



Parcours d'arbres

Présenté par Yann Caron
skyguide

ENSG Géomatique



Plan du cours

Parcours en profondeur

Pre-order

In-Order

Post-Order

Parcours en largeur

Implémentation



Enjeu

- ✓ Un arbre est une structure non linéaire
NB : Liste chaînée en est un cas particulier
- ✓ Dans quel ordre parcourir ses nœuds
- ✓ Phase de CodeGen en compilation

Parcours en profondeur

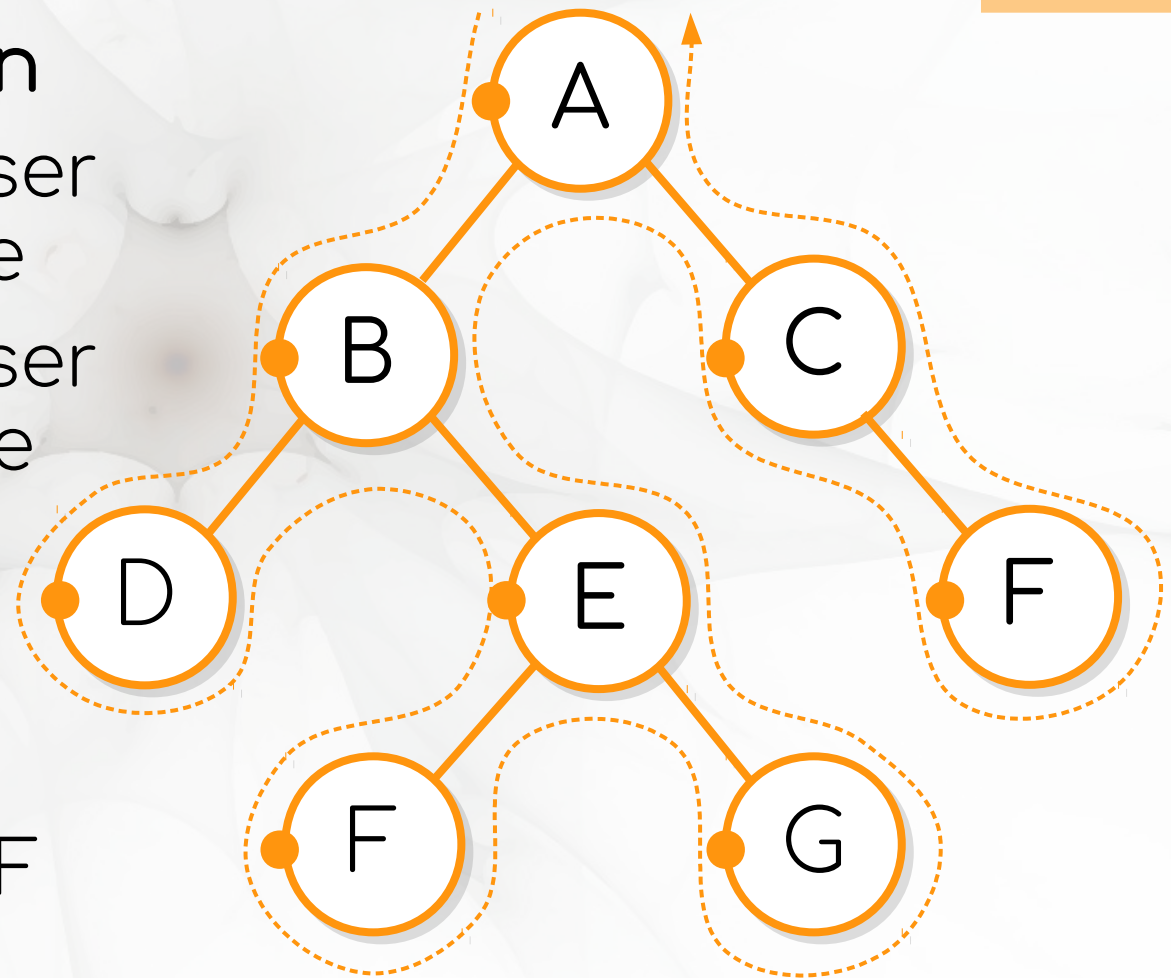


Pre-Order

Récuratif

- ✓ Effectue l'action
- ✓ S'il existe, traverser l'élément gauche
- ✓ S'il existe, traverser l'élément à droite

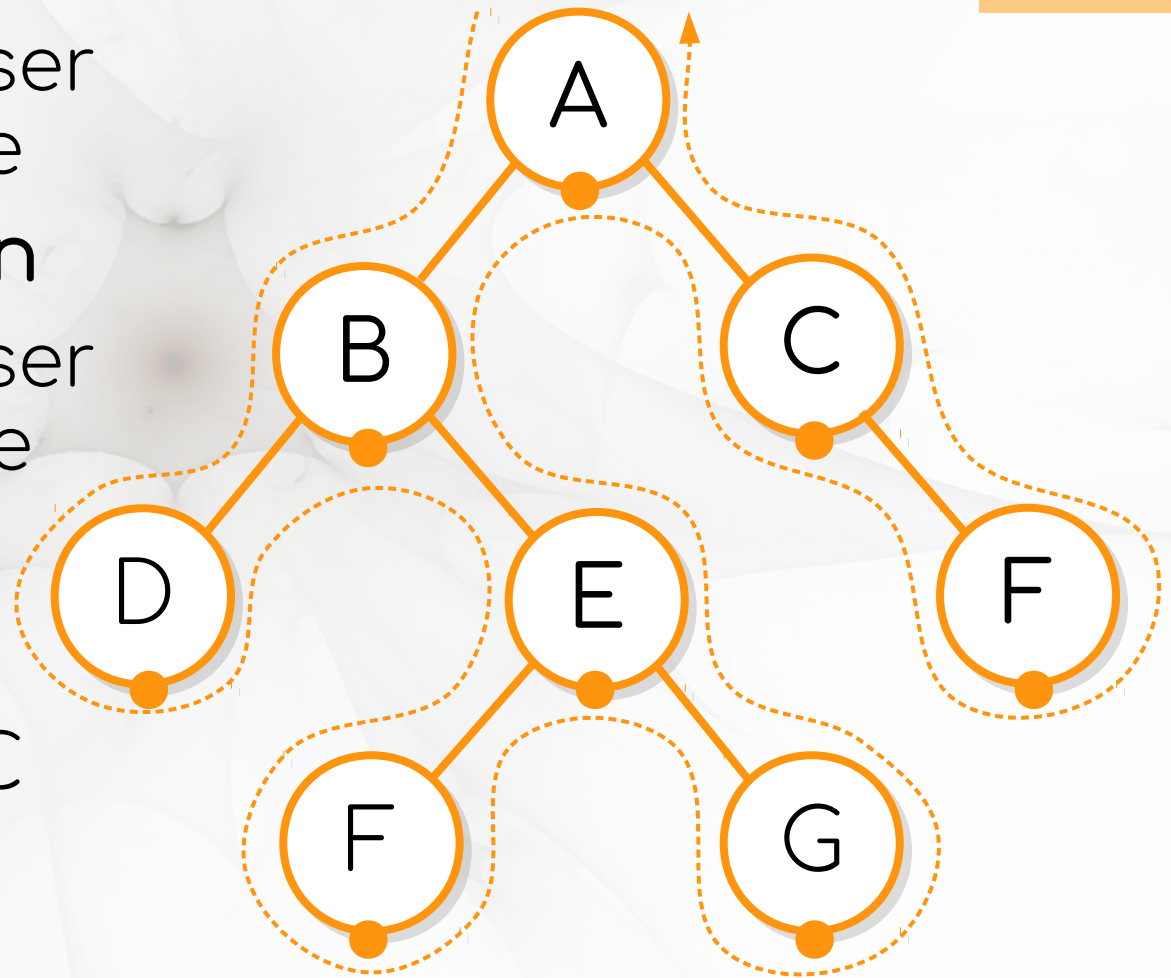
- ✓ Ordre
A, B, D, E, F, G, C, F



In-Order

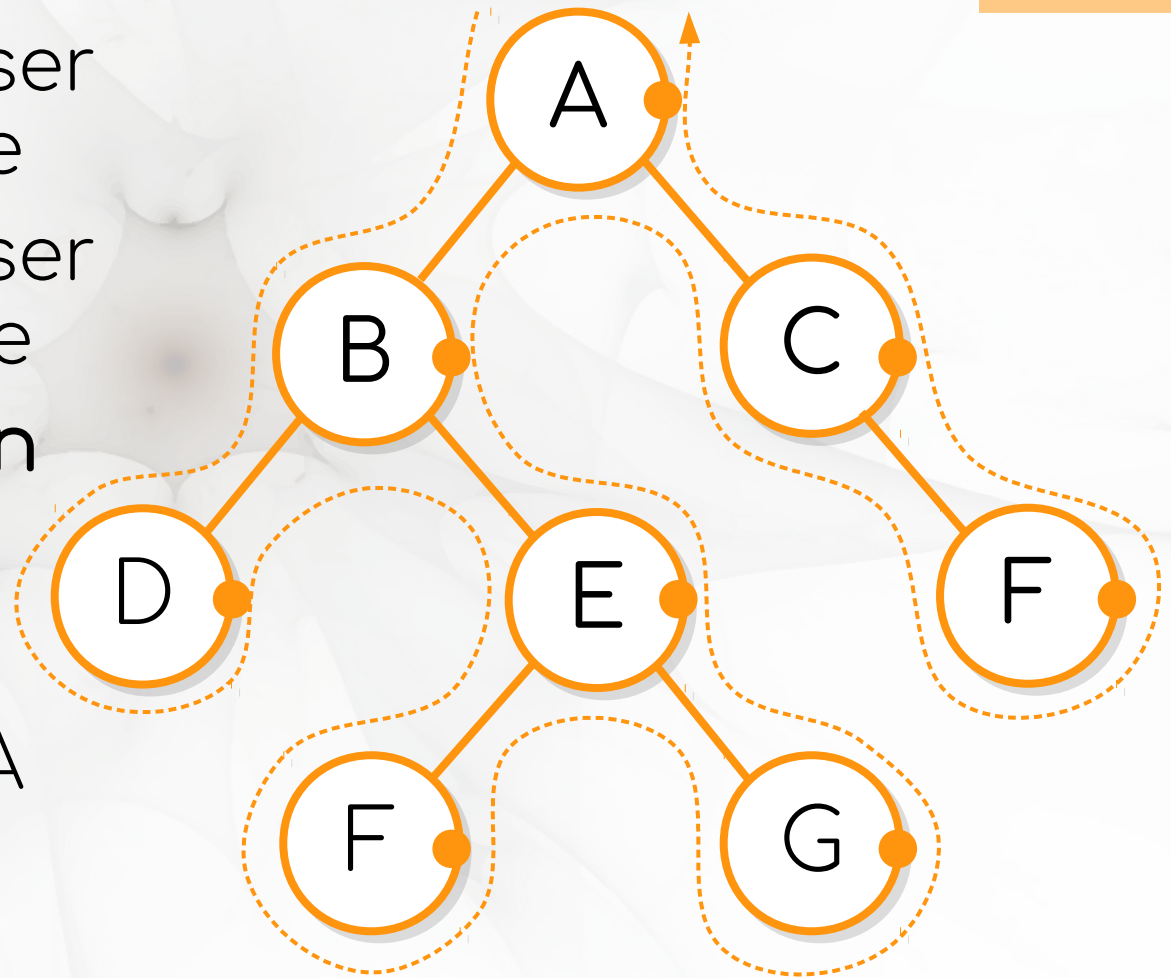
Récurusif

- ✓ S'il existe, traverser l'élément gauche
- ✓ **Effectue l'action**
- ✓ S'il existe, traverser l'élément à droite
- ✓ Ordre
D, B, F, E, G, A, F, C



Post-Order

- ✓ S'il existe, traverser l'élément gauche
- ✓ S'il existe, traverser l'élément à droite
- ✓ **Effectue l'action**
- ✓ Ordre
D, F, G, E, B, F, C, A



Algorithme générique

Récuratif

- ✓ Effectue l'action pré-order
- ✓ Pour tous les éléments enfants :
 - ✓ Traverser l'élément enfant
 - ✓ Effectue l'action in-order
- ✓ Effectue l'action

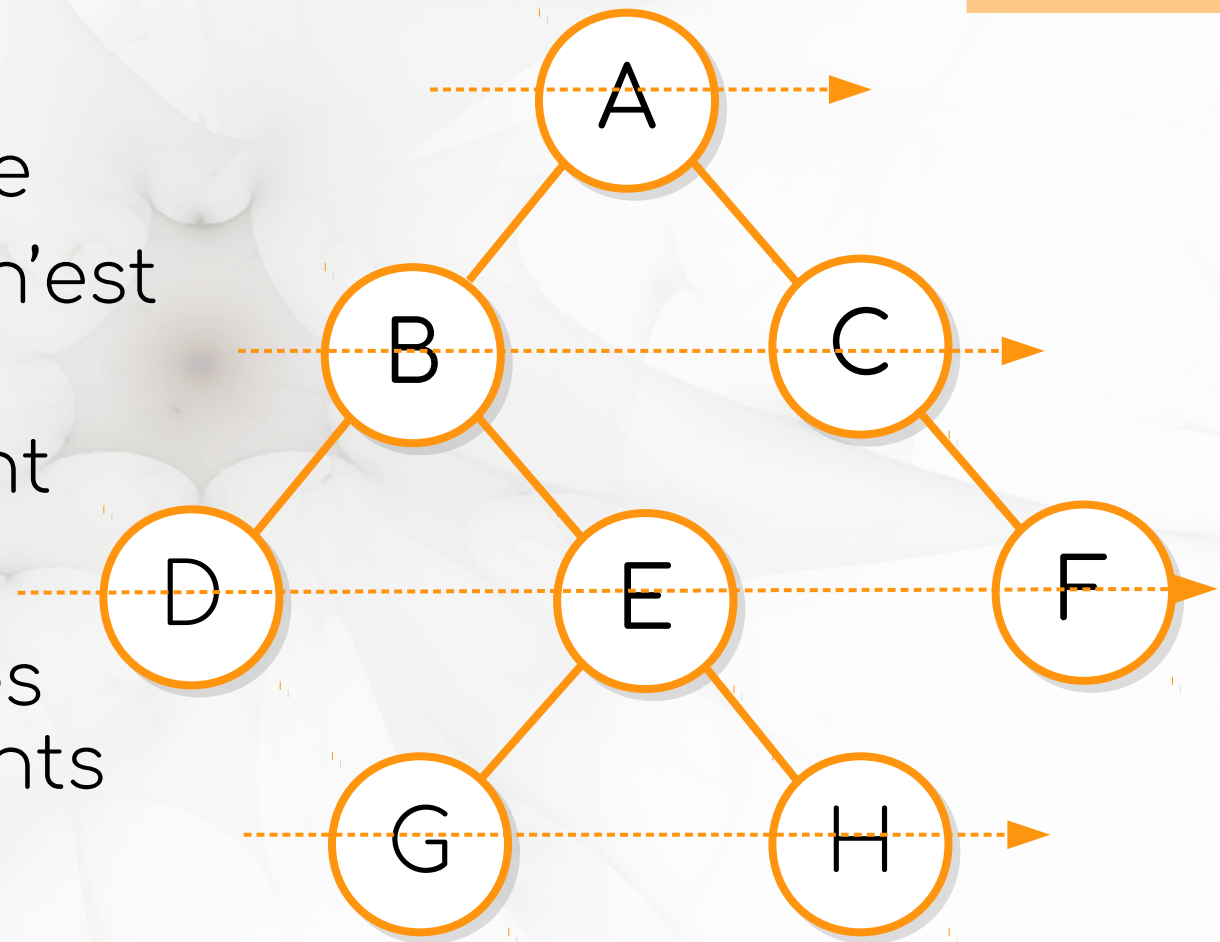
Parcours en largeur



En largeur

Impératif

- ✓ Créer une file
- ✓ Ajouter la racine
- ✓ Tant que la file n'est pas vide
- ✓ Retirer l'élément
- ✓ **Action**
- ✓ Ajouter tous les éléments enfants
- ✓ Ordre
A, B, C, D, E, F, G, H



Algorithme générique

- ✓ Créer une structure (pile, file, file de priorité)
- ✓ Ajouter l'élément racine
- ✓ Tant que la structure n'est pas vide
 - ✓ Retirer l'élément
 - ✓ Effectuer le traitement (ou tester le critère recherche)
 - ✓ Ajouter les enfants de l'élément dans la structure

Observations

- ✓ Si la structure est une pile (stack)
 - ✓ Parcours en profondeur pre-order
 - ✓ Attention : il faut inverser l'ordre des enfants lorsqu'ils sont ajoutés à la pile
- ✓ Si la structure est une file (queue)
 - ✓ Parcours en largeur

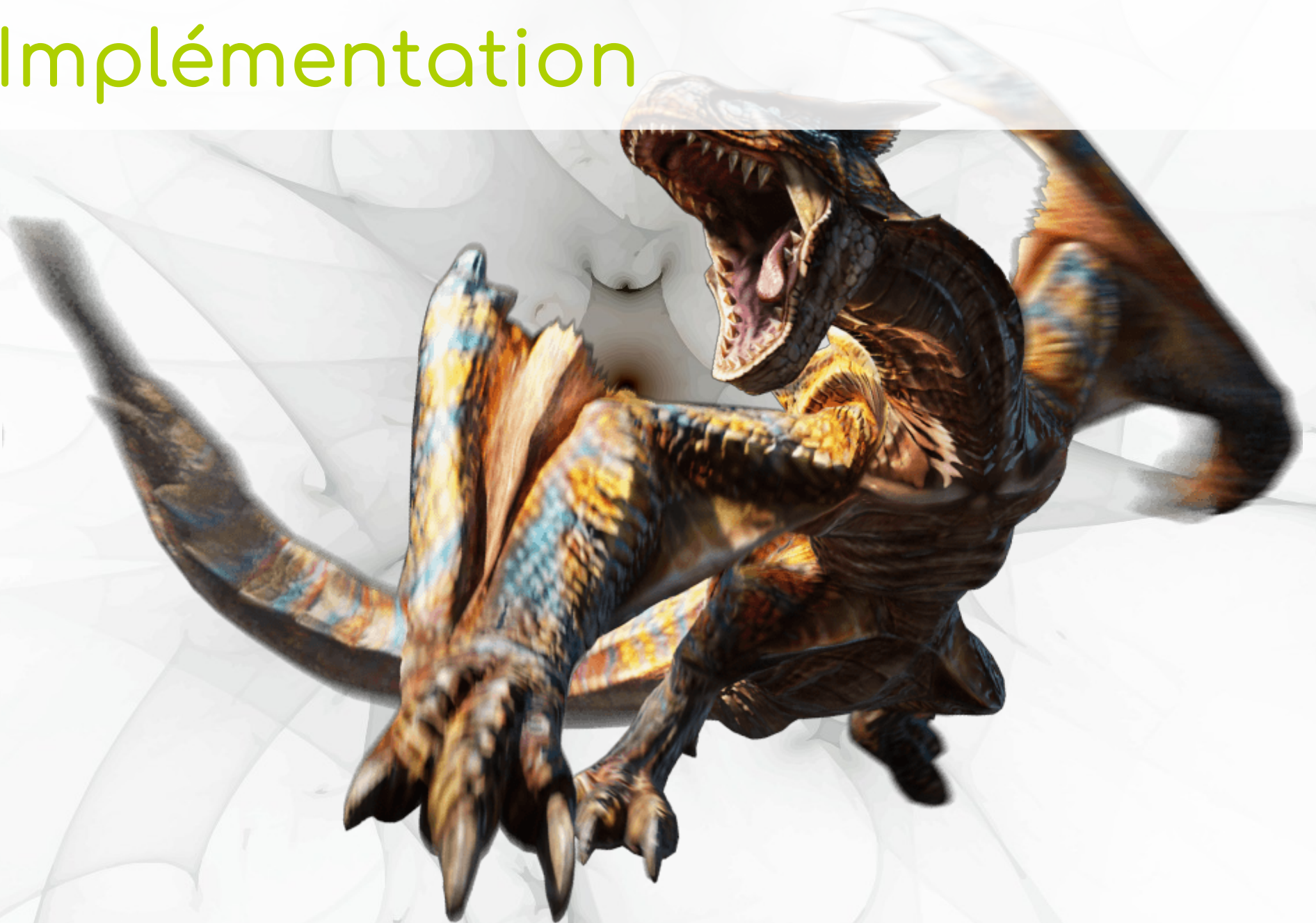
Observations

- ✓ Si la structure est une file de priorité :
 - ✓ Et si la priorité c'est le cumul de distance
 - ✓ = Algorithme de Dijkstra
- ✓ Et si la priorité est la distance de l'hypoténuse
 - ✓ = A* search

Autres algorithmes

- ✓ Best first search
 - ✓ Algorithme de Dijkstra
 - ✓ A* search
 - ✓ Bean search (profondeur limitée)
- ✓ Min-max – alpha-beta
- ✓ Binary search tree
- ✓ Kd-tree
- ✓ Quad-tree

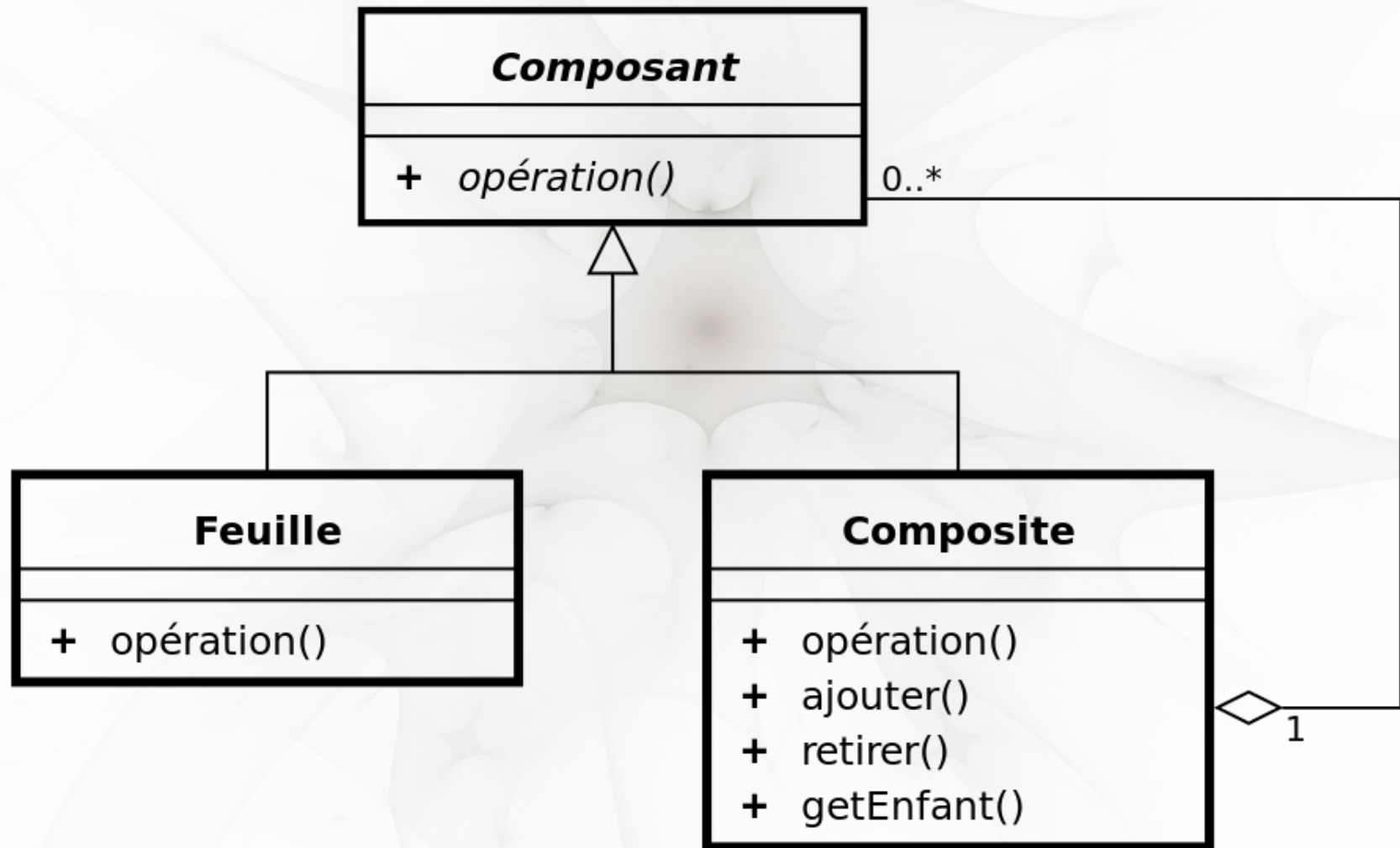
Implémentation



Composite

- ✓ En programmation orientée objet le patron de conception **Composite** permet de déclarer un arbre
- ✓ Une super classe
- ✓ Une feuille qui en hérite
- ✓ Un nœuds qui en hérite et que contient une liste d'éléments de type super classe

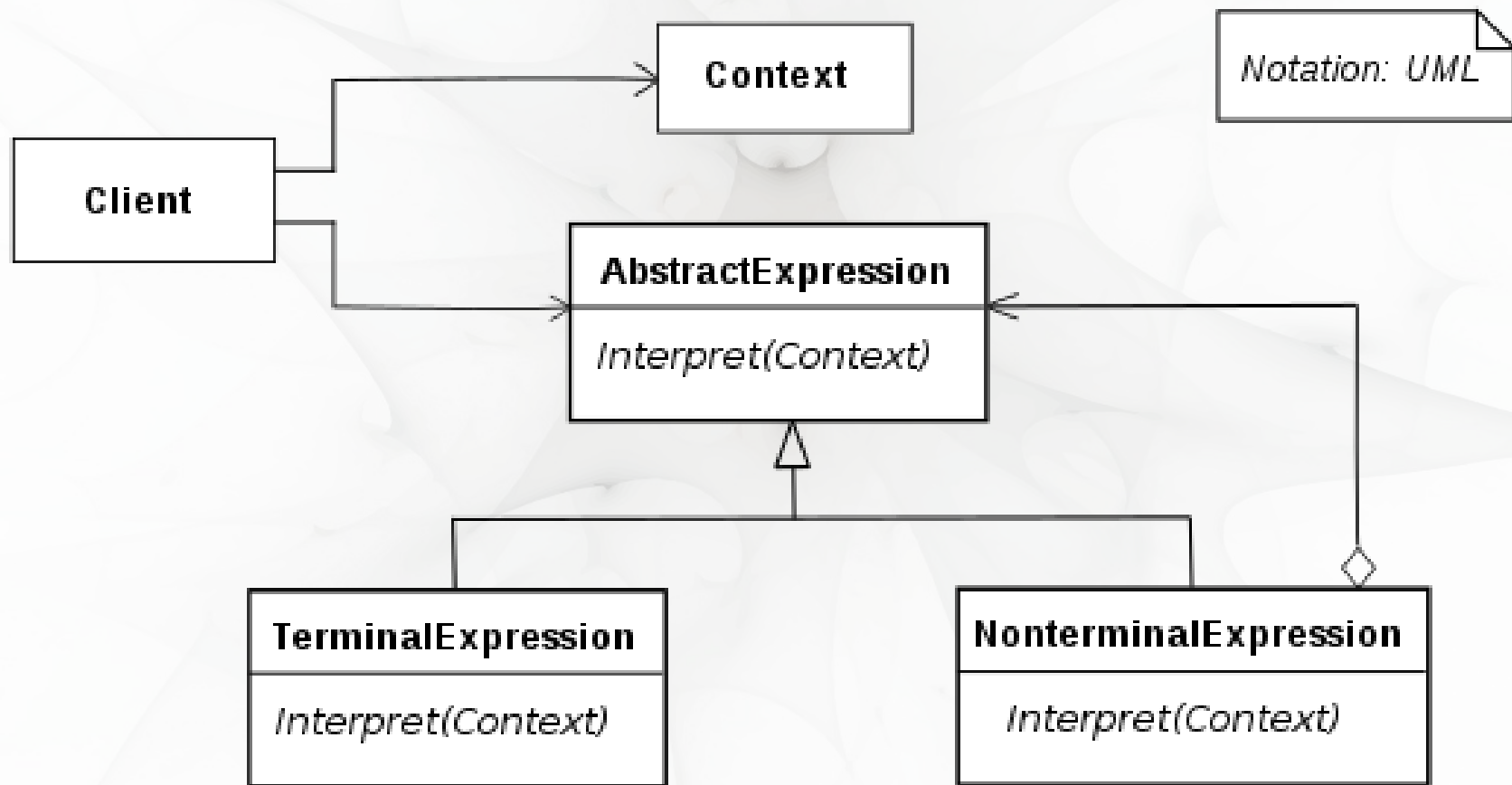
Composite



Interpreter

- ✓ Le patron **interpreter** est une extension du patron composite
- ✓ Il ajoute une méthode **eval** aux composants qui permet une évaluation en profondeur en fonction de la topologie de l'arbre
- ✓ Un contexte (l'ensemble des ressources nécessaires) est passé en paramètre aux méthodes **interpret**

Interpreter

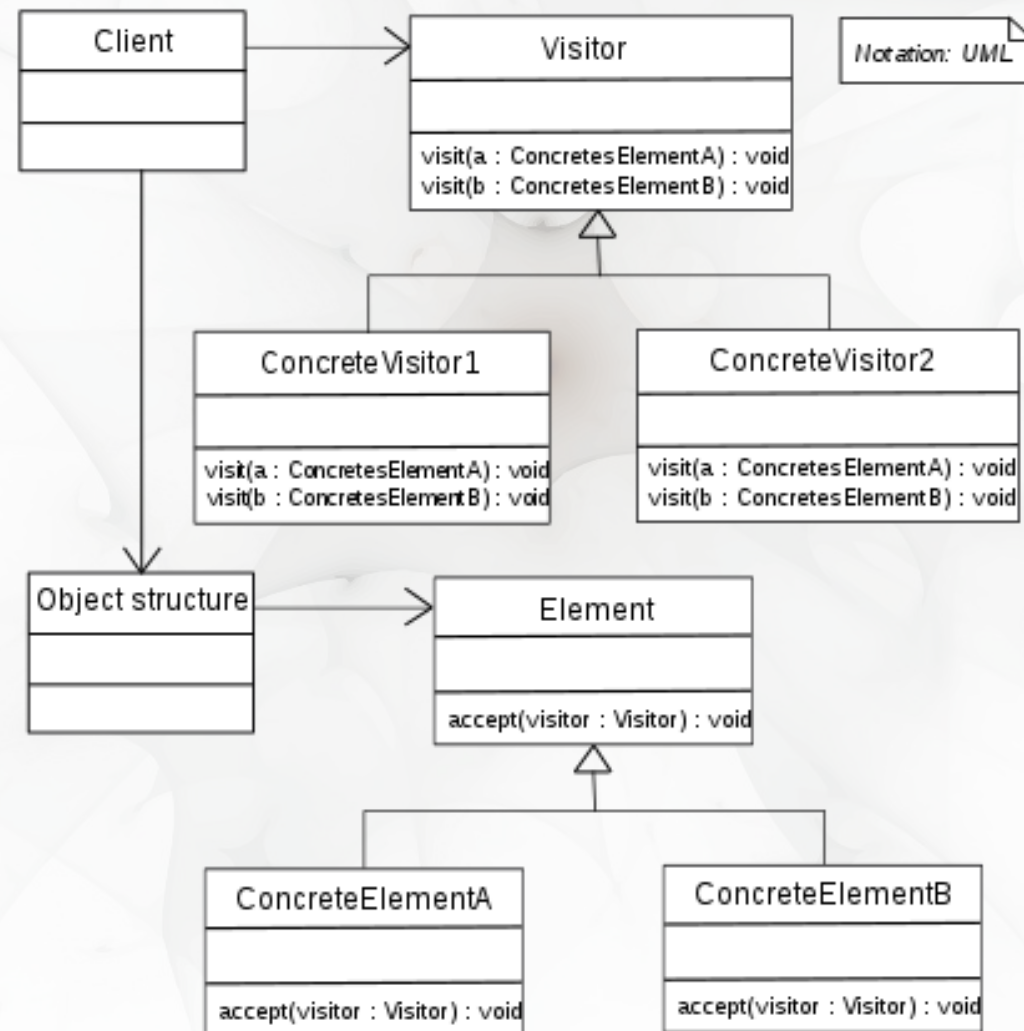


Visitor

- ✓ Il peut être utile de séparer la structure des données de son implémentation
- ✓ Dans ce cas, on utilise un patron visiteur
- ✓ Celui ci intègre tous les comportements des nœuds de l'arbre dans des méthodes distinctes
- ✓ Il s'enregistre auprès de la structure par un parcours en profondeur
- ✓ La méthode **eval** de **l'interpreter** se contente de déléguer l'appel au **visitor**

Visitor

Source : GofVisitor.java



Merci de votre attention

