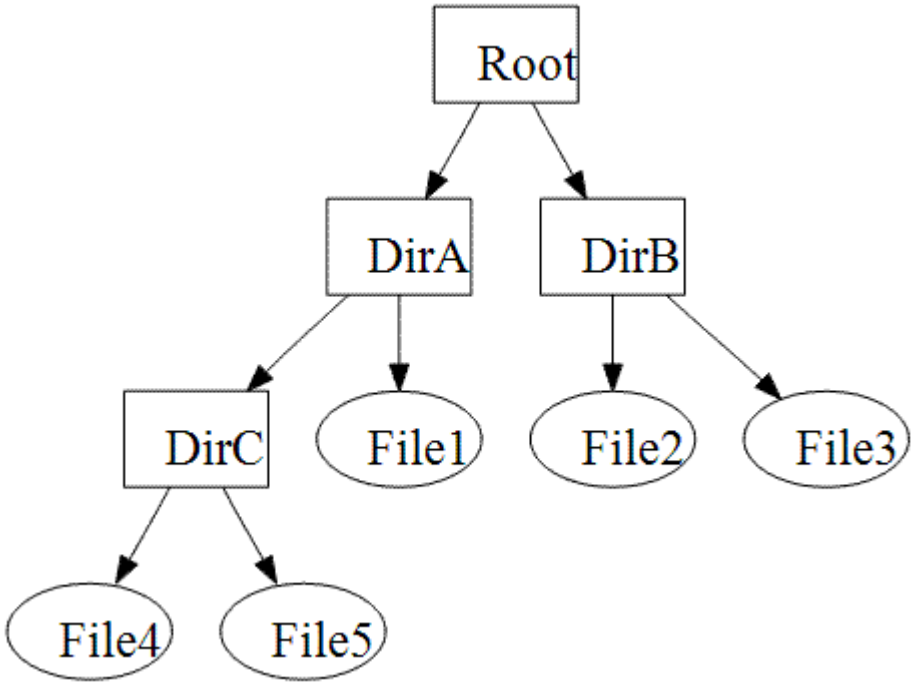
Het Composite Pattern

Objectgeoriënteerd programmeren en ontwerp

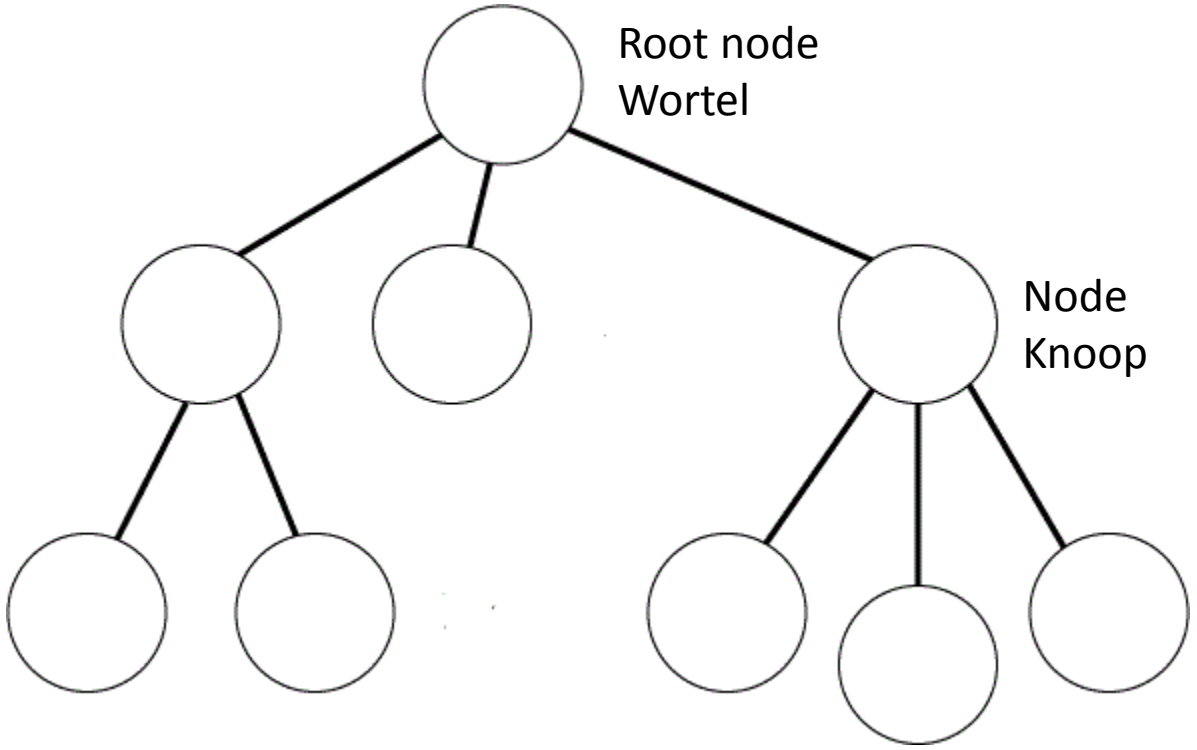
katholieke hogeschool
associatie KU Leuven
vives

Doel van het Composite Pattern

- **Boomstructuren** modelleren,
- zo dat de hele boom en zijn onderdelen op dezelfde manier behandeld kunnen worden.



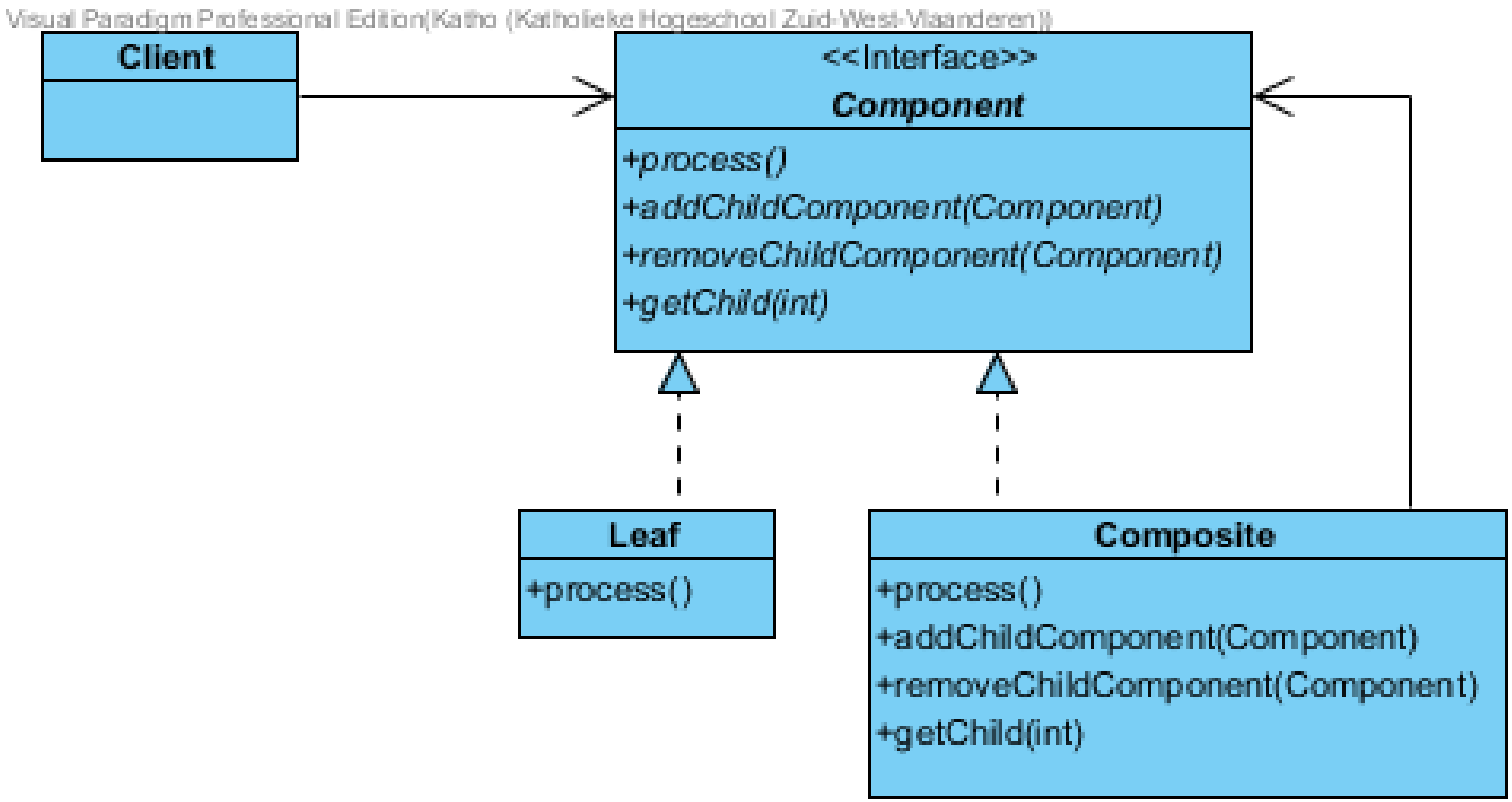
Wat termen



Wat termen

- Een boom bestaat uit **knopen** (*nodes*)
- Elke knoop heeft nul of meer **kinderen** (*child nodes*)
- Elke knoop heeft 1 **ouder** (*parent*), behalve de **wortel** (*root*)
- Een knoop zonder kinderen is een **blad** (*eindknoop*, *leaf*)
- Kinderen van dezelfde ouder zijn **broers** of **zussen** (*siblings*)

Composite Pattern
Structuur



Composite Pattern
Structuur

- *Leaf* modelleert een node die geen kinderen kan hebben.
- *Composite* modelleert een node die wel kinderen kan hebben.
- *Component* is een gemeenschappelijke interface voor Leaf en Composite.

- *Component* heeft alle operaties, zowel die van Leaf als die van Composite als gedeelde operaties.
- *Component* kan een abstracte klasse zijn, met een default-implementatie voor elke operatie.
  (Vb: `throw new UnsupportedOperationException();`)

Composite klasse

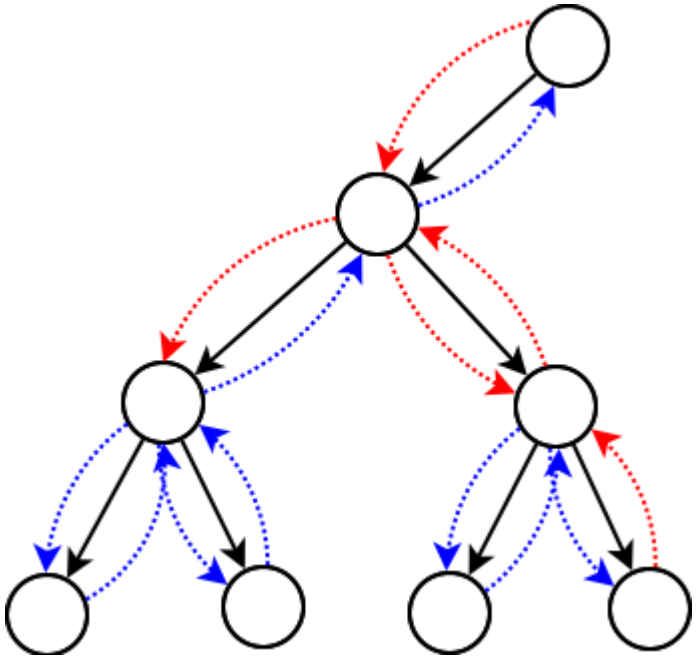
Methodes van Composite kunnen indien nodig itereren over alle kinderen van de Composite.

Die kinderen kunnen dan weer itereren over hun kinderen, enz.

-> een soort van recursie.

Iteratie over een boomstructuur

- Een iterator voor een boomstructuur is redelijk ingewikkeld.



Het Null-pattern

- Een iterator die de kinderen van een bladnode overloopt lijkt zinloos. Er is niks om over te itereren!

- Maar met zo een iterator wordt de code van de Composite iterator eenvoudiger. Er zijn dan geen tests nodig om het onderscheid te maken tussen Leaf en Composite knopen.

- Daarom: een schijniterator: de Null-iterator.
  (zie ook: de Null-operatie in het Command pattern).