Le plan de la présentation

- Le pattern de comportement Commande
- Le pattern de comportement Observateur
- Le pattern structurel Composite

Le pattern de comportement : Commande

- **Motivation**
  - On considère une application qui permet de faire des dessins.
  - L'item de menu "nouveau" permet de créer et d'ajouter à l'application un nouveau dessin
  - L'item de menu "coller" permet d'insérer dans un dessin le contenu du presse-papier

Le pattern de comportement : Commande

- On remarque que les deux objets "Item" ont à transmettre une requête à des objets totalement différents :
  - Un objet Application pour le premier
  - Un objet Dessin pour le second.

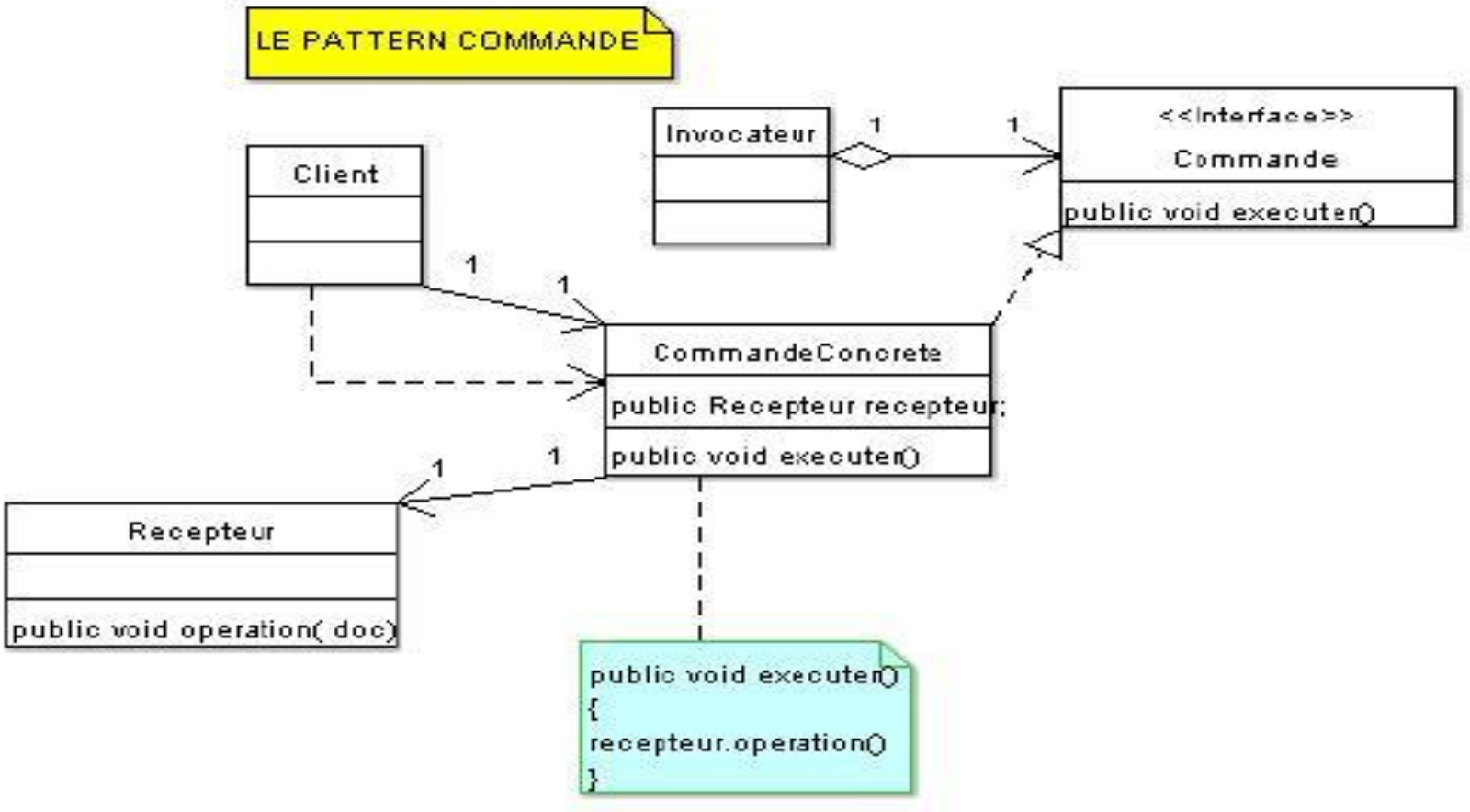
- Pourtant ces deux objets ont en commun d'être des items de menu donc issus de la même classe.

Le pattern de comportement : Commande

- Intention
  - d'encapsuler une requête comme un objet pour découpler complétement l'objet qui émet la requête de celui qui va la traiter. L'objet emetteur n'aucune information sur la nature du traitement de la requête.
  - Le traitement de la requête n'est plus exécutée directement par l'objet concerné.

Le pattern de comportement : Commande

- **Motivation**



Le pattern de comportement : Commande

- **Constituants**
  - **Une interface Commande pour exécuter une opération**
  - **Une ou plusieurs classes commandes concrètes (CmdColler, CmdNouveau) qui implémentent Commande.**
  - **Une ou plusieurs classes Invocateur (les items de menus)**

Le pattern de comportement : Commande

- **Constituants**
  - **Des classes Récepteur (Application, Document) dont les objets sont impactés par l'exécution de la commande. Ses objets exécutent la commande.**

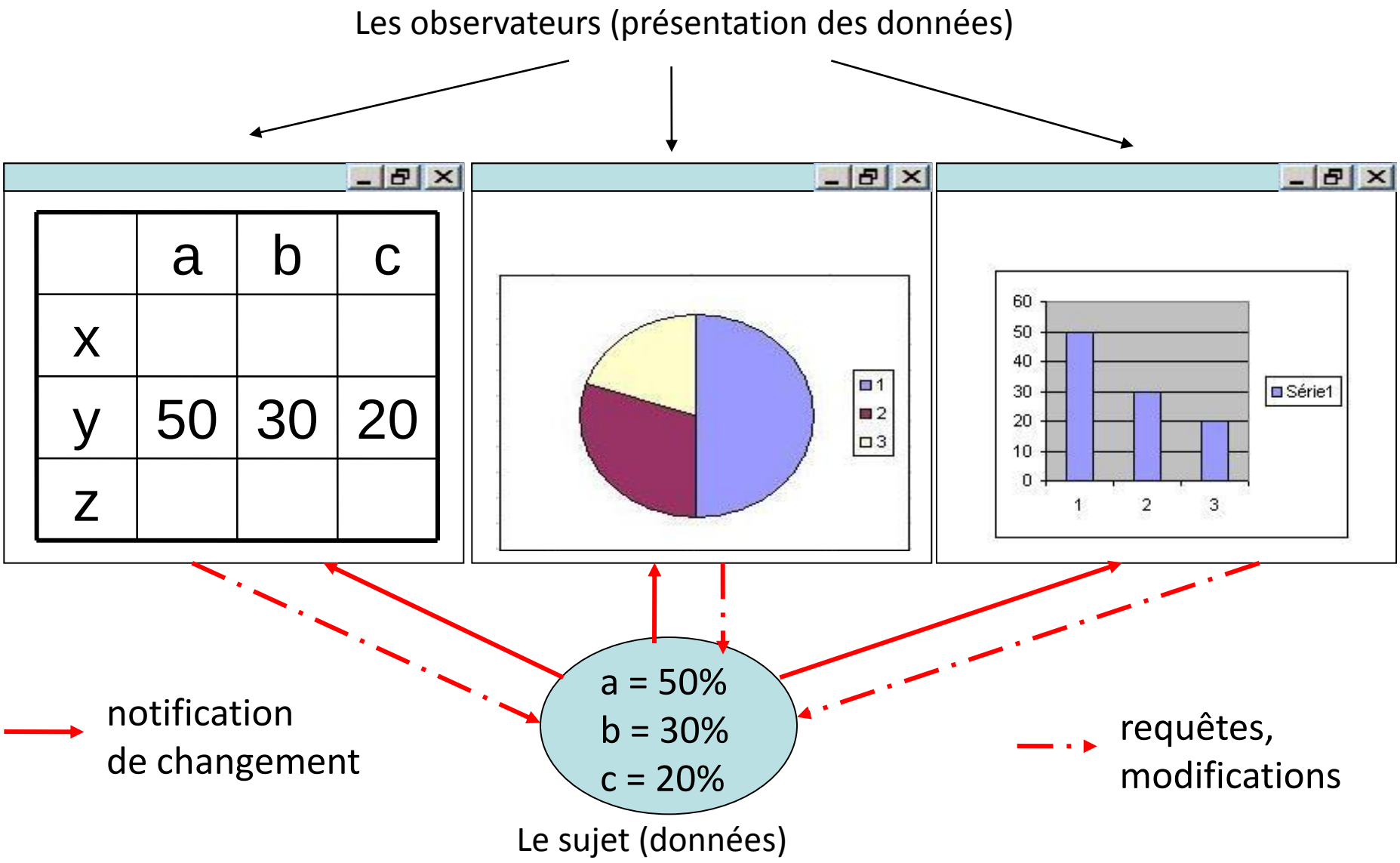
Le pattern de comportement : Commande

- **Constituants**
  - **Une classe Client (Application) qui crée la commande concrète, le récepteur et le positionnement du récepteur dans la commande concrète.**

Le pattern de comportement : Observateur

- **Motivation**
  - Dans de nombreuses IHM, les données peuvent être présentées à l'utilisateur de plusieurs façons.
  - Comment rendre les données et leurs représentations indépendantes.

Le pattern de comportement : Observateur

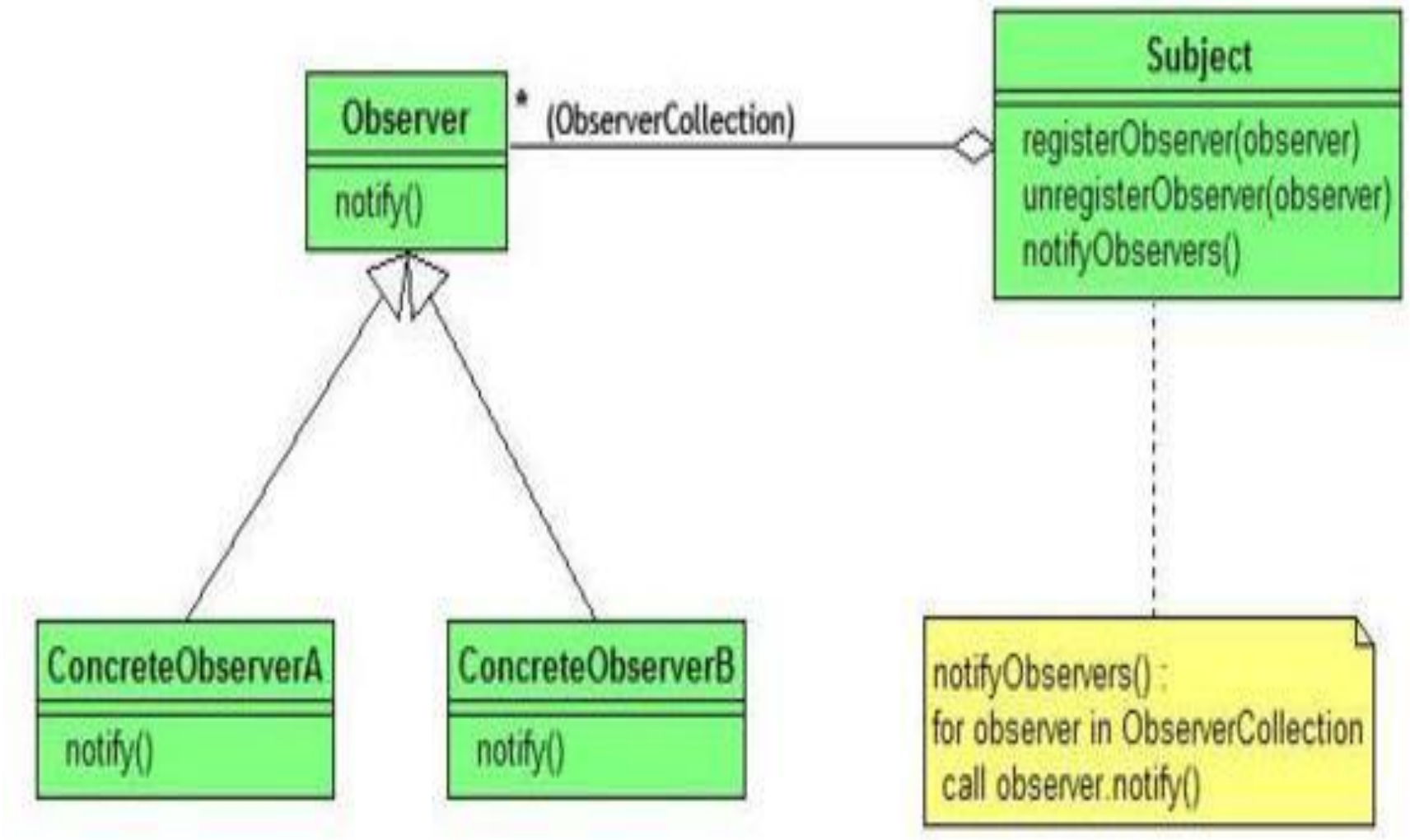


Le pattern de comportement : Observateur

- **Intention**

  Créer des dépendances entre un objet X et ses objets liés de telle façon que dès que l'objet X change d'état tous les objets liés en soient avertis (notifiés) automatiquement.

La structure du pattern Observateur



Ce schéma a été obtenu à l'adresse http://en.wikipedia.org/wiki/Observer_pattern

Le pattern structurel : Composite

- **La motivation**

  Soit un éditeur de dessin pour créer des objets graphiques.

  Ces objets graphiques sont :

  – soit des objets primitifs tels que lignes, Polygones et textes
  – soit des objets définissant un agrégat d'objets graphiques.

Le pattern structurel : Composite

La motivation

On désire traiter de la même façon :

- les objets primitifs et
- les objets agrégats d'objets graphiques

Ils ont en commun d'être des graphiques

Le pattern structurel : Composite

- **La motivation**
  - Image, la classe définissant un agrégat d'objets graphiques.
  - Graphique, une classe abstraite pour manipuler des objets graphiques qui peuvent être :
    - des objets de la classe Image
    - des objets primitifs dont les classes sont Ligne, Polygone et Texte

Le pattern structurel : Composite

- On en déduit qu'un objet Image est une arborescence dont :
  - les noeuds internes sont des agrégats d'objets graphiques
  - les noeuds externes (ou feuilles) sont des formes primitives (lignes, polygones, texte)

Le pattern structurel : Composite

- D'un point de vue général, avec un objet nœud, on doit pouvoir :
  - le dessiner
  - ajouter un noeud enfant (un objet graphique)
  - supprimer un noeud enfant (un objet graphique)
  - récupérer l'un de ses nœuds enfants.

Le pattern structurel : Composite

- Tous les nœuds héritent d'une classe abstraite Graphique qui :
  - déclare l'opération **dessine()**
  - implémente les trois autres opérations avec le code quasiment vide pour les objets primitifs

- Chaque objet primitif implémente son opération dessine().

- Chaque objet de la classe Image implémente l'opération dessine() avec le code suivant :

Pour tout objet graphique g de l'image
f.dessine()

Le pattern structurel : Composite

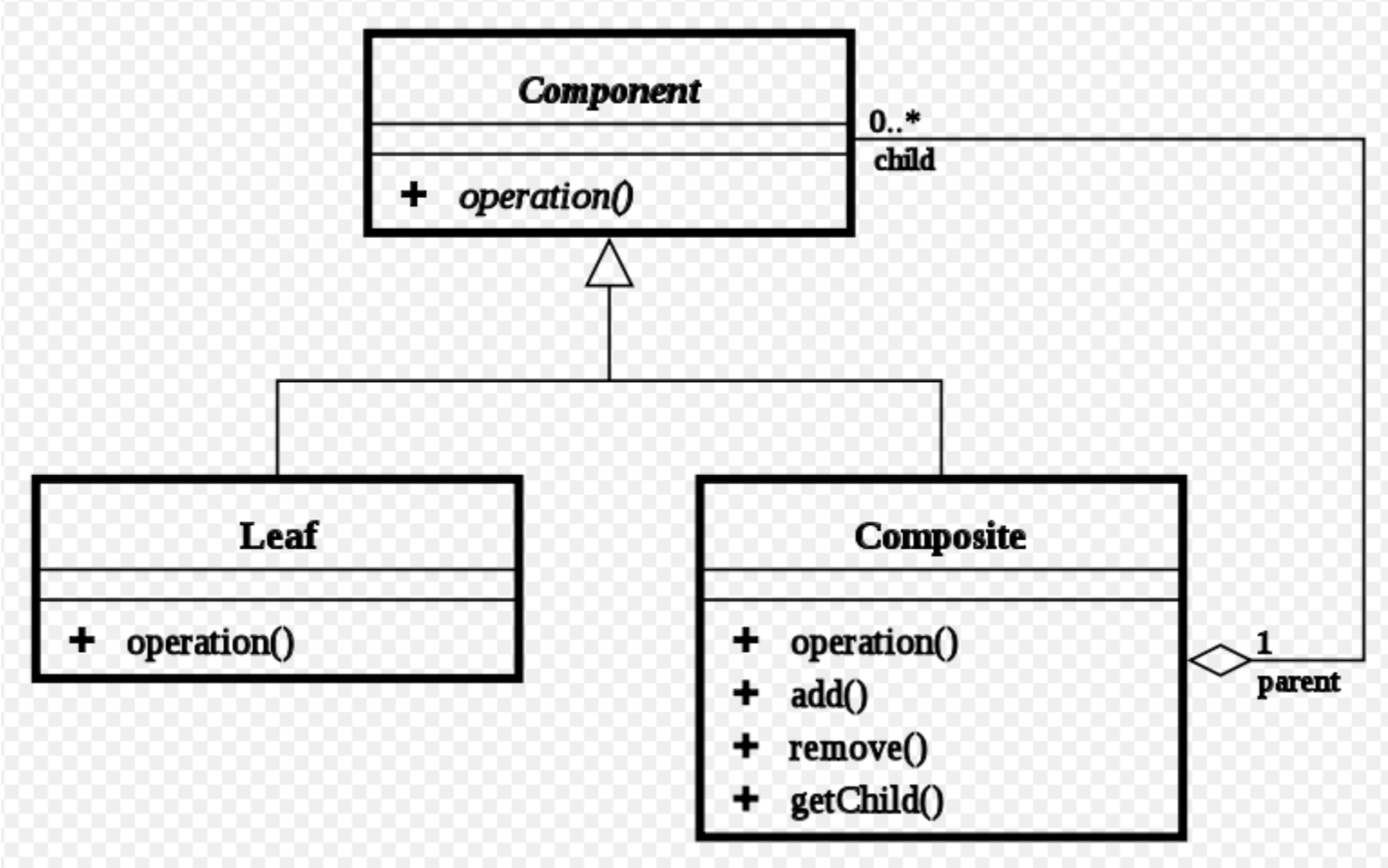
- **L'intention de ce pattern** est d'organiser des objets en structure arborescente.

- Un objet Composite permet de traiter les objets individuels (feuilles de l'arbre) de la même façon que les ensembles organisés (noeuds de l'arbre).

Le pattern structurel : Composite

- **Les constituants**
  - Une classe Composite (Image) qui dérive de la classe Composant
  - Elle définit les opérations liées aux enfants (supprimer,ajouter, récupérer)
  - stocke des composants enfants
  - définit les opérations liées aux composants d'enfants (dessine)

Le pattern structurel : Composite



Ce schéma provient du site wikipedia (en anglais)

Le pattern structurel : Composite

- Les constituants
  - Une classe **Client** (éditeur de dessin) dont les objets manipulent des objets de composition en utilisant des références de la classe abstraite Composant.