Haute École Bruxelles-Brabant
École Supérieure d'Informatique
Bachelor en Informatique

ATL – Ateliers Logiciels

Design pattern Composite

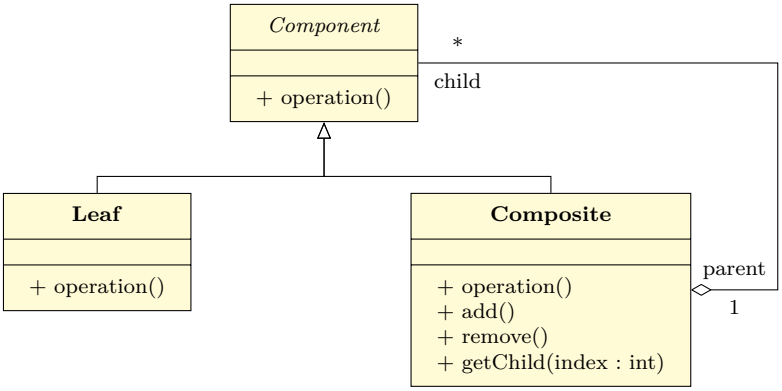
Introduction

Consignes

Ce document introduit le *design pattern* (patron de conception) *Composite*.
Vous pouvez obtenir les ressources de cette fiche via le lien `https://git.esi-bru.be/ATL/designpatterncards`.
N'oubliez pas de vous **connecter** au serveur `git.esi-bru.be` pour avoir accès au téléchargement.

1 Composite

Dans le *Design Pattern Composite*, chaque objet ou composant peut être soit un composant individuel soit un composant composite c'est-à-dire une collection de composants individuels et/ou de composants composites.



Une structure d'arbre permet également de décrire les hiérarchies d'objets de ce *pattern* : un nœud d'un arbre représente soit une feuille (composant individuel) soit un nœud interne (racine d'un sous-arbre, composant composite).

Pour permettre de traiter les composants individuels et les composants composites de manière identique, une interface ou une classe abstraite mère commune sera définie. Un service (méthode) sera directement implémenté dans les composants individuels tandis que les composants composites délégueront en ajoutant éventuellement un comportement.

Version du document
17 octobre 2022

2 Exemple

À titre d'illustration, un exemple basique est disponible sur le serveur *gitlab* de l'école : `https://git.esi-bru.be/ATL/designpatterncards`.

Dans cet exemple, nous modélisons un système de gestionnaire de fichiers. Dans ce système :

- tous les composants héritent de `FileSystemComponent` (abstraite) et possèdent un nom et une taille de fichier ;
- les fichiers sont des `FileSystemLeaf`, un composant individuel du système de fichiers ;
- les répertoires sont des `FileSystemComposite`. En tant que composants composites, ils possèdent une liste de composants enfants, qui sont employés dans le calcul de la taille occupée par le répertoire.

Le système de fichiers est représenté par un répertoire racine, qui peut contenir des fichiers ou des répertoires. Puisque les répertoires sont des composites, chacun peut à son tour contenir des fichiers ou des répertoires.

```
run:
mainFolder, size=10000
subFolder1, size=3000
file22.txt, size=4000
BUILD SUCCESSFUL (total time: 0 seconds)
```

FIGURE 1 – Exemple avec utilisation du *Design Pattern Composite*

3 Dans JavaFX

La librairie JavaFX implémente elle-même le *Design Pattern Composite* dans ses composants de scène. Tous les composants héritent de `javafx.scene.Node`, et certains de ces composants comme les *Layouts* contiennent une liste de nœuds enfants.

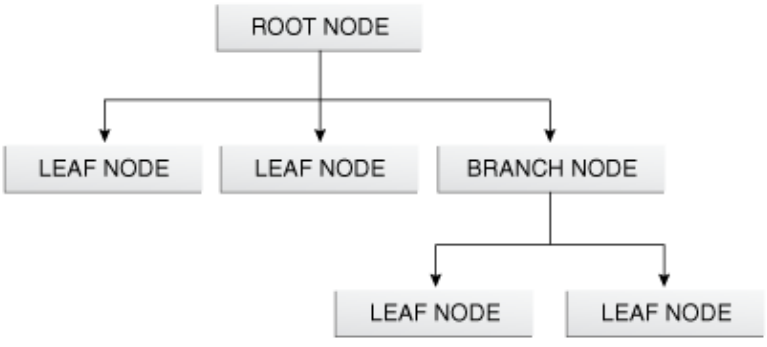


FIGURE 2 – Structure d'arbre du graphe de scène de *JavaFX*

4 Alternative

Dans la structure présentée et dans l'exemple, seules les méthodes partagées par les composants individuels et composants composites sont dans *Component* (*FileSystemComponent*). Une alternative consiste à tout mettre dans l'interface ou la classe abstraite *Component* et à ne rien faire ou à lancer une exception dans l'implémentation pour le composant individuel.