

# DESIGN PATTERN COMPOSITE

Introduction aux Design Patterns

Kylian ROUSSEAU  
Julian RAY-CONSTANTY  
Adam SOUAMES  
Barsbold MYANGANBAATAR



# SOMMAIRE

## I. Introduction

- 1.Introduction générale aux Design Patterns
- 2.Un Besoin
- 3.Un Problème

## II. Détails du Pattern

- 1.Définition du Pattern Composite
- 2.Diagramme de classes généralisé
- 3.Rôle des classes participantes
- 4.Lien avec les principes SOLID
- 5.Limites du pattern
- 6.Le Composite et le Décorateur

## III. Exemple et Live-coding

- 1.Second Contexte Métier
- 2.Live-coding (Vidéo)
- 3.Rétro-conception
  - a.Diagramme de Classes
  - b.Diagramme de Séquences

## IV. Conclusion

- 1.Synthèse
- 2.Bibliographie
- 3.QCM

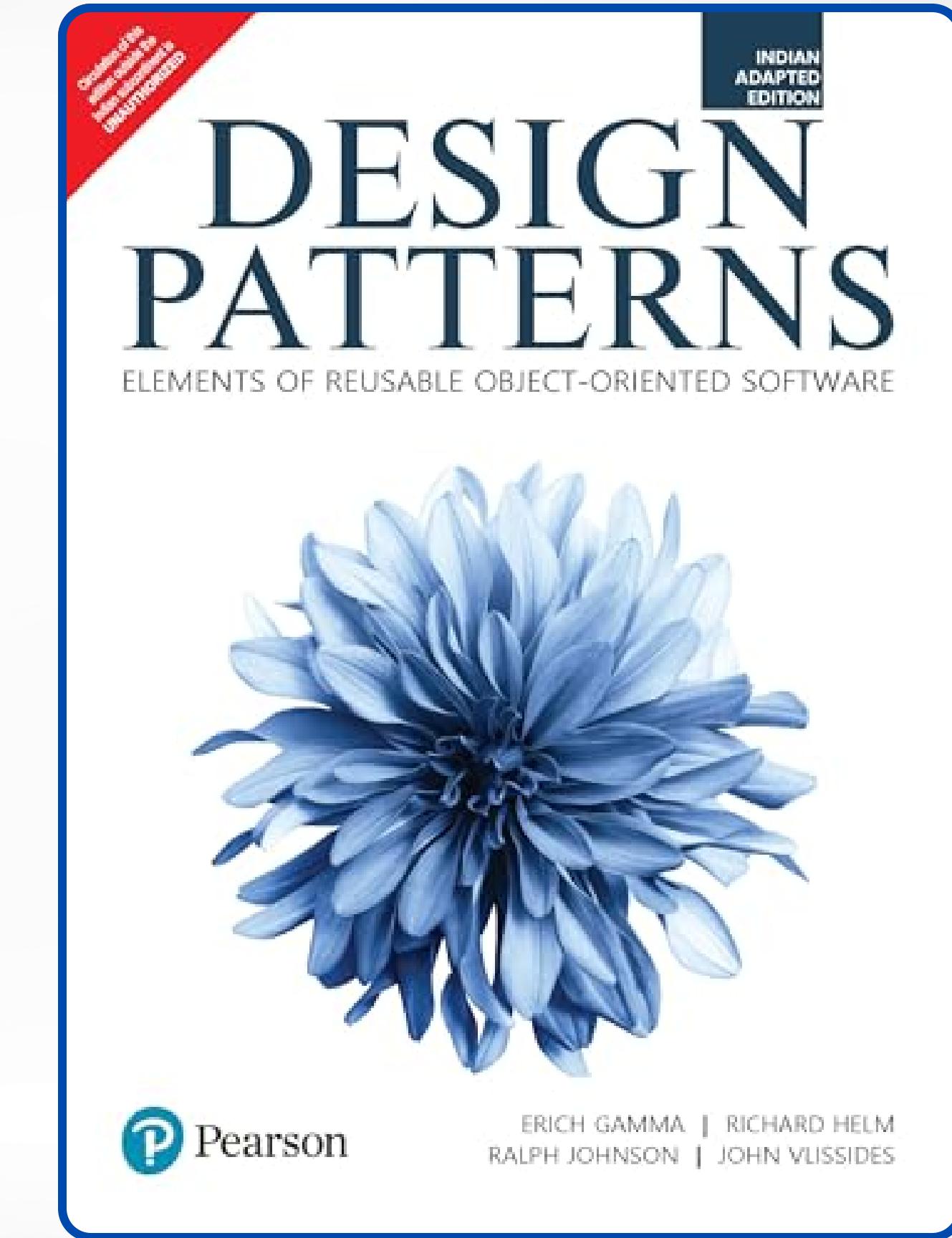


# QU'EST CE QU'UN DESIGN PATTERN ?

Un patron pour réaliser un code de meilleur qualité

Ils permettent de :

- Réutiliser des solutions existantes
- Améliorer la lisibilité/maintenabilité du code
- Faciliter la communication entre les développeurs



# LE BESOIN

## GÉRER UNE STRUCTURE HIÉRARCHIQUE UNIFORME

### L'objectif métier

Développer un logiciel d'exploration de fichiers

Il possède la fonctionnalité de calculer le poids total d'un dossier

### La complexité naturelle

Tâche simple pour un fichier ou dossier

Beaucoup plus complexe sur un dossier de dossiers

Nécessité de faire appel à la récursivité



# LE PROBLÈME

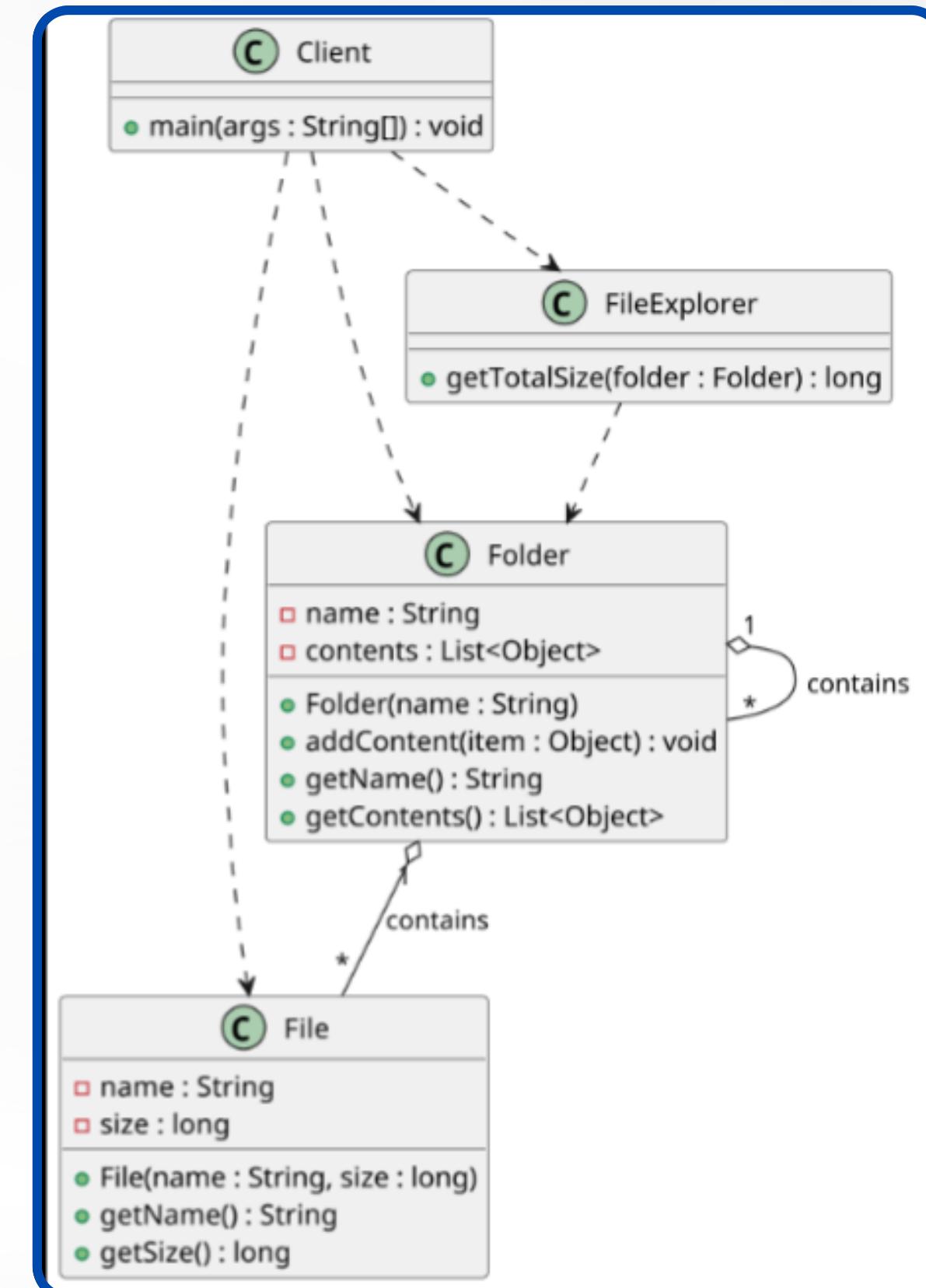
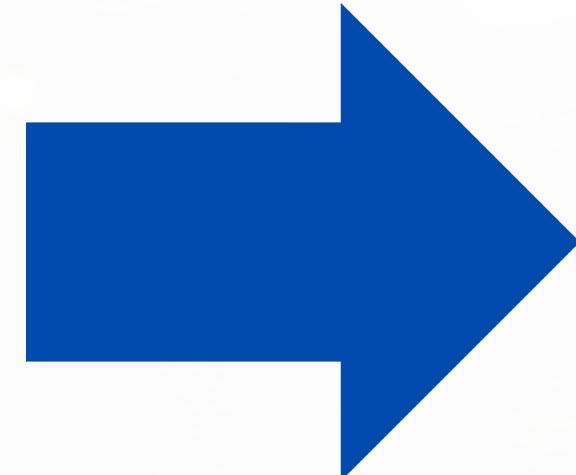
## Un code fragile

Chaque élément traité différemment  
→ Sans cesse vérifier son type  
→ Multiplie les boucles et conditions

## Les conséquences

- Un couplage fort
- Maintenance difficile
- Fermé aux extensions

Pour résoudre cela → Composite



# Les détails du Design Pattern

## COMPOSITE



# QU'EST CE QUE LE DESIGN PATTERN COMPOSITE ?

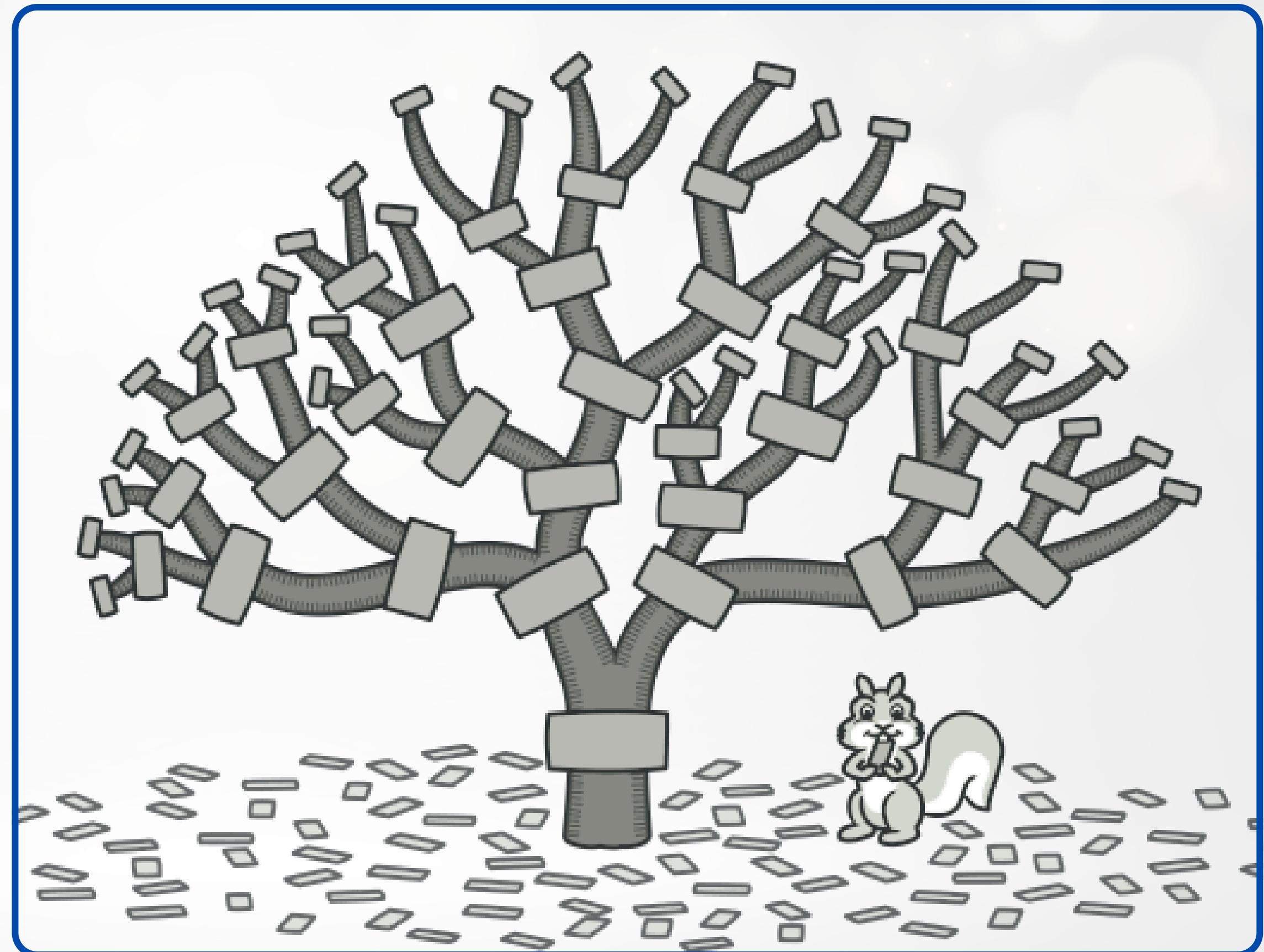
Un pattern structurel du GoF

## Intention :

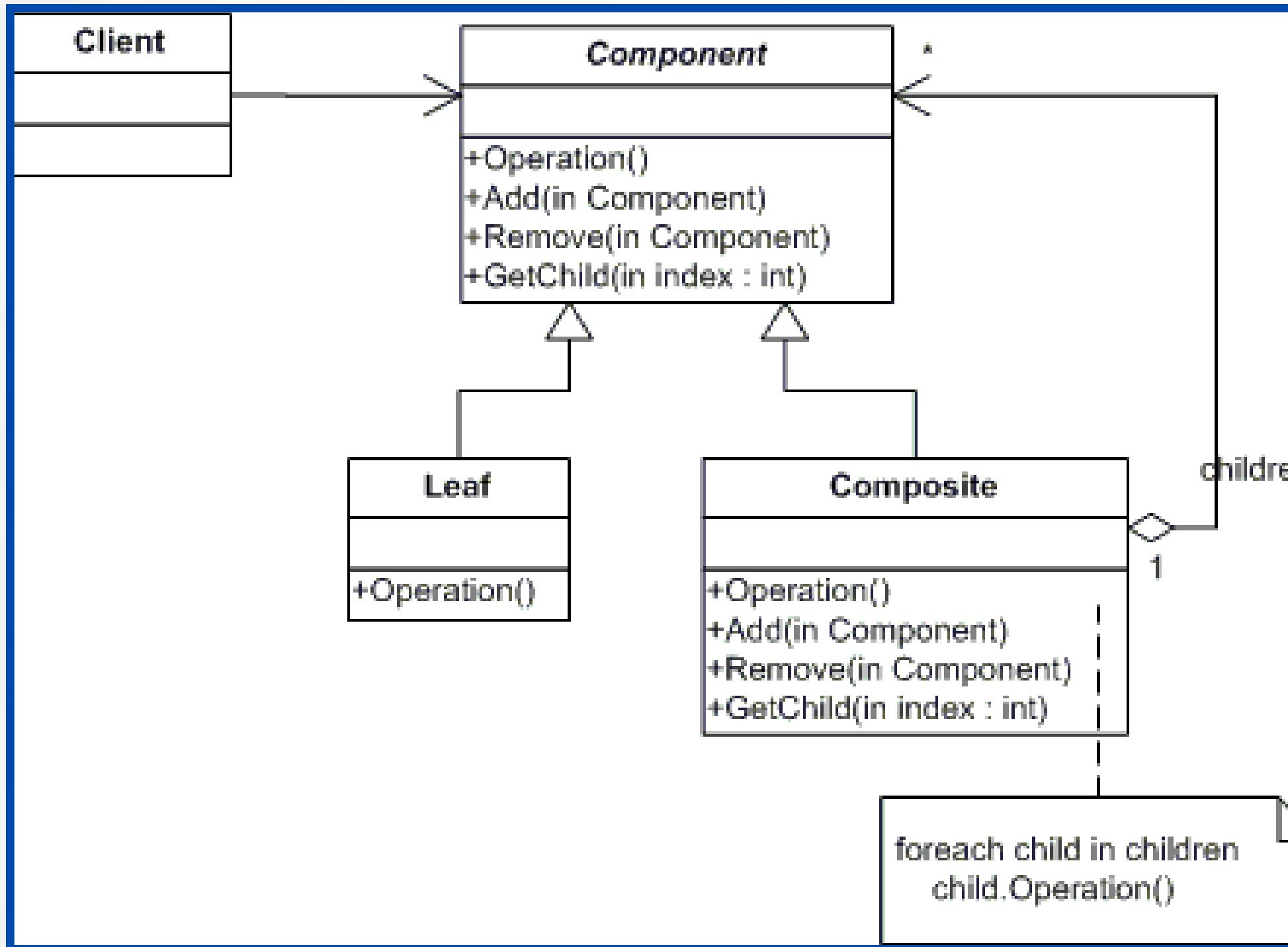
- Composer des objets en structure arborescente
- Traiter les objets individuels et compositions de manière uniforme

## Problématique :

- Permet au client d'interagir avec éléments simples/complexes avec une même interface
- Elimine le besoin de vérifier le type de chaque objet



# DIAGRAMME DE CLASSES GÉNÉRIQUE



## Classes participantes

Component (Composant)

Leaf (Feuille)

Composite (Conteneur)

Client



# LIEN AVEC LES PRINCIPES **SOLID**

**S.O.L.I.D.**



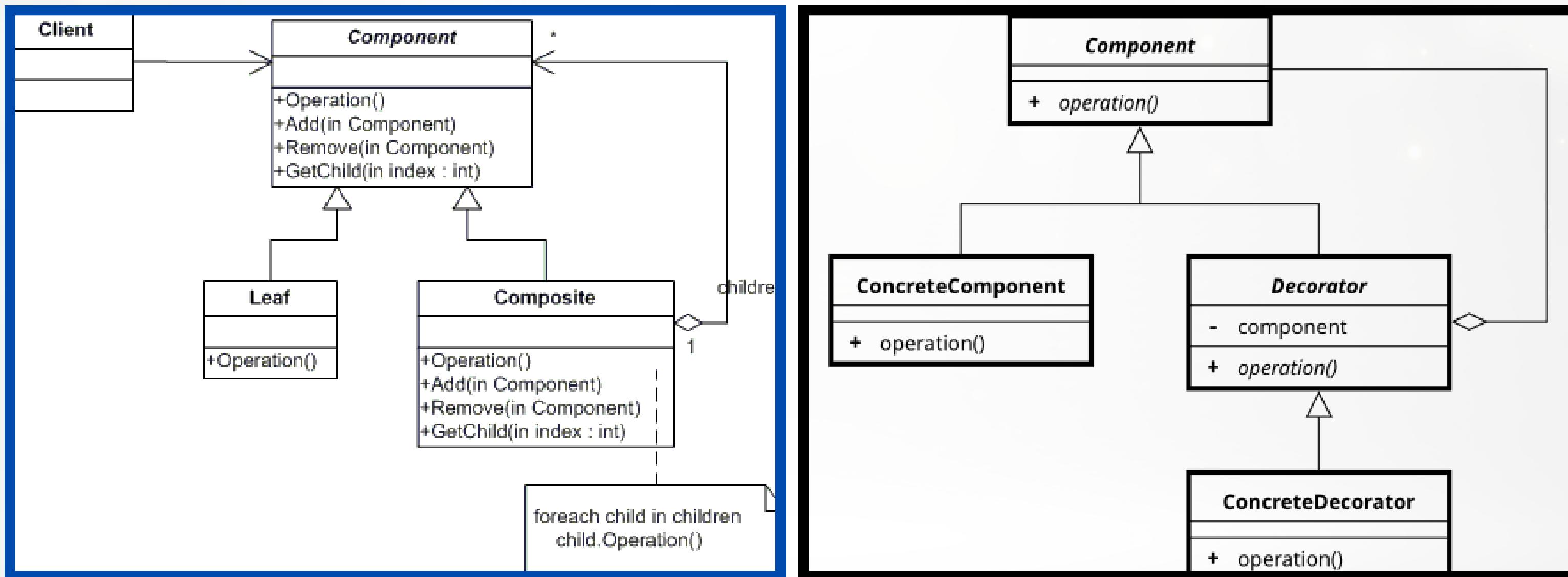
# LES LIMITES DU **PATTERN COMPOSITE**

**Complexité de  
l'interface**

**Sécurité et  
Transparence**



# LE COMPOSITE ET LE DÉCORATEUR



# EXAMPLE ET LIVE CODING



# UN DEUXIÈME EXEMPLE

1

## Component

L'interface IPersonnel

Possède la méthode getSalary()

2

## Leaf

La classe Employee

Un simple employé

3

## Composite

La classe Department

Contient des départements/employés

La gestion du personnel dans une entreprise

Comment calculer le salaire d'un département, y compris ses salariés et sous-départements de manière uniforme ?



# LIVE CODING

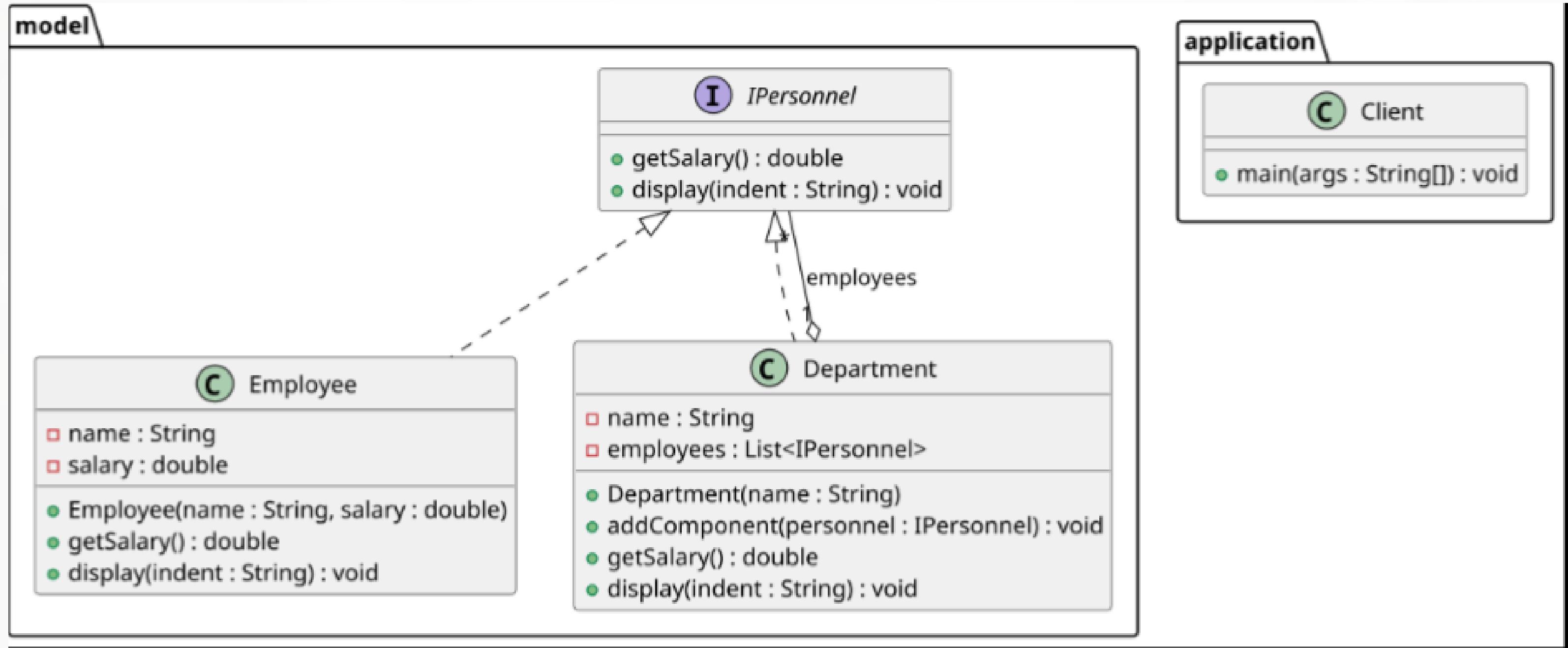
The screenshot shows a Java code editor with the title "Exemple d'utilisation du Design Pattern Composite". The code implements the Composite pattern for managing personnel in a company. The class `Department` implements the `IPersonnel` interface and contains methods for adding components, calculating total salary, and displaying personnel details.

```
public class Department implements IPersonnel {    private List<IPersonnel> employees = new ArrayList<>();  
  
    public Department(String name){        this.name = name;  
    }  
  
    public void addComponent(IPersonnel personnel){        employees.add(personnel);    }  
  
    @Override public double getSalary(){        double totalSalary = 0;        for(IPersonnel personnel : employees){            totalSalary += personnel.getSalary();        }  
        return totalSalary;    }  
  
    @Override public void display()  
}
```

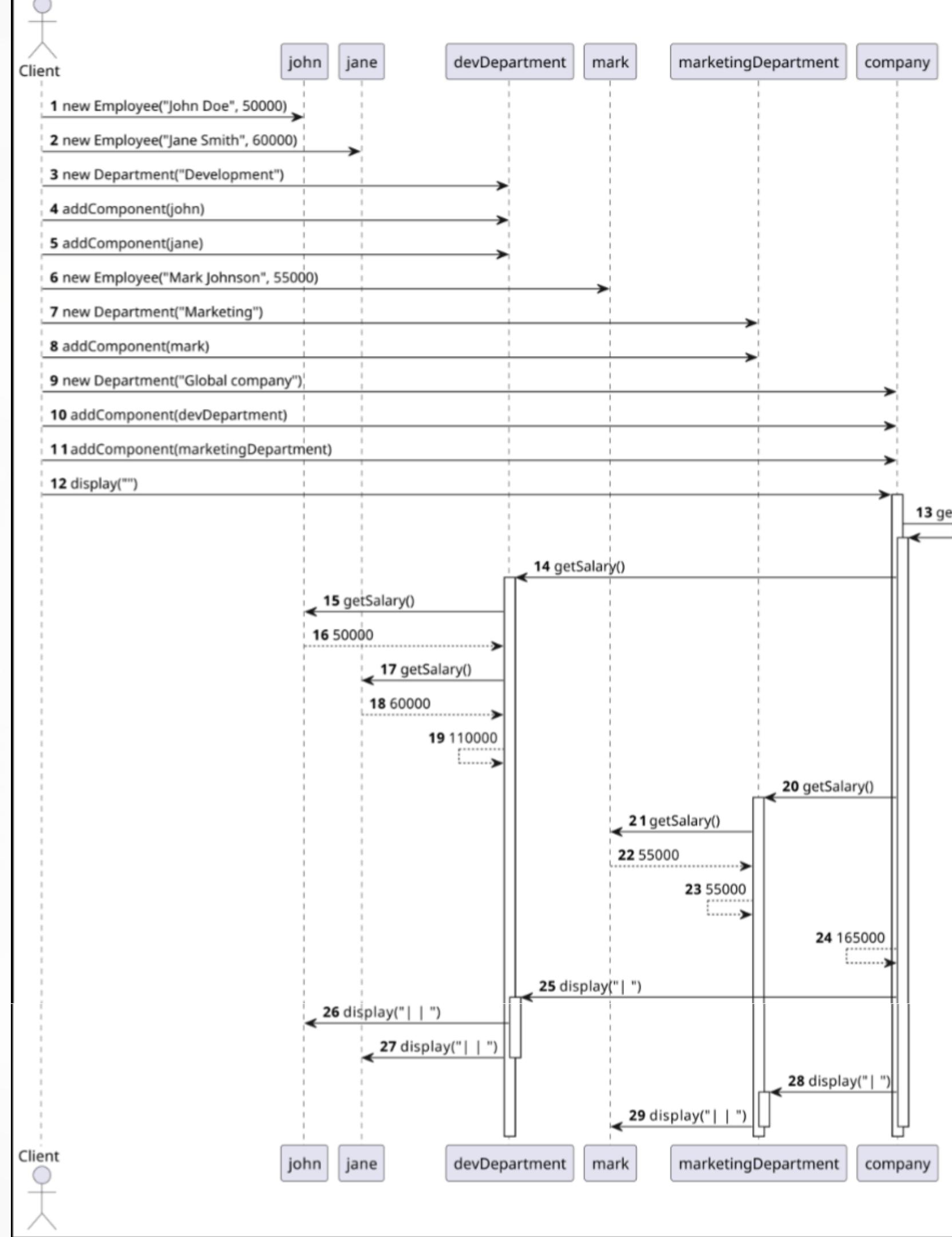
A large red YouTube play button icon is overlaid on the code editor window. At the bottom left, there is a "Watch on YouTube" button.



# DIAGRAMME DE CLASSES



# DIAGRAMME DE SÉQUENCES



# CONCLUSION

BUT2 - INFORMATIQUE

BUT2 Informatique - 2025



# BIBLIOGRAPHIE

**Refactoring.Guru** : Pattern Composite / Decorator

**Wikipédia** : Design Patterns

**DigitalOcean** : Composite Design Pattern in Java

**Ionos.fr** : Composite-pattern

**Sfier.dev** : Les design patterns structurels

