# Design Pattern Decorator

http://www.tutorialspoint.com/design_pattern/decorator_pattern.htm

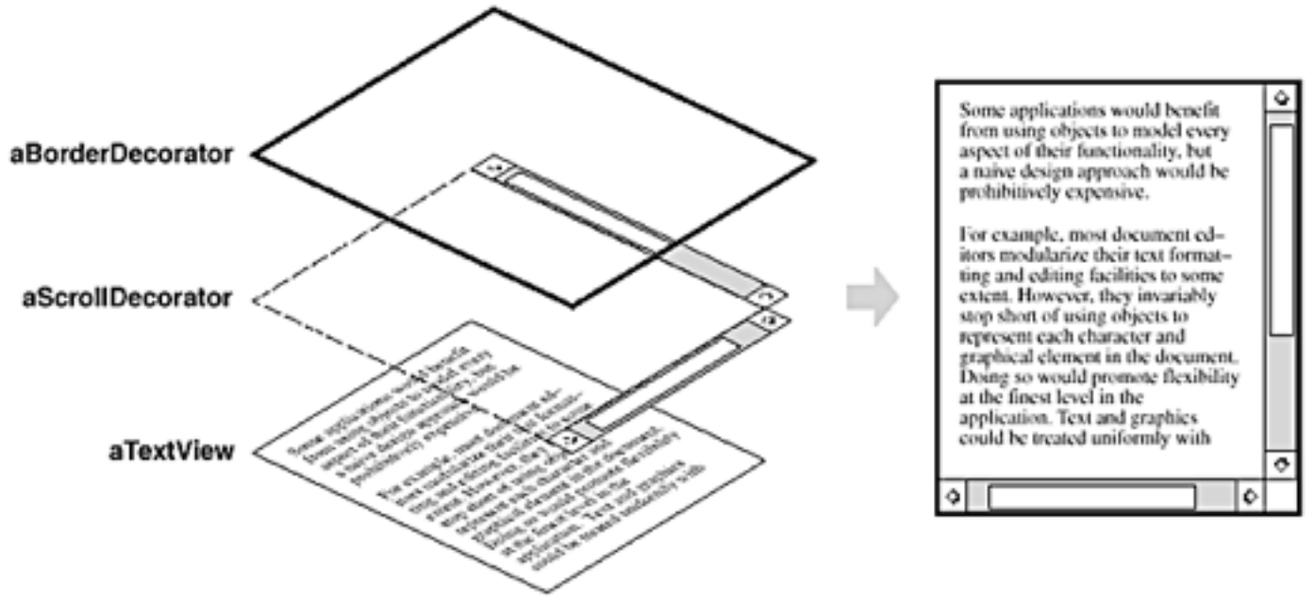
Traité à partir des codes

# Patron Décorateur : le problème

- Il doit être possible d'ajouter dynamiquement des fonctionnalités à des objets.
- Le nombre de sous classes serait très grand si on devait définir autant de sous-classes que de variantes d'une classe ou bien elles doivent évoluer.

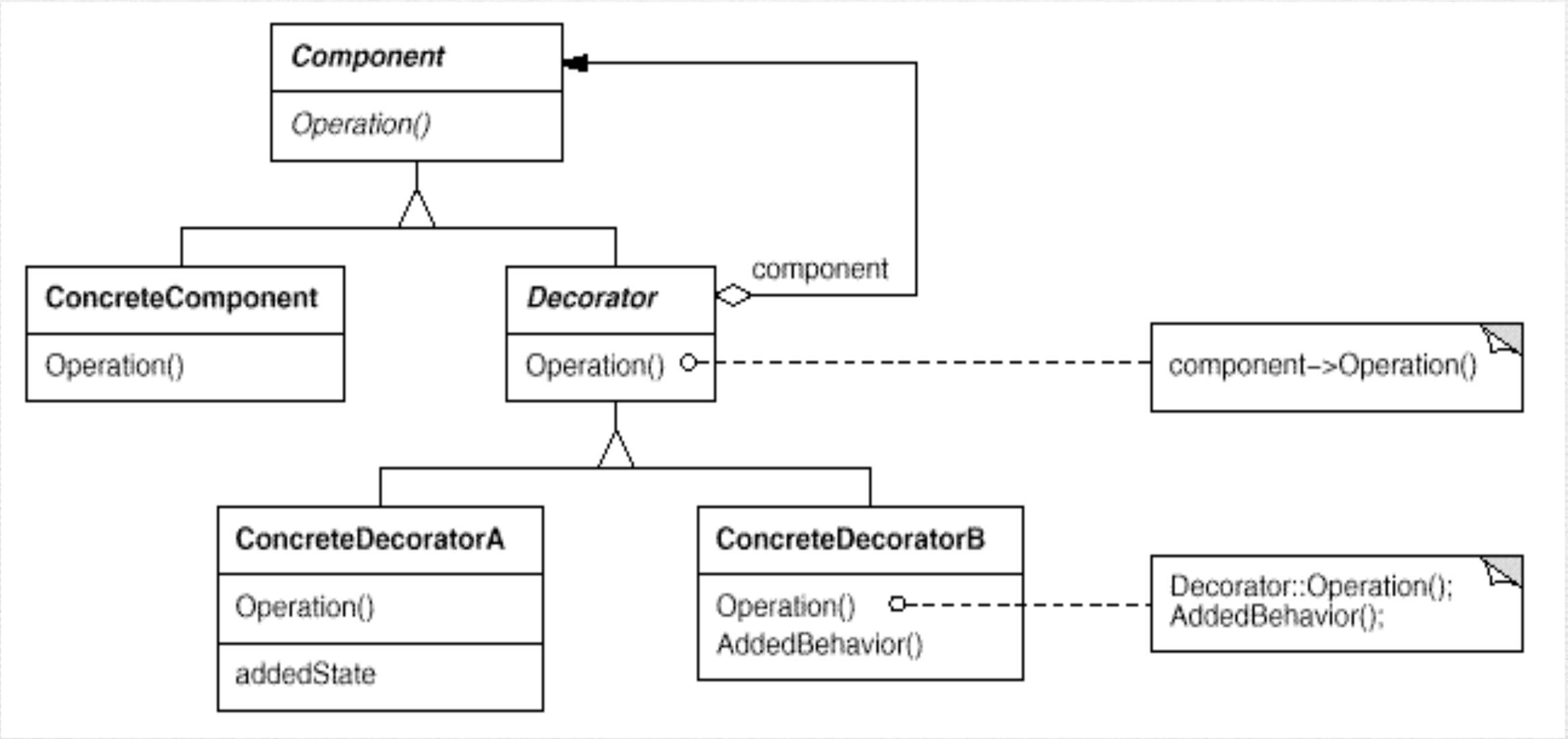
# Motivation

- A TextView has 2 features:
  - borders:
    - 3 options: none, flat, 3D
  - scroll-bars:
    - 4 options: none, side, bottom, both




- How many Classes?
  - 3 x 4 = **12** !!!
    - e.g. TextView, TextViewWithNoBorder&SideScrollbar, TextViewWithNoBorder&BottomScrollbar, TextViewWithNoBorder&Bottom&SideScrollbar, TextViewWith3DBorder, TextViewWith3DBorder&SideScrollbar, TextViewWith3DBorder&BottomScrollbar, TextViewWith3DBorder&Bottom&SideScrollbar, ... .....

# Patron Décorateur : la solution



# Patron Décorateur : les rôles

✓Component

➡ définit l'interface des objets auxquels de nouvelles responsabilités peuvent être ajoutées dynamiquement.

✓ConcreteComponent

➡ un objet de base auquel de nouvelles responsabilités peuvent être ajoutées.

✓Decorator

➡définit une interface conforme à l'interface de «Component» ; il maintient une réference vers un objet composant

✓ConcreteDecorator

➡› ajoute des responsabilités à un «component»

# Decorator en action

- On calcule le prix d'un café en fonction des ingrédients qui lui sont ajoutés : lait, sucre, .. en utilisant le pattern décorateur.