design Patterns)*



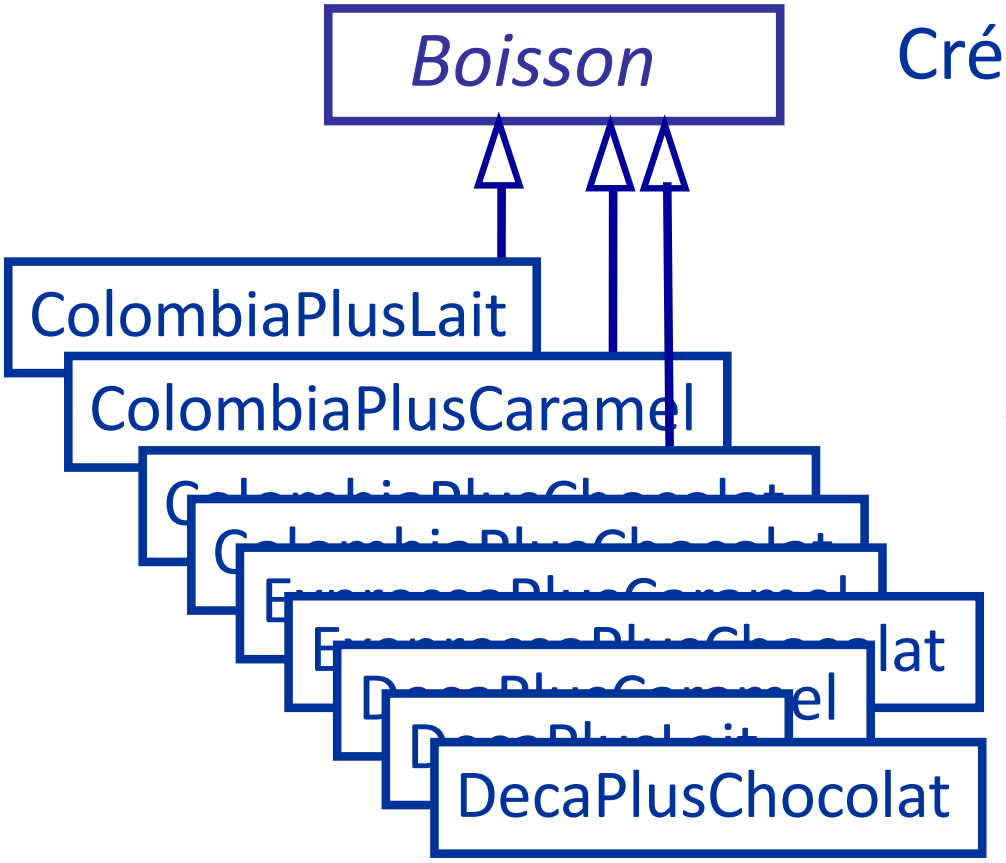
Comment modéliser l’ajout d’ingrédients: lait, caramel, …?



Décorateur *(Decorator)*

Exemple

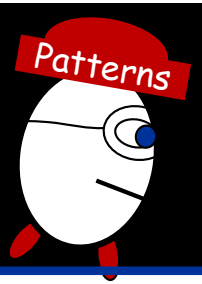
Proposition de solution N°1



Créer des Sous-classes?

Explosion combinatoire

Et Si le prix du lait augmente?
Et si on ajoute un nouvel ingrédient?



Décorateur *(Decorator)*

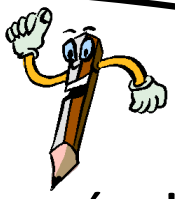
Exemple

Proposition de solution N°2

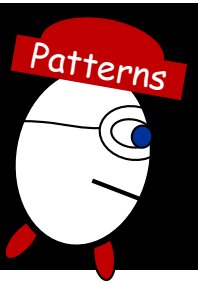
Boisson
description lait: booléen caramel: booléen
getDescription() cout() …

- Ajouter des attributs
- Implémenter cout() dans Boisson
- Surdéfinir cout() dans les sous-classes

Très mauvaise idée!!



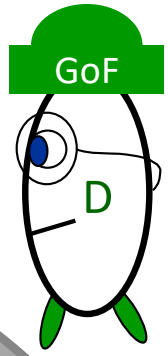
Pensez au principe « Ouvert-Fermé »!



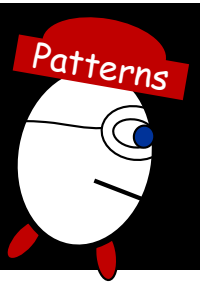
Décorateur *(Decorator)*

Exemple

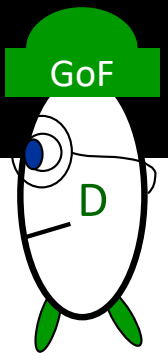
Il faut « décorer » les boissons !



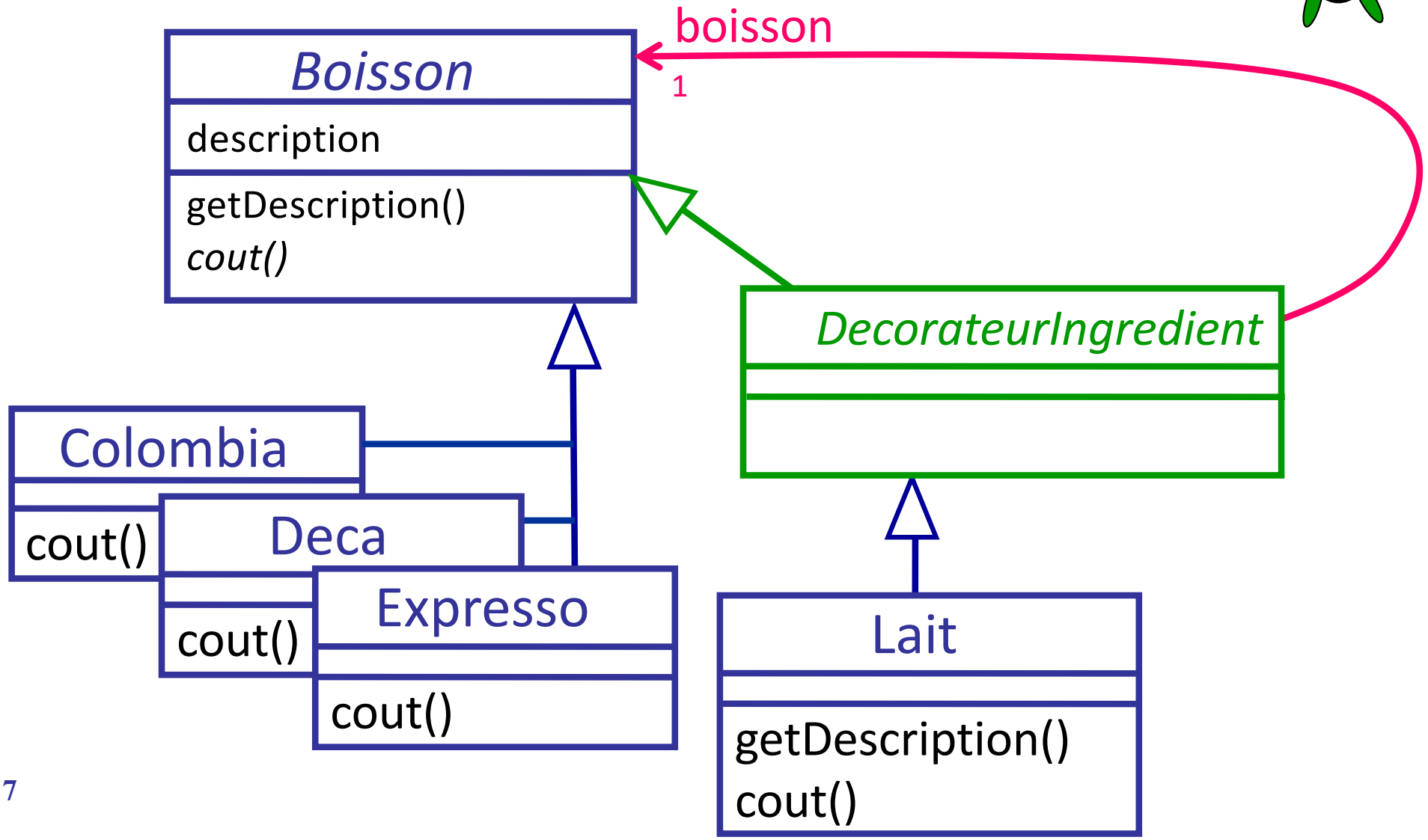
Un expresso lait chantilly= une boisson « décorée »

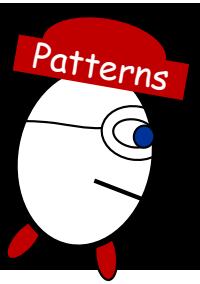


Décorateur *(Decorator)*

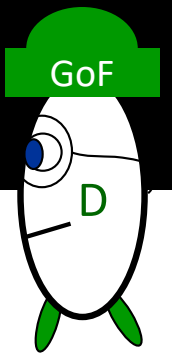


Exemple

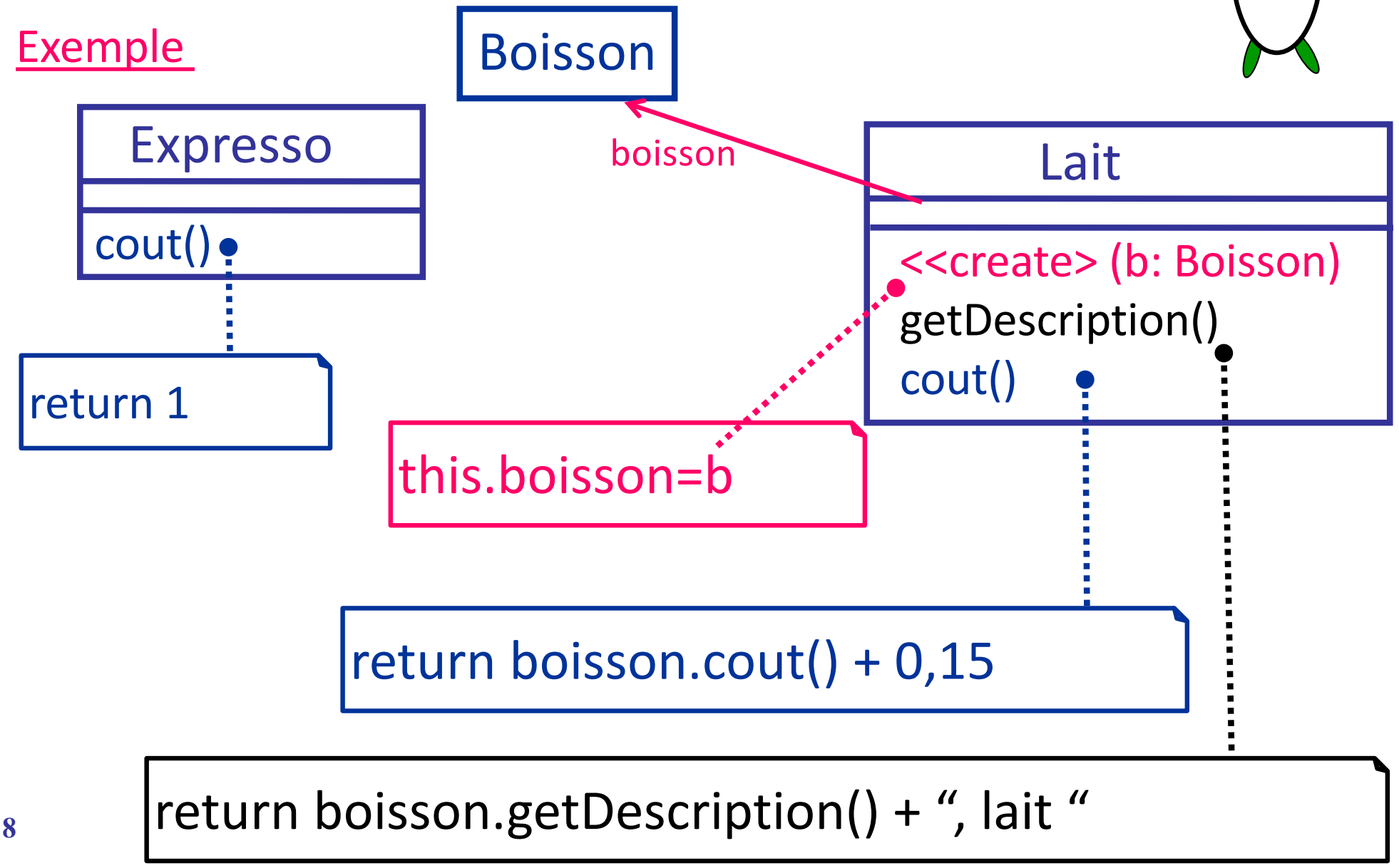


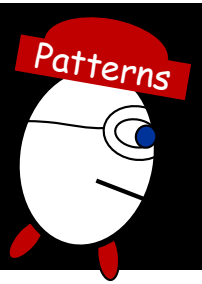


Décorateur *(Decorator)*

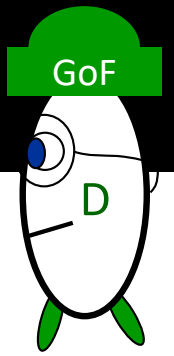


Exemple

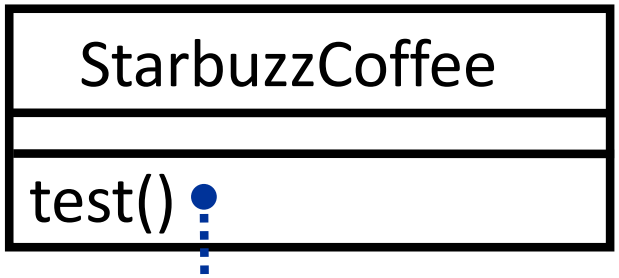




Décorateur *(Decorator)*

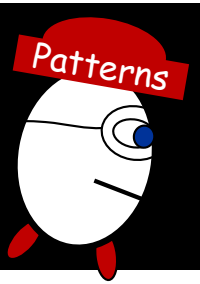


Exemple




```
Boisson b1=new Expresso();
Boisson b2=new Lait(b1);
Boisson b3=new Chantilly(b2);

System.out.println(b3.getDescription() + " : " +
b3.cout() + " €" );
```

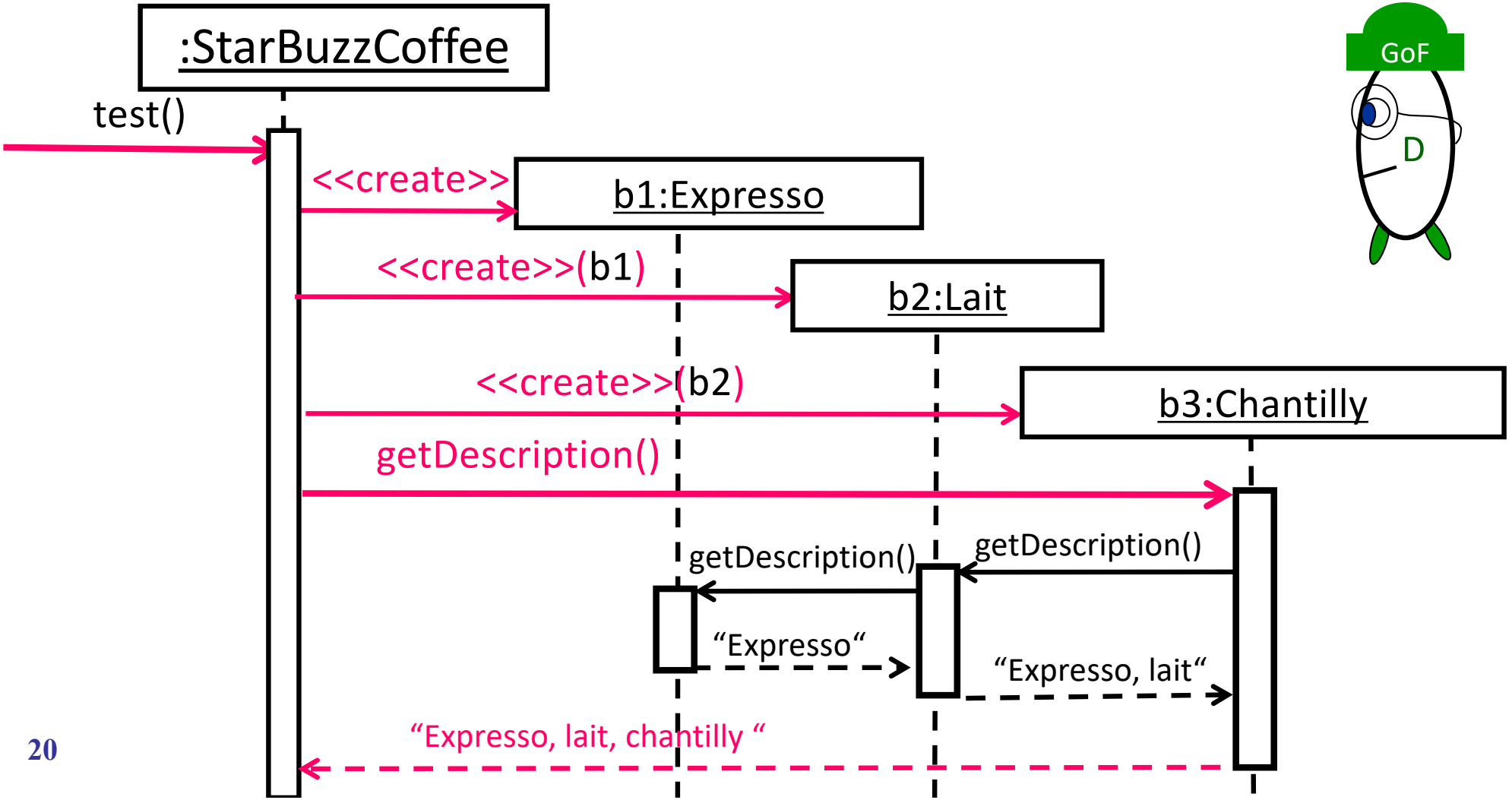
Expresso, lait, chantilly : 1,35 €

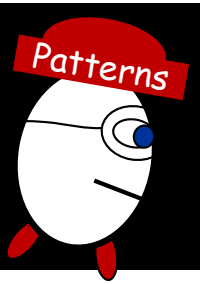


Décorateur *(Decorator)*

Diagramme de séquence

Exécution de la méthode test sur un objet StarBuzzCoffe

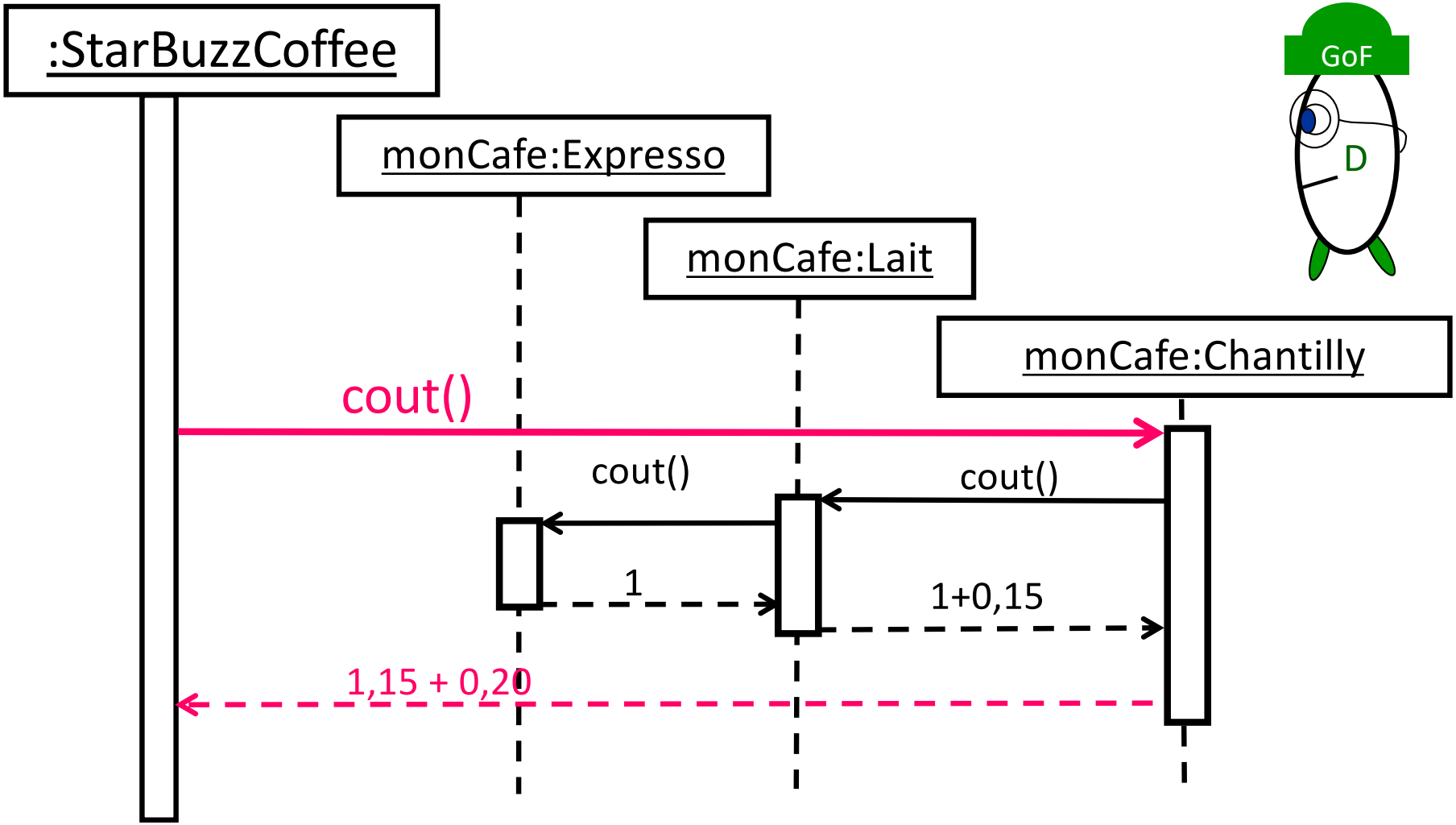


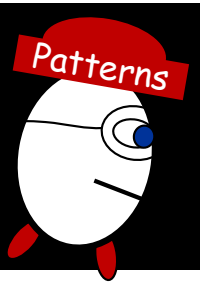


Décorateur *(Decorator)*

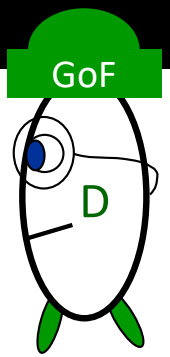
Diagramme de séquence (suite)

Exécution de la méthode test sur un objet StarBuzzCoffe

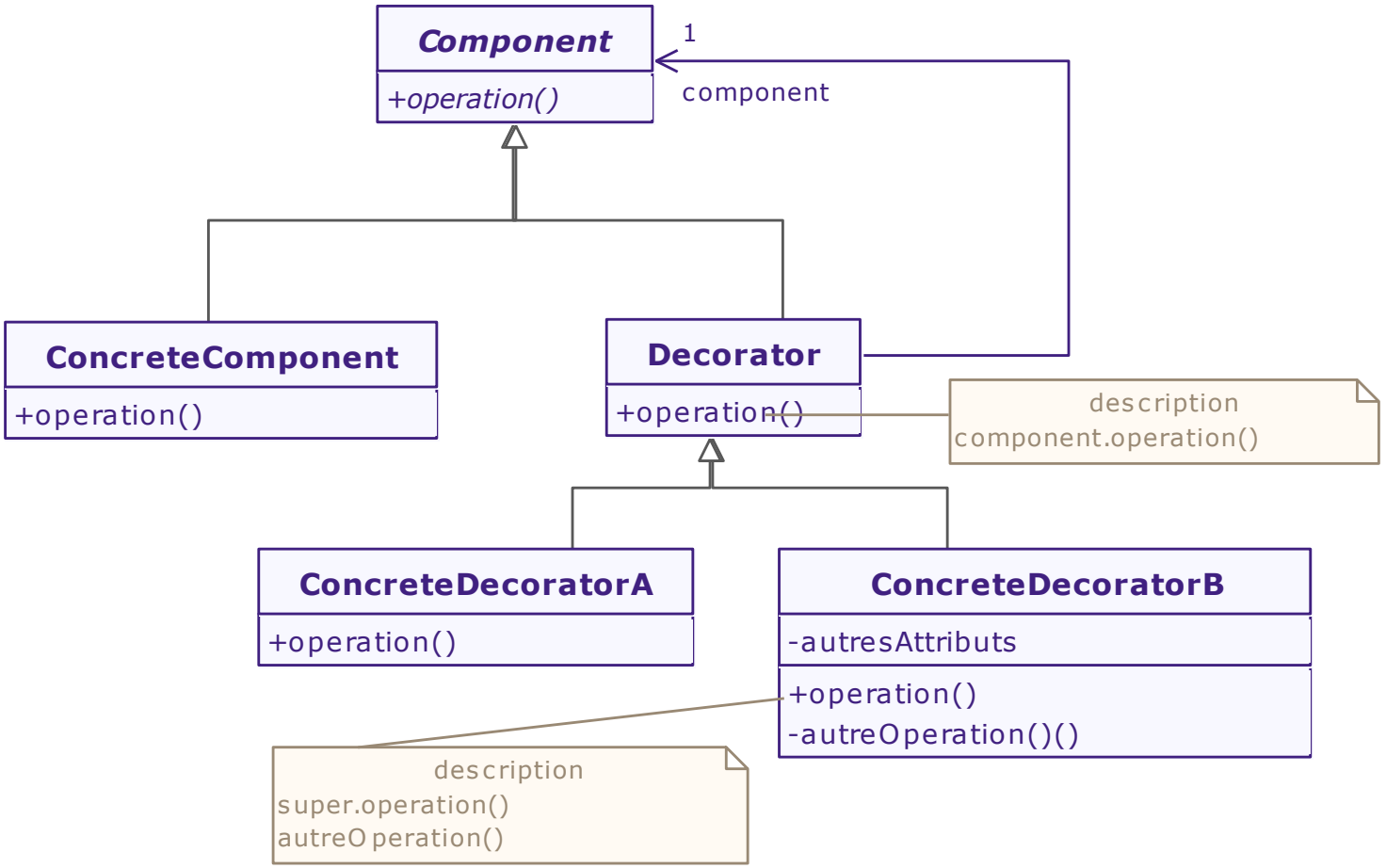


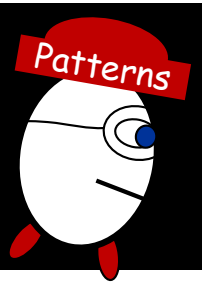


Décorateur *(Decorator)*

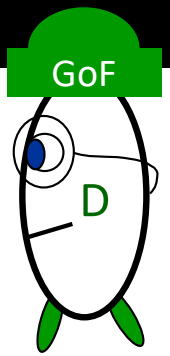


Solution (cas général)





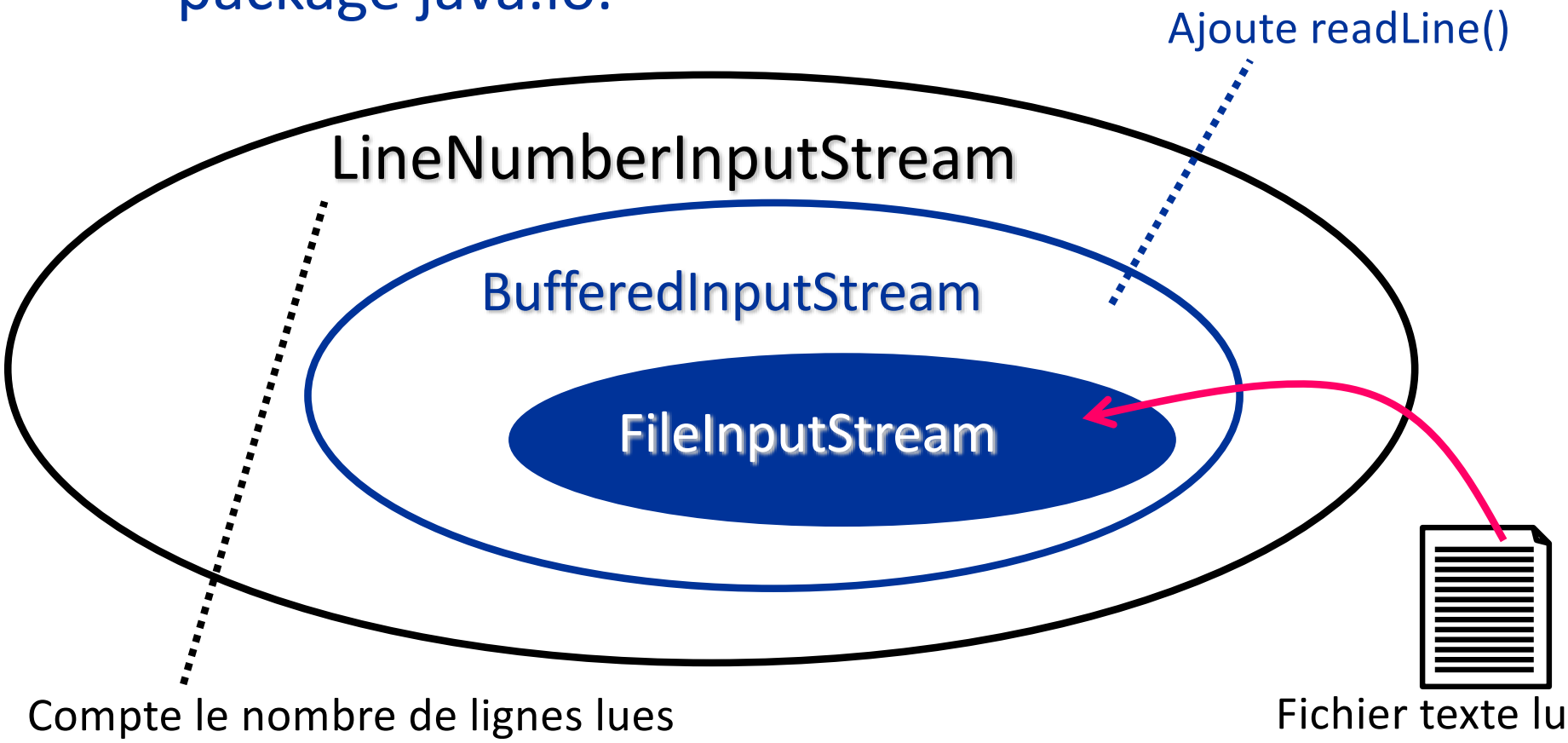
Décorateur *(Decorator)*

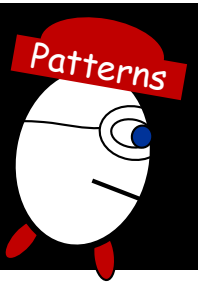


Exemple

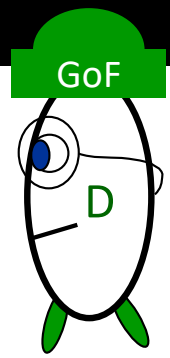
E/S en Java

Décorateur est utilisé dans l'API Java du package java.io.

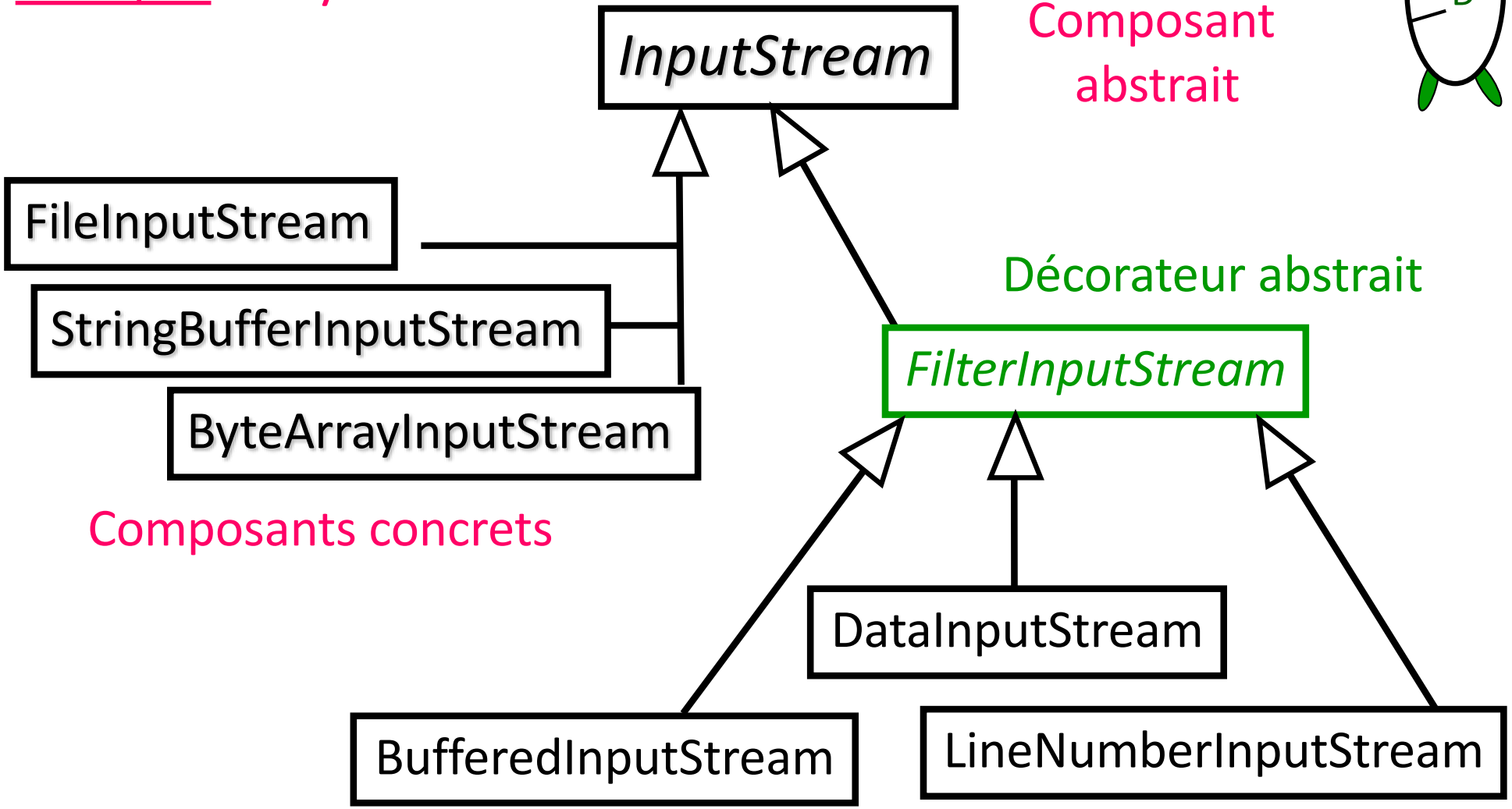


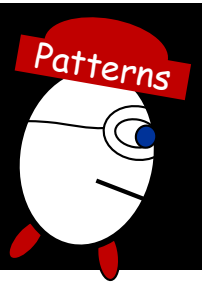


Décorateur *(Decorator)*



Exemple E/S en Java



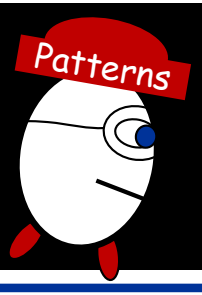


Décorateur *(Decorator)*

Conséquences (inconvénients)

- Décorateur augmente la complexité du code nécessaire à l'instanciation d'un composant: il faut envelopper puis envelopper …

- Le diagramme de classe obtenu peut-être difficile à comprendre: nombreuses petites classes.

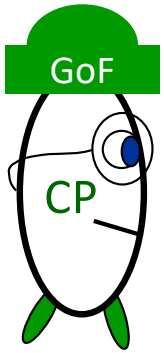


Composite

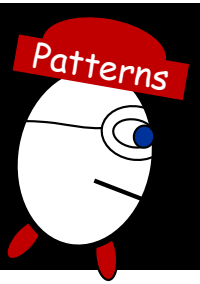
Problème

Comment traiter une structure hiérarchique d'objets (objet composite) de la même façon qu'un objet atomique (composant)?

Solution

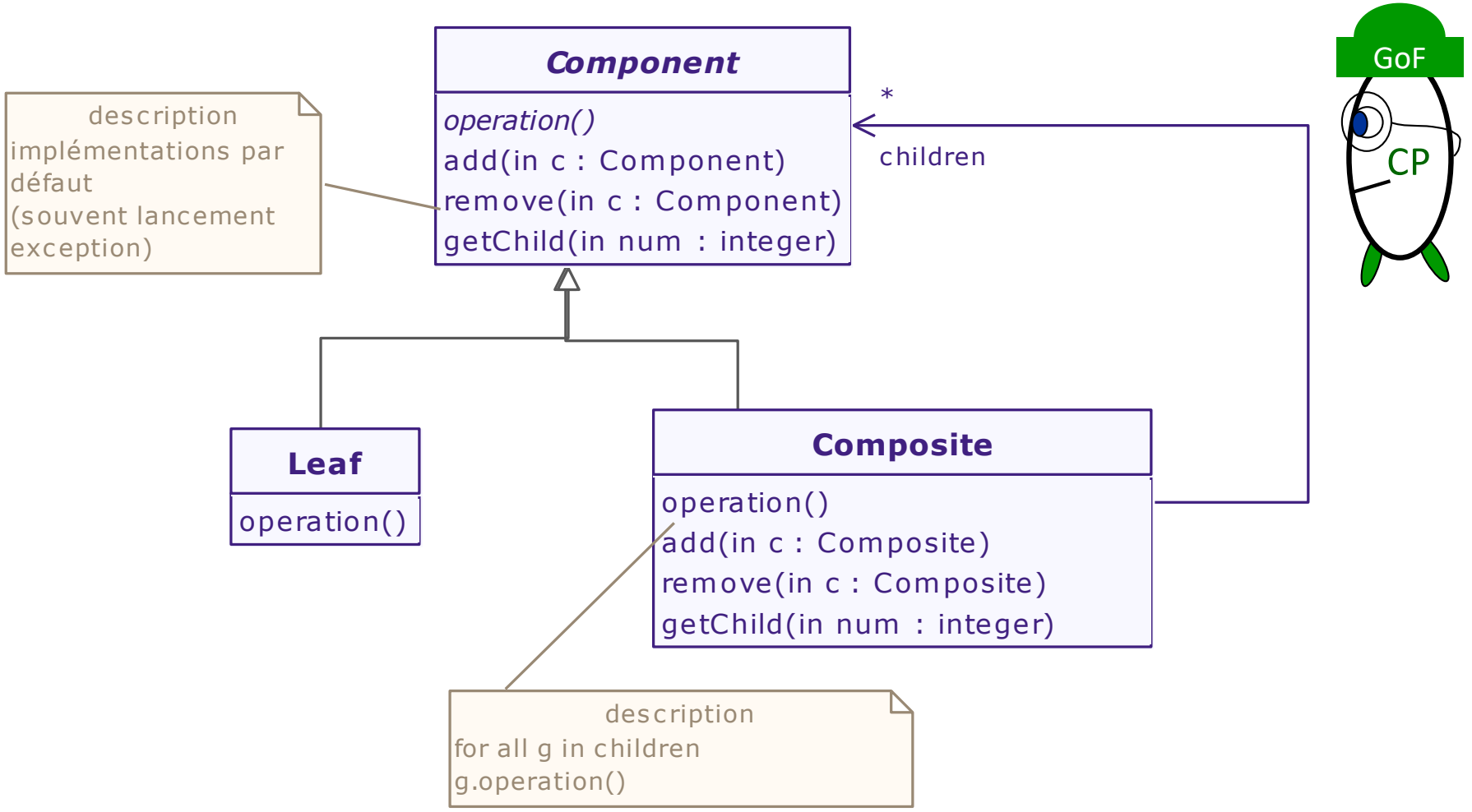


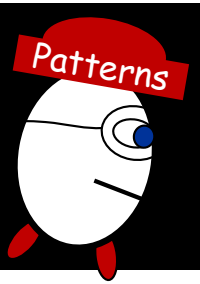
Définir des classes pour les objets composites et atomiques de façon à ce qu'elles héritent d'une même classe.



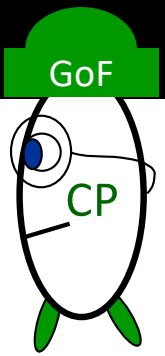
Composite

Solution (cas général)

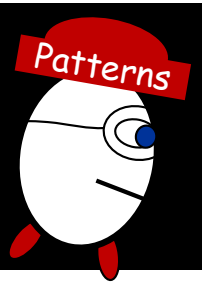




Composite

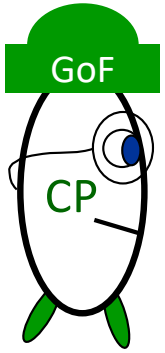


Exemple

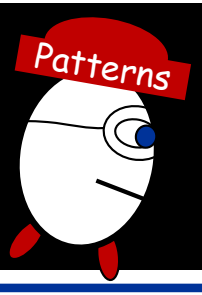


Composite

Conséquences



- Composite simplifie la tâche du client et facilite l'ajout de nouveaux types de composants.
- La définition de contraintes sur la structure d'un composite peut s'avérer difficile.
- Le pattern Itérateur est souvent utilisé pour parcourir les composites.



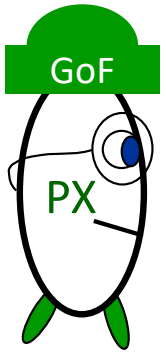
Procuration « Proxy »

Problème

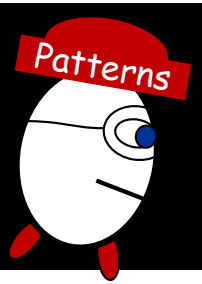
Comment permettre le contrôle de l'accès à un objet ?

- Objet distant
- Vérification de droits d'accès
- Objet dont la création est «coûteuse».

Solution

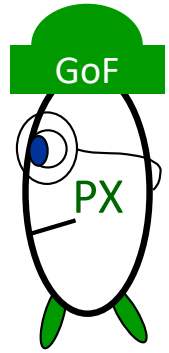


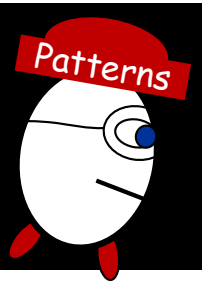
Définir un objet « remplaçant » (le proxy) qui implémente la même interface que le sujet et a la responsabilité de contrôler l'accès à ce dernier.



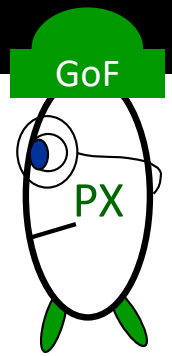
Procuration « Proxy »

Solution (cas général)





Procuration « Proxy »

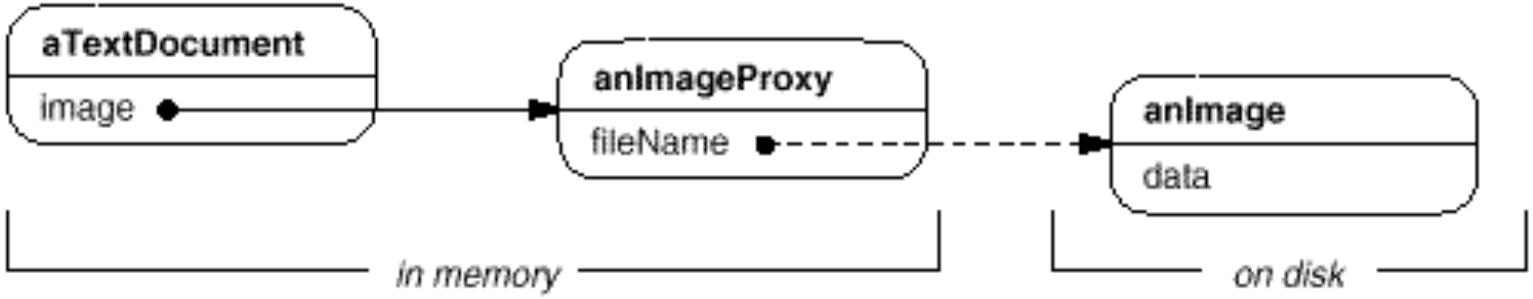


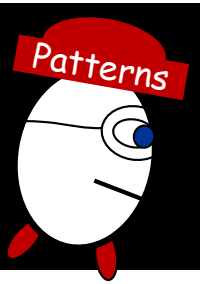
Exemple (problème)

Editeur de documents textes.

Certains objets graphiques volumineux tels que des images ne doivent pas être créés lors de l'ouverture du document.

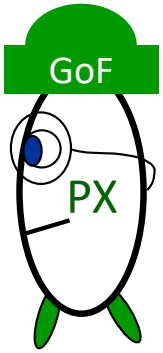
Ils sont remplacés par des «imagesProxy » et ne seront créées que si leur visualisation est nécessaire.

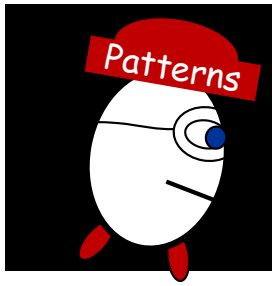




Procuration « Proxy »

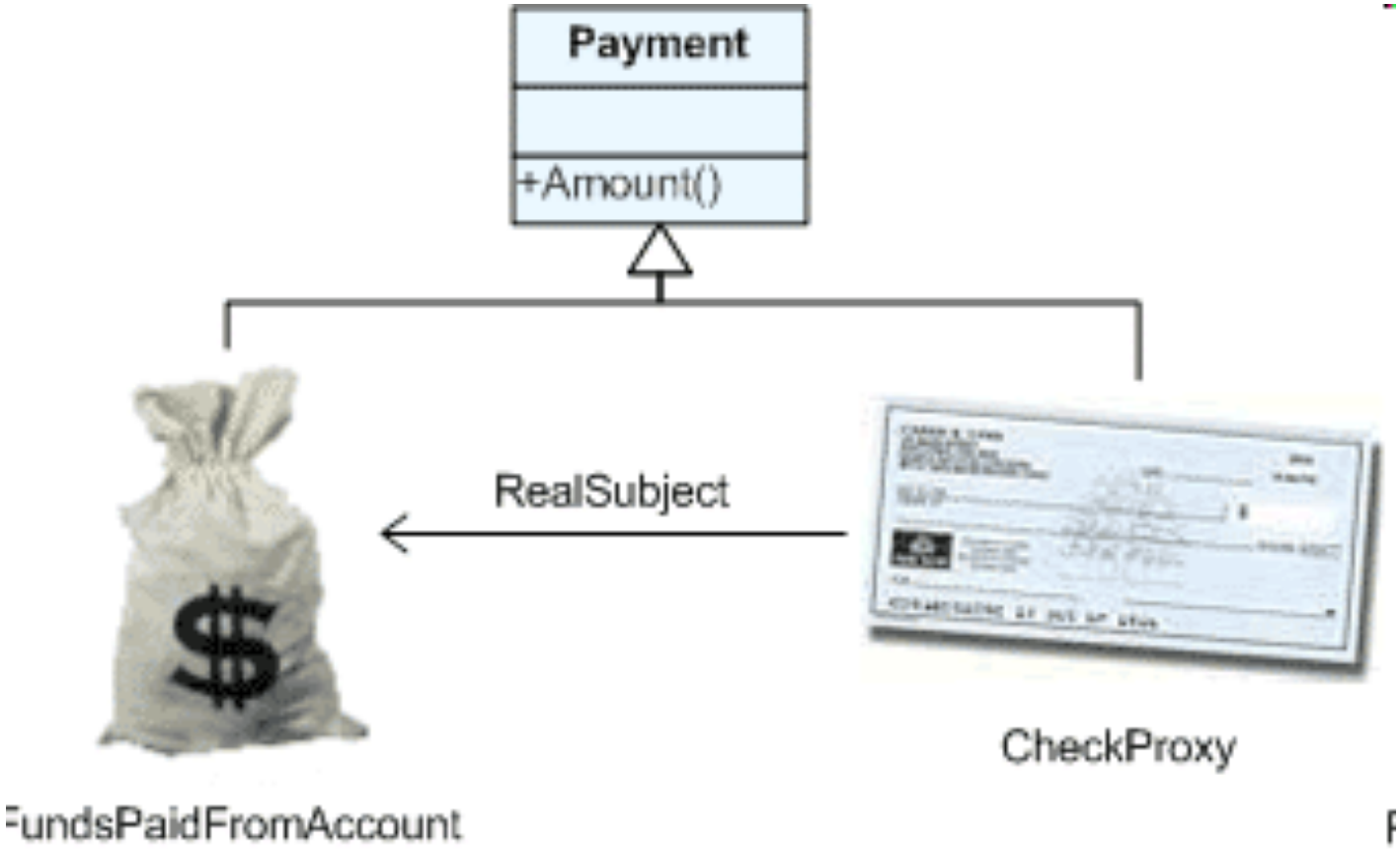
Solution (exemple de proxy virtuel)



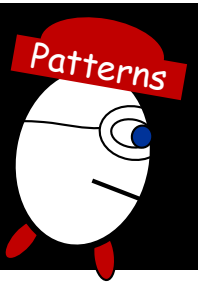


Proxy

➔Exemple

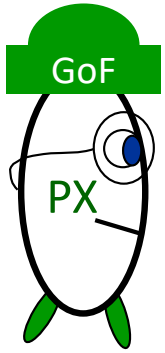


Source: "Non-software examples of software design patterns", *Object Magazine*, Jul 97



Procuration « Proxy »

Conséquences



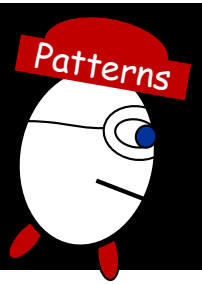
On distingue 3 types de proxy:

Proxy virtuel: Le sujet réel est créé « à la demande » (optimisation).

Proxy distant: Le sujet réel s'exécute sur une machine distante.

Proxy de protection: Le sujet réel n'est accessible que de manière conditionnelle (droits d'accès).

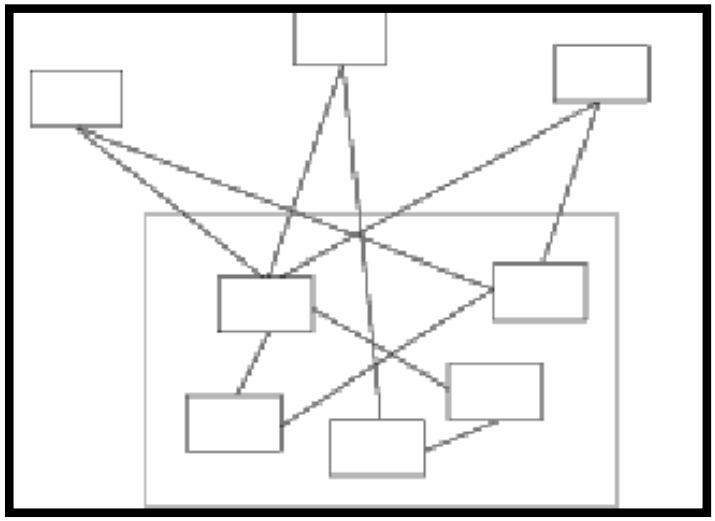
Retour Plan

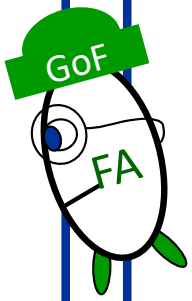


Facade

Problème

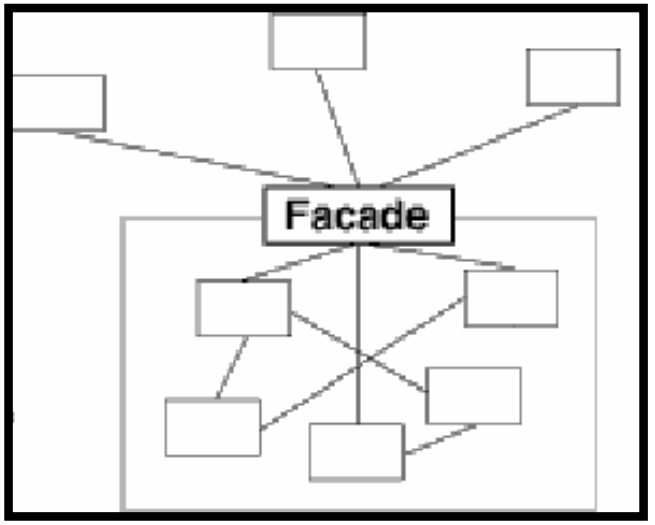
Comment définir une interface unique pour un ensemble disparate de composants (classes/interfaces)?

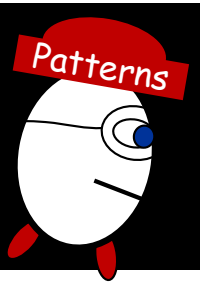




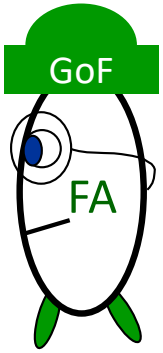
Solution

Définir une classe façade qui présente une interface unifiée qui collabore avec les différents composants.

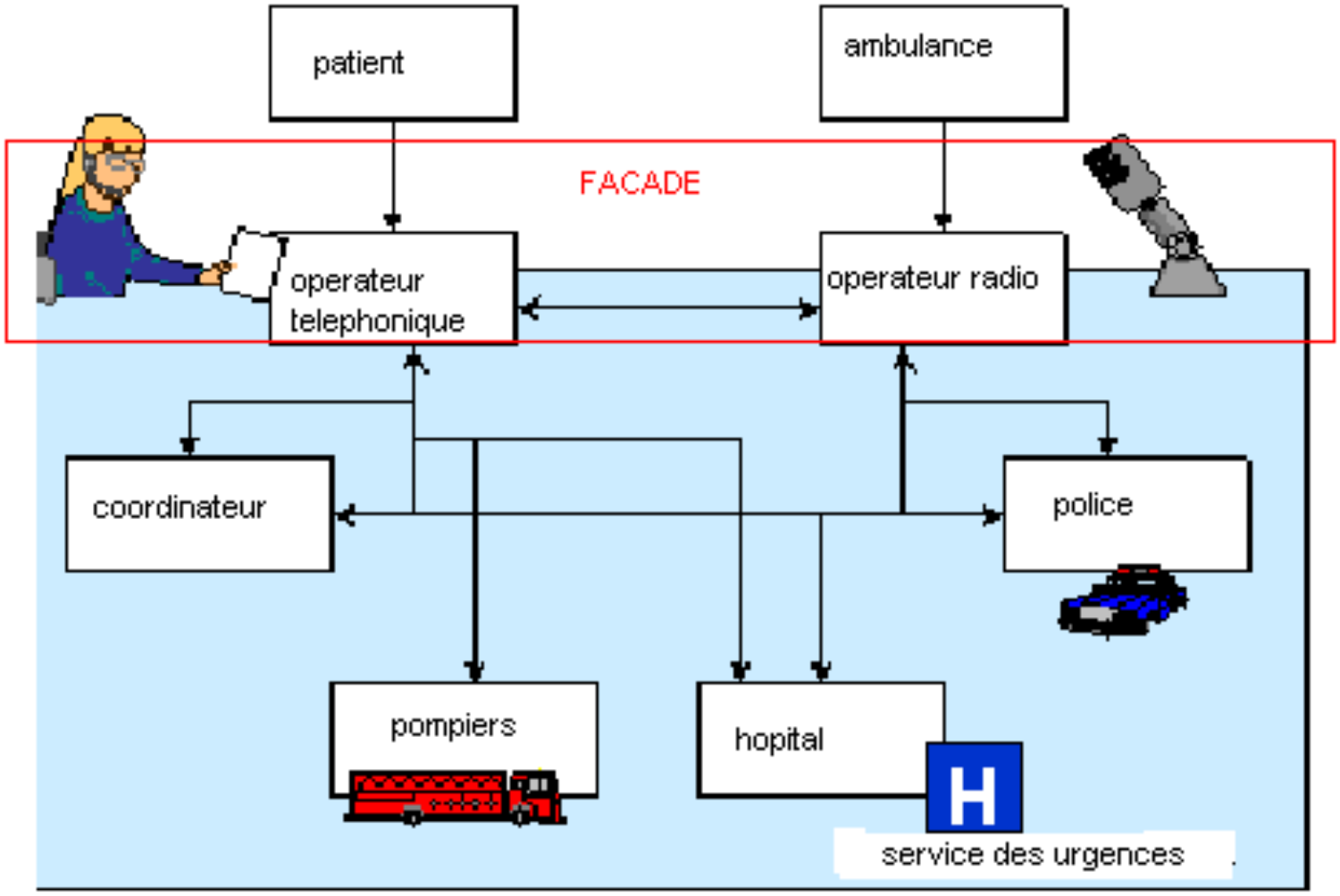


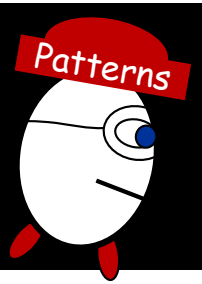


Facade



Exemple

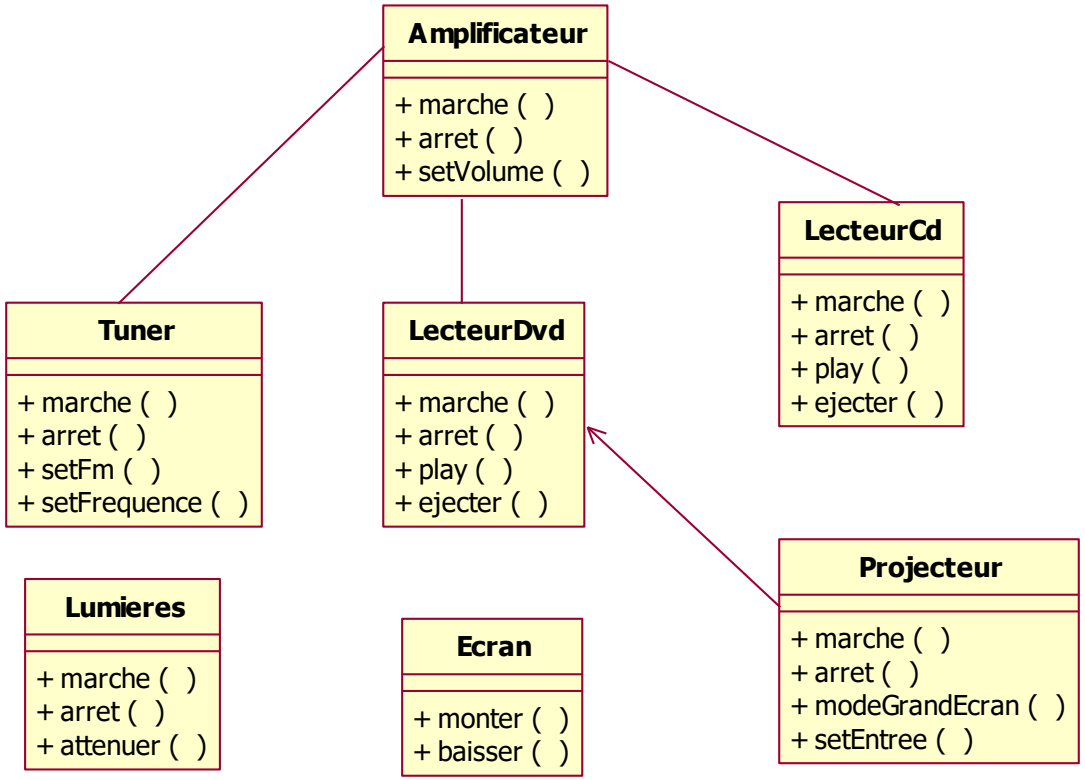




Facade

Exemple (problème)

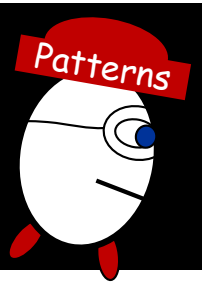
Sous-système home cinéma



Client

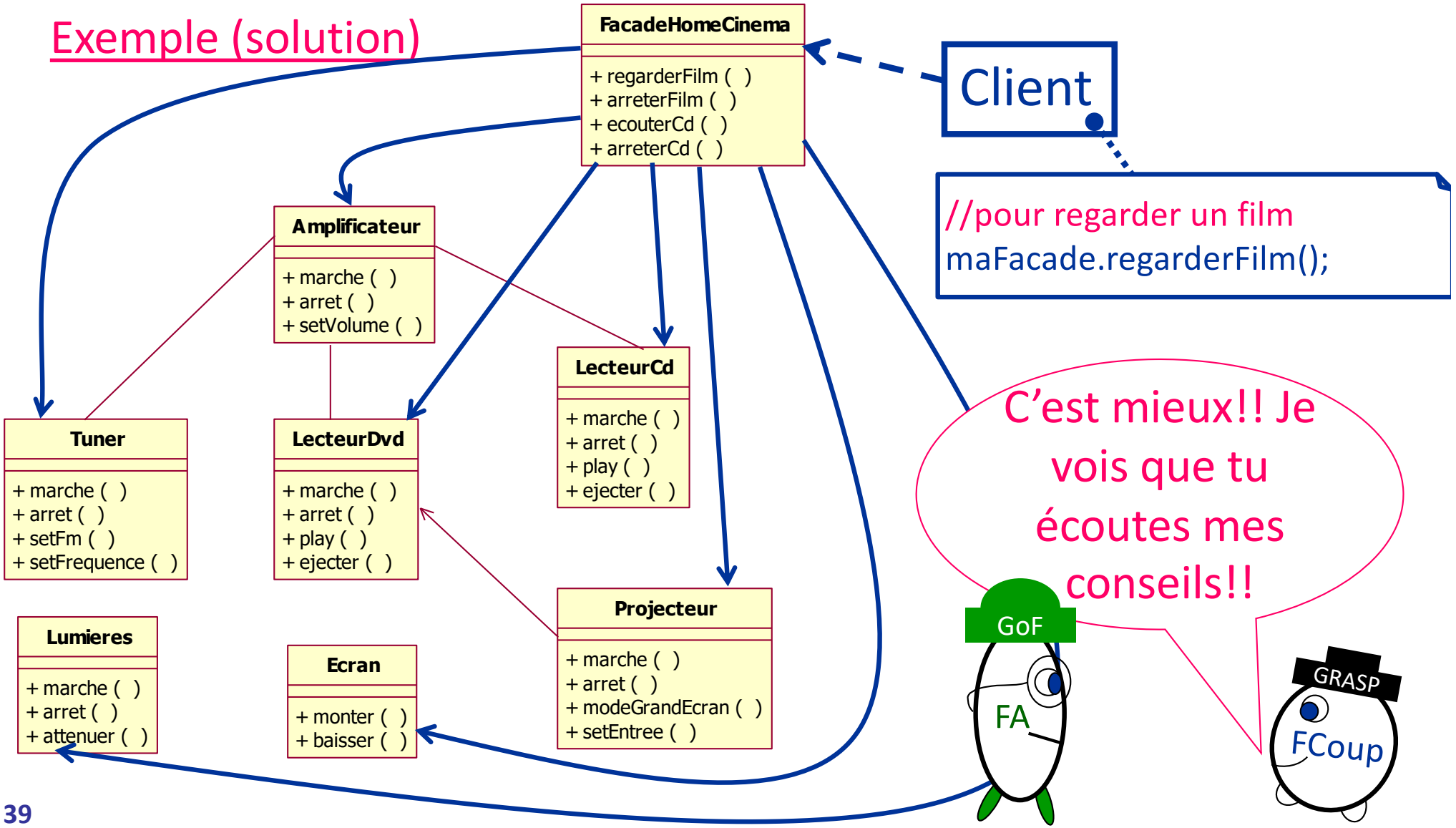
```
//pour regarder un film

lumieres.attenuer(10);
projecteur.marche()
projecteur.setEntree(dvd);
projecteur.modeGrandEcran();
amp.marche();
amp.setDvd(dvd);
amp.setVolume(5);
ecran.baisser()
dvd.marche();
dvd.play();
```

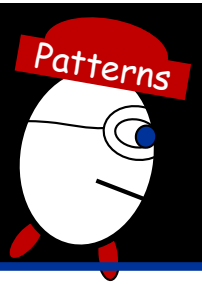
Avec combien de classes le client interagit-il?



Facade

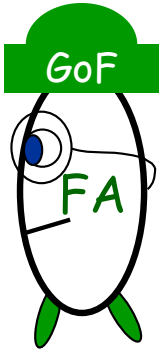
Exemple (solution)

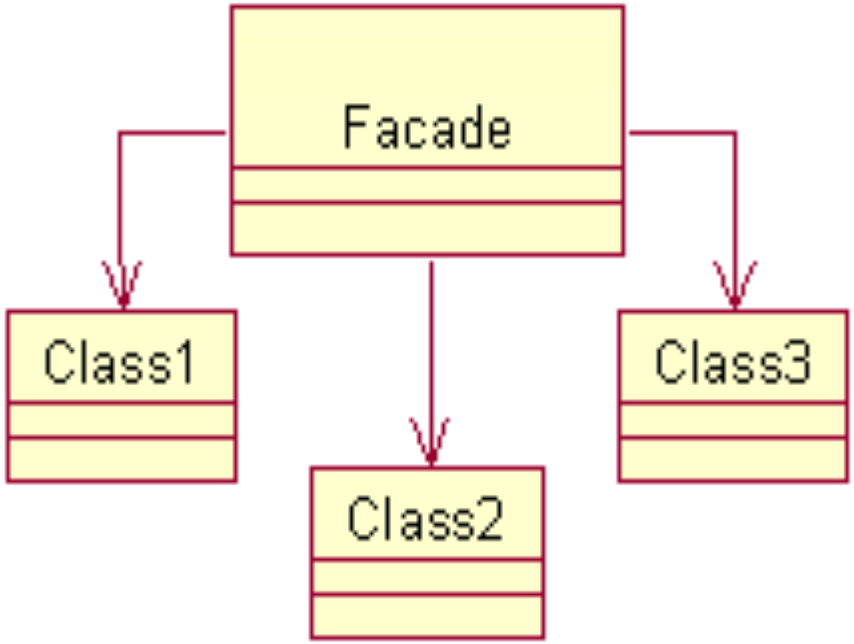


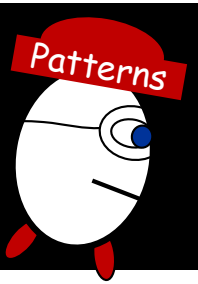


Facade

Solution (cas général)

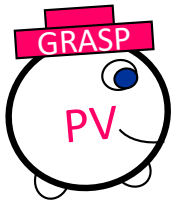


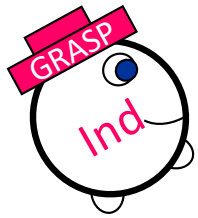




Facade

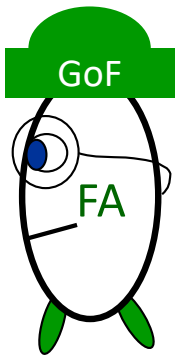
Conséquences

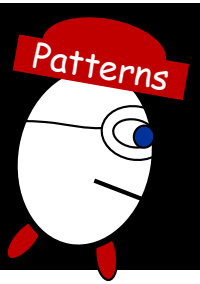




- L'objet façade assure la protection des variations des différents composants en ajoutant une Indirection qui assure le faible Couplage.

- On accède très souvent à une façade via un Singleton.



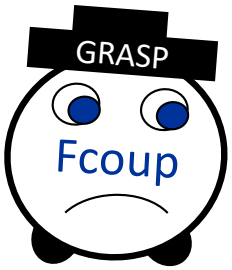


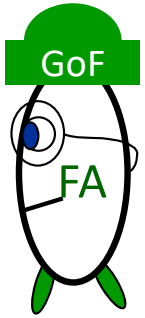
Facade

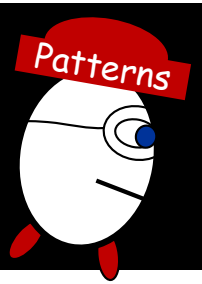
Conséquences

- L'objectif principal de Façade est de fournir une interface de plus haut niveau qui rend le sous-système plus facile à utiliser.

- On peut utiliser une façade tout en offrant la possibilité au client d'accéder aux classes du sous-système.







3 – Patterns Structurels

Résumé

DECORATOR	Enveloppe un objet pour offrir un autre comportement
FACADE	Simplifie l'interface d'un ensemble de classes
PROXY	Enveloppe un objet pour en controler l'accès
COMPOSITE	Permet aux clients de traiter les collections et les objets individuels de la même manière
ADAPTER	Enveloppe un objet et fournit une interface différente pour y accéder