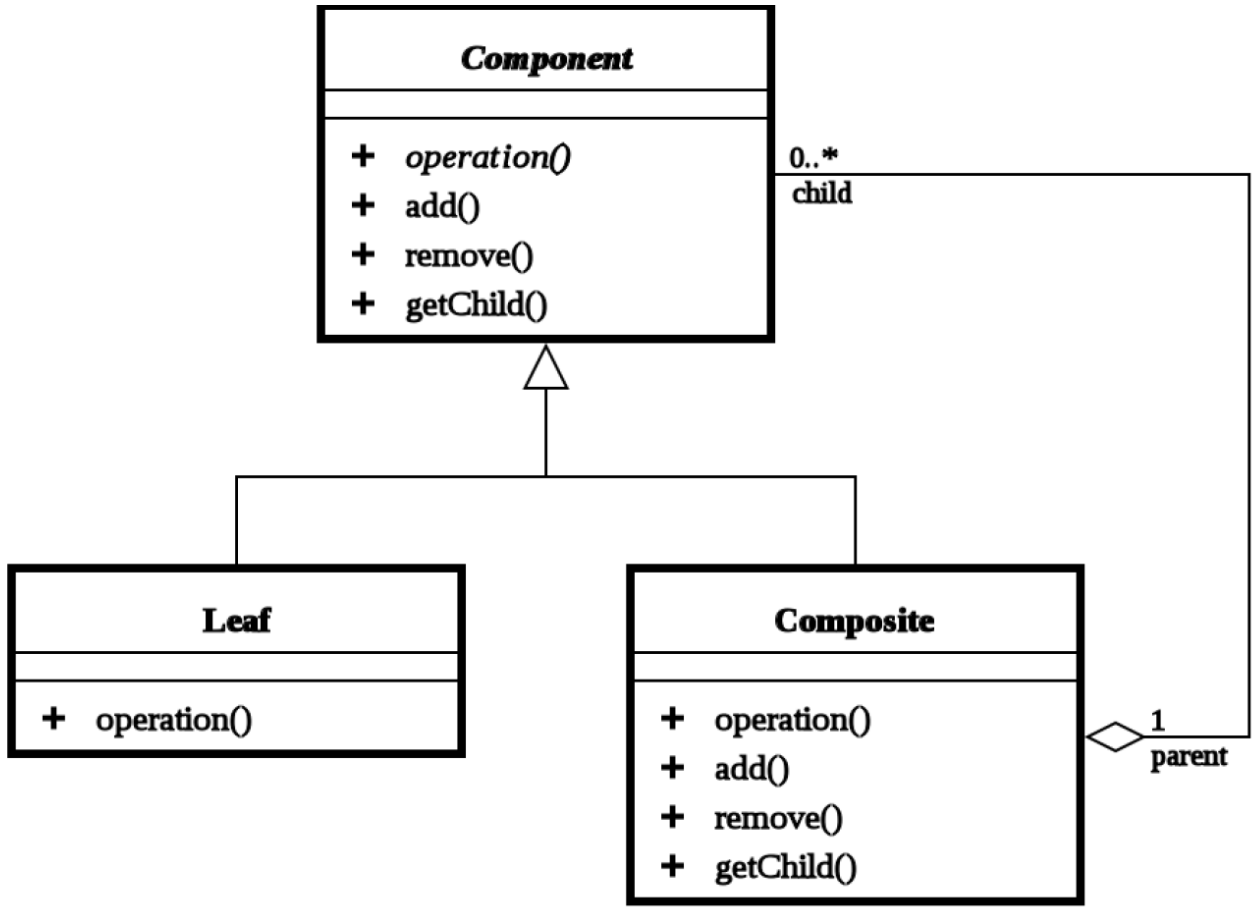
Composite

Composite

- structural pattern
- GoF: *"Compose objects into tree structures to represent part-whole hierarchies"*
- client can execute commands on the root
  - command delegation deep into the tree structure
- Example: any libraries of graphical components (panels/containers + controls).

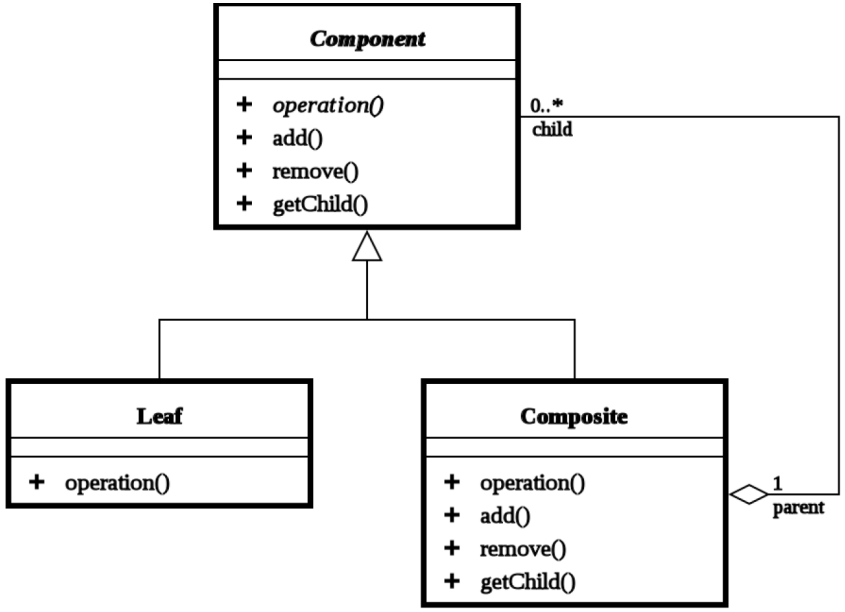
Composite - structure

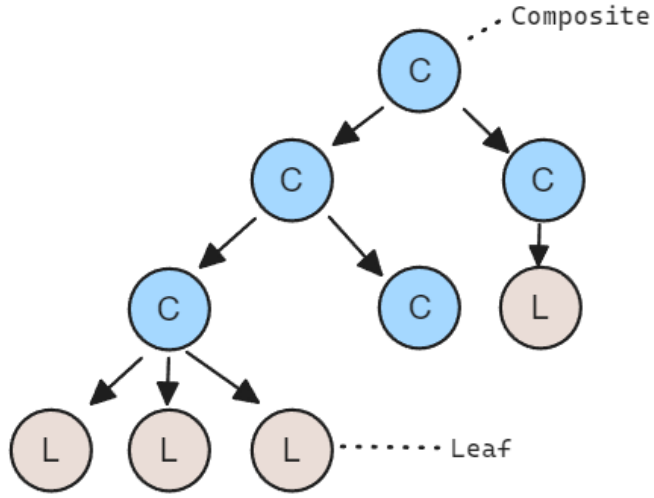


Composite lets clients treat individual objects and compositions of objects uniformly.

Composite - structure

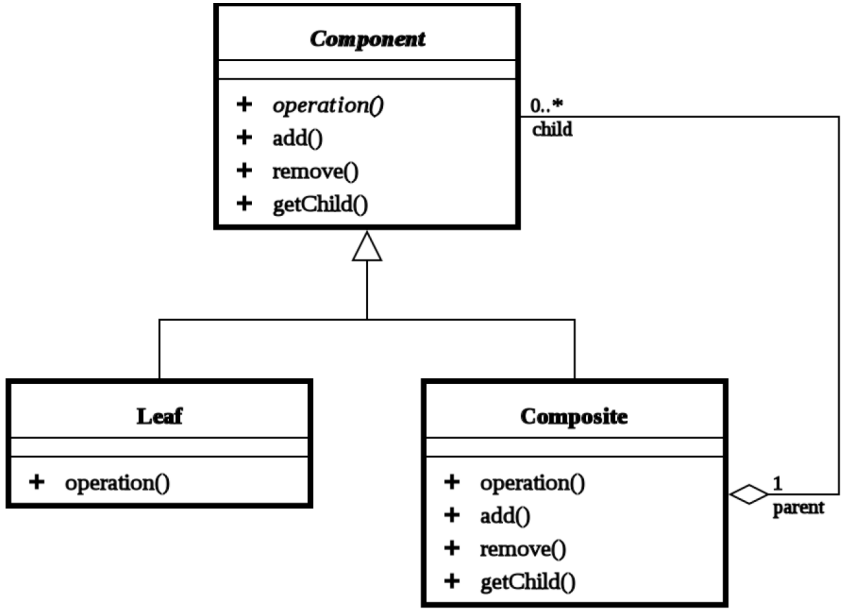
In the oryginal version there is only one non-abstract *Composite*.

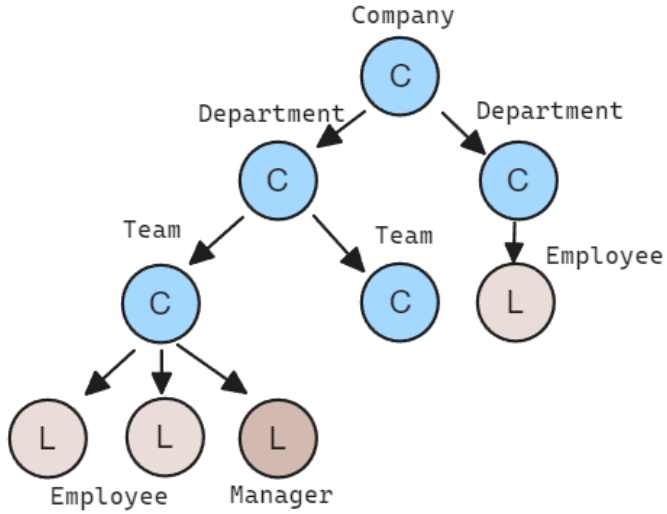




Composite - structure

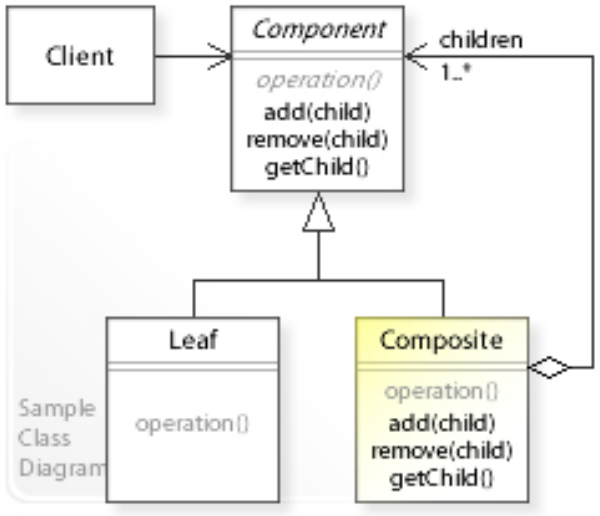
In the oryginal version there is only one non-abstract *Composite*.



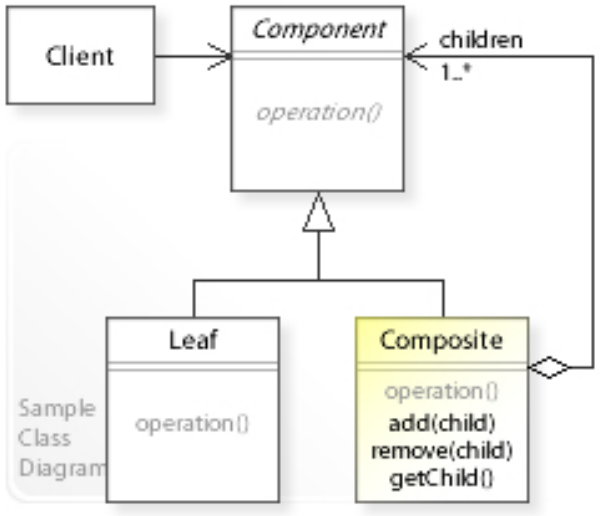


Composite - where to put `add`/`remove` methods?

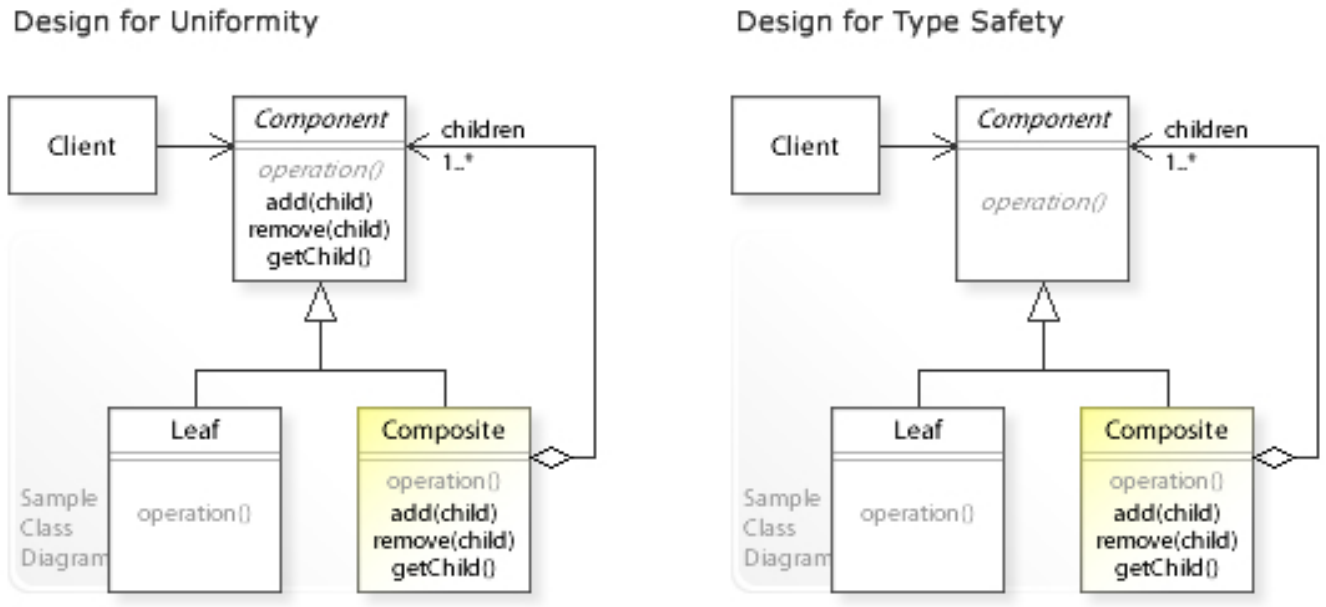
Design for Uniformity



Design for Type Safety

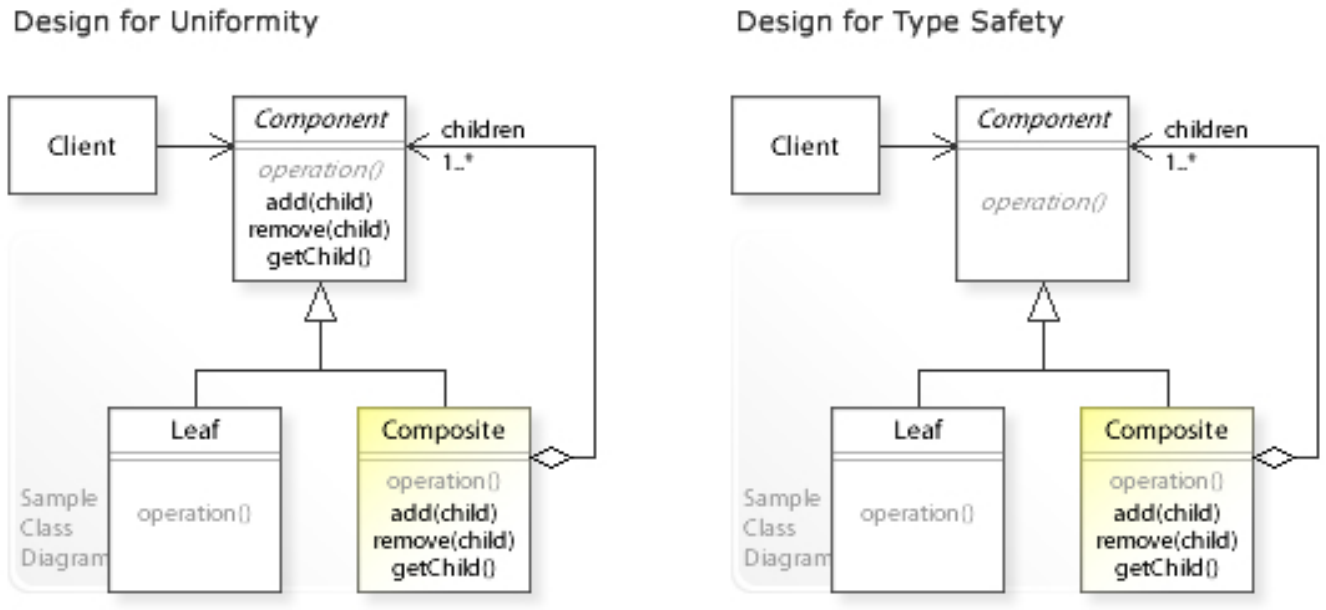


Composite - where to put `add`/`remove` methods?



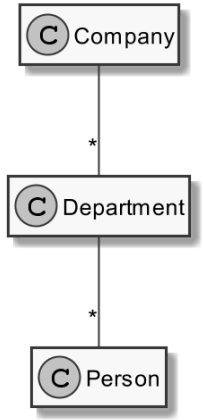
If in **`Component`** - that's great for the pattern intent ("treat as individual objects") but violates "L" (what **`add`**&**`remove`** will do in **`Leaf`**? Throw an exception? Probably the best is at least to return **`Option`**-like result. Or upgrade **`Leaf`** to some composite 🤔)

Composite - where to put `add`/`remove` methods?

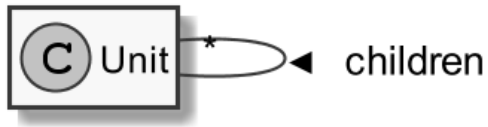


If in **`Composite`** - clients must treat **`Leaf`** and **`Composite`** objects differently, requires *type checking* or similar

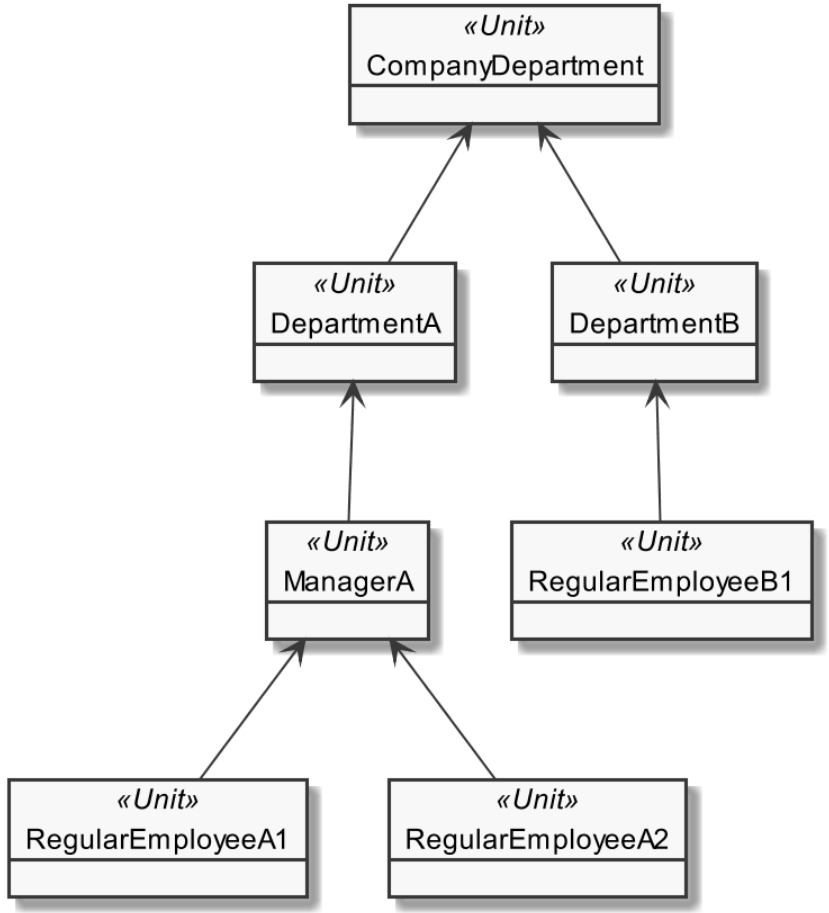
Composite - representing a hierarchy



Composite - representing a hierarchy



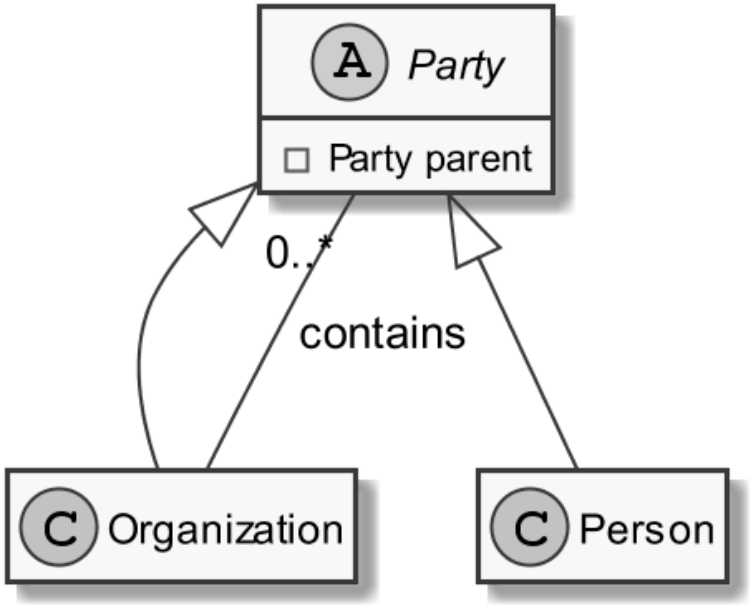
Composite - representing a hierarchy



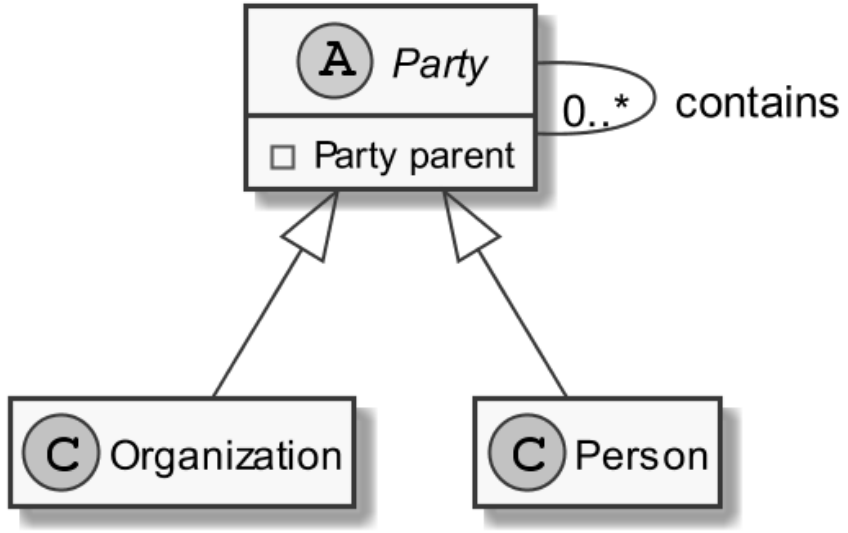
Composite - representing a hierarchy

Composite on various levels:

a) konkretnego podtypu



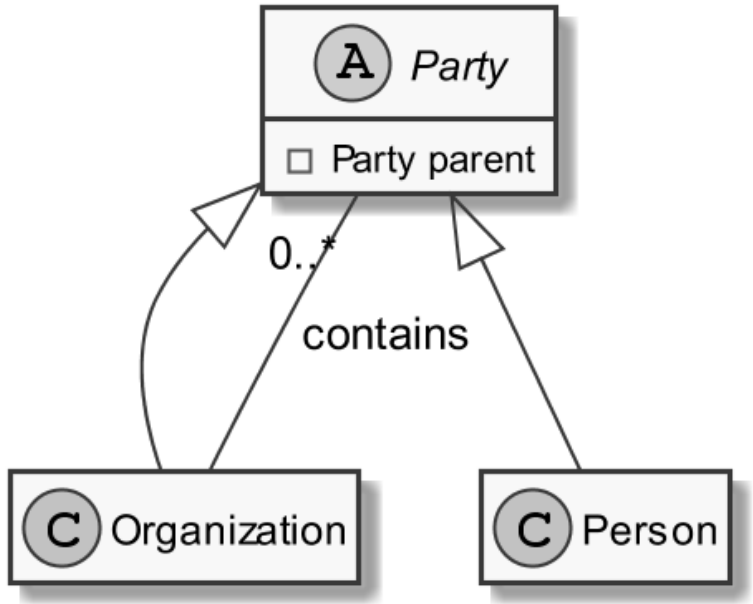
b) ogólnym



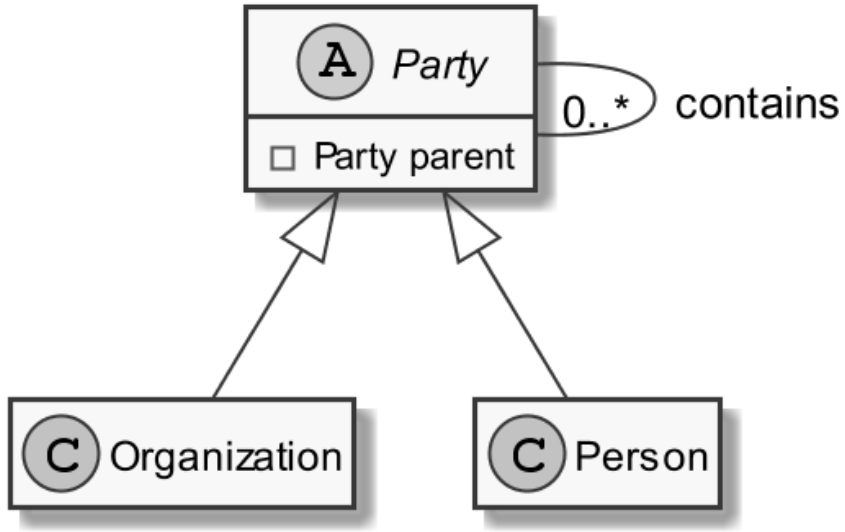
Composite - representing a hierarchy

Composite on various levels:

a) konkretnego podtypu



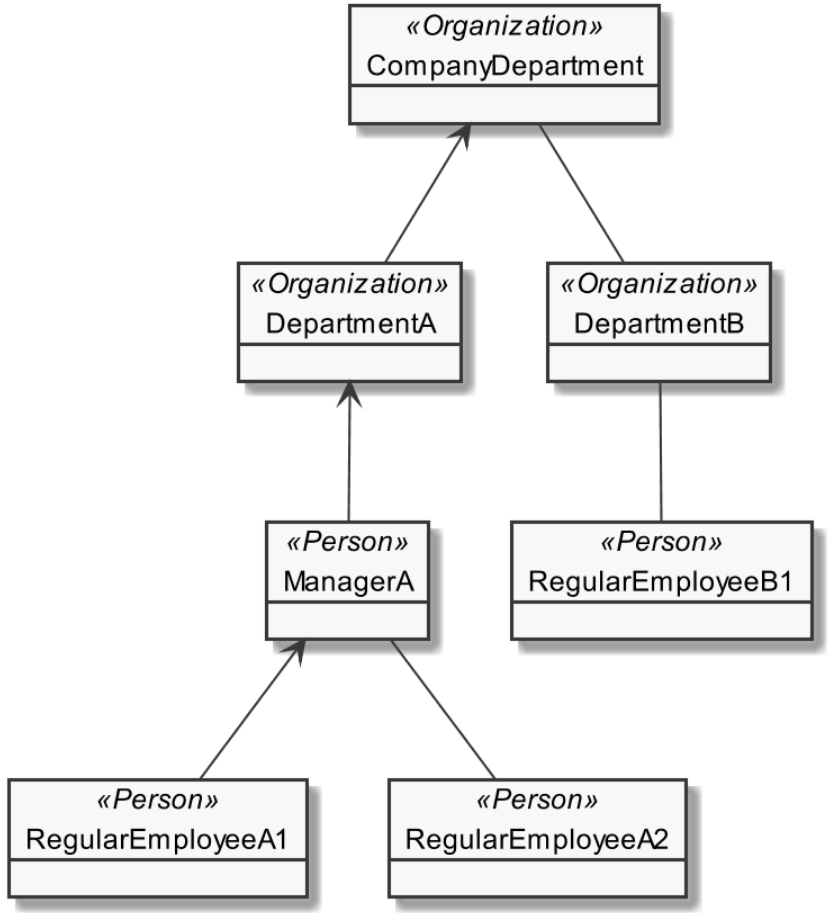
b) ogólnym



Wzorzec **Party** (aka Player, Person, Contractor, Legal Entity, Unit)

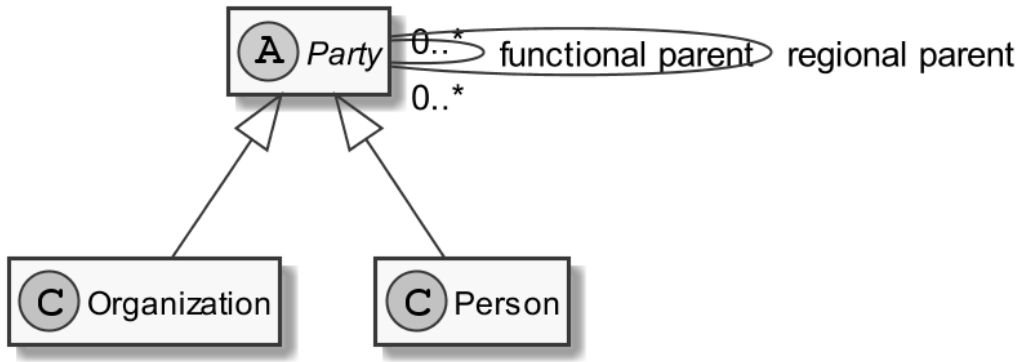
- "supertype of person and organization" - Martin Fowler in Analysis Patterns

Composite - reprezentowanie hierarchii



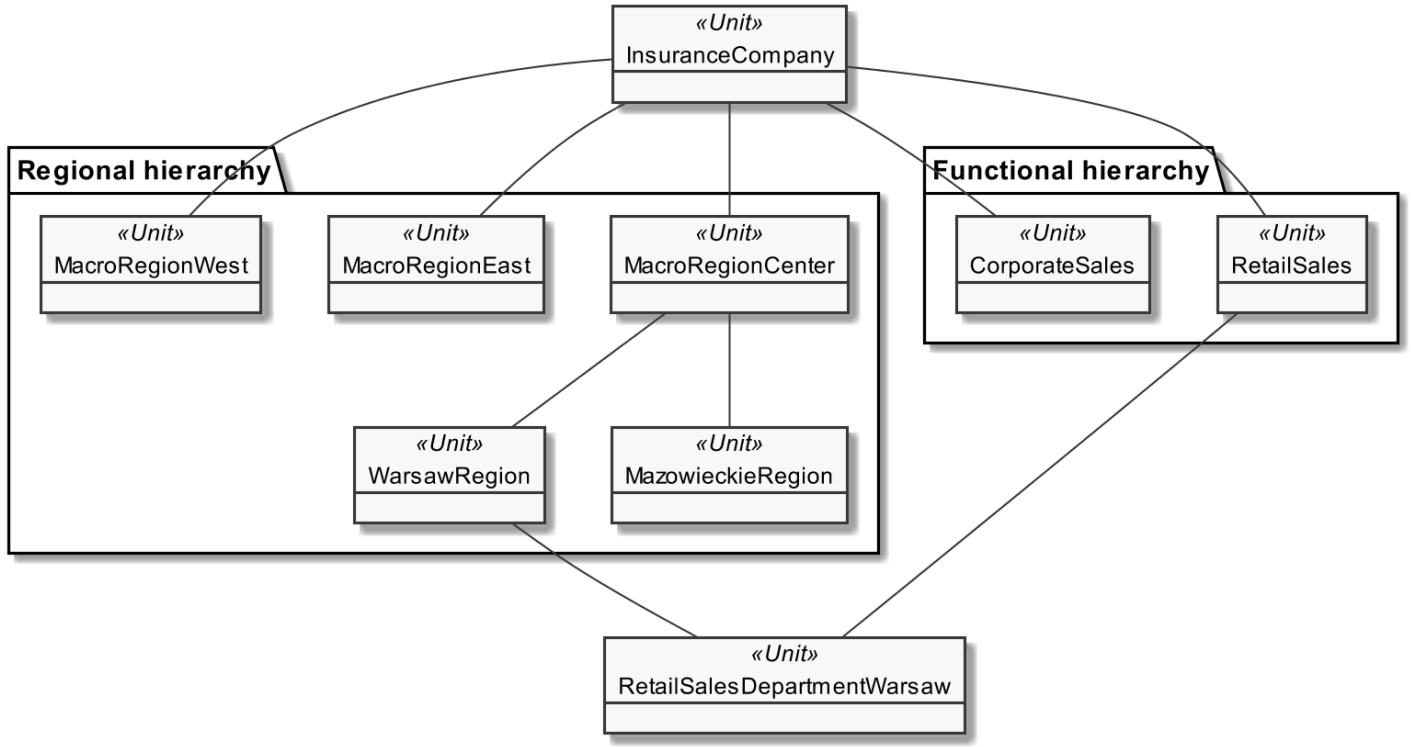
Composite - reprezentowanie wielu, nakładających się hierarchii

Zakodowanie wprost:



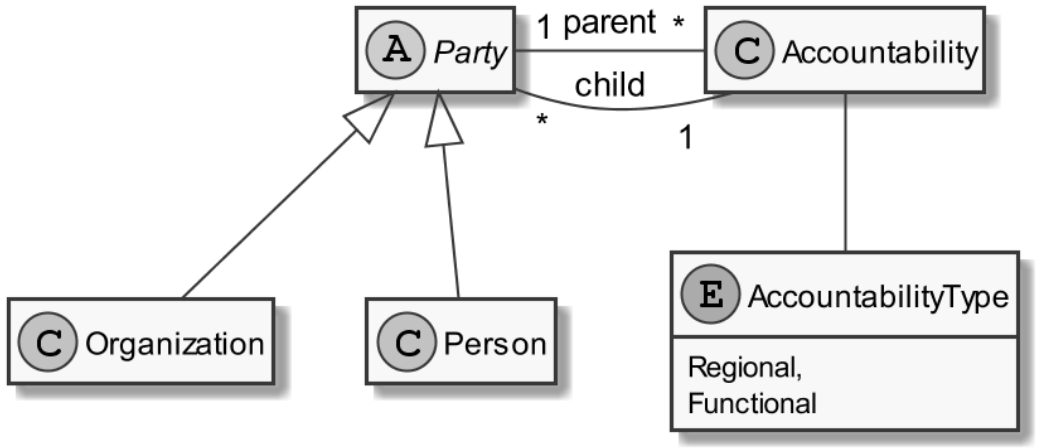
Composite - reprezentowanie wielu, nakładających się hierarchii

Zakodowanie wprost:



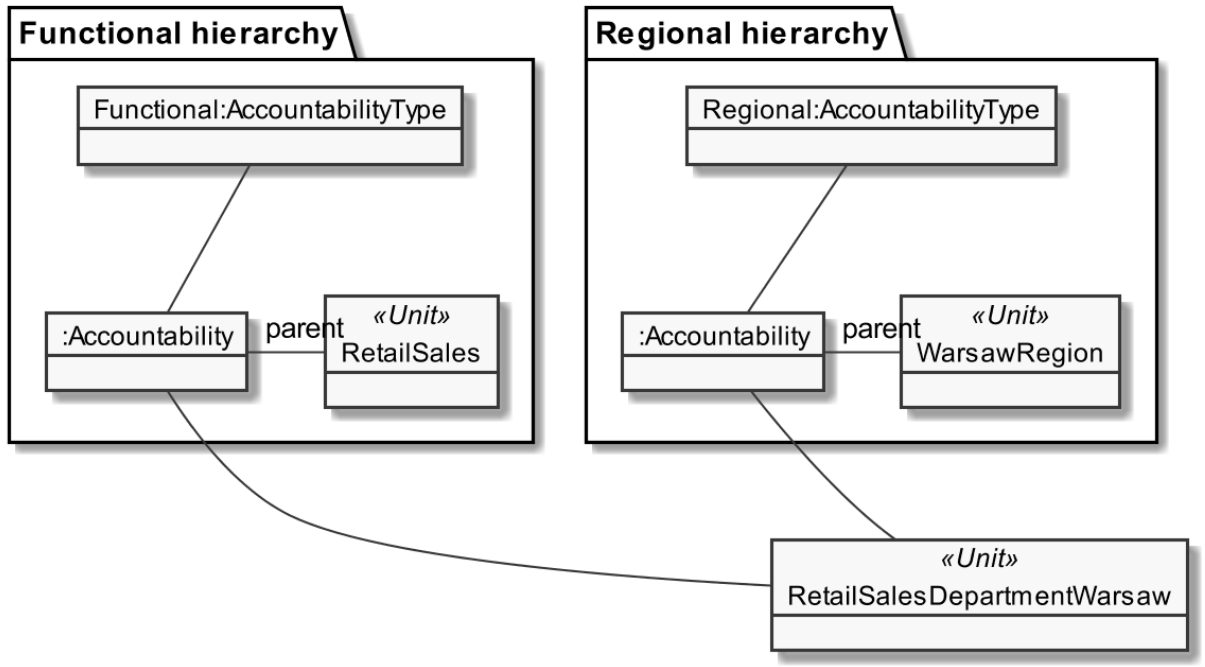
Composite - reprezentowanie wielu, nakładających się hierarchii

Wzorzec pośrednika **Accountability** (Odpowiedzialność)



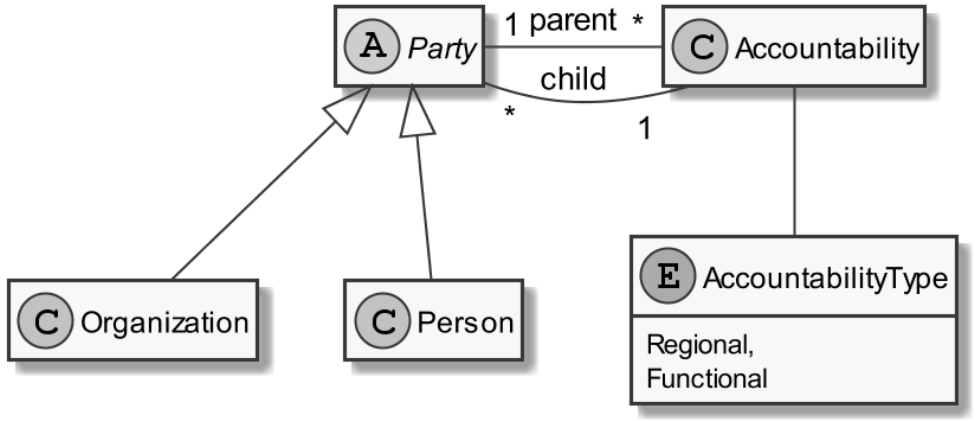
Composite - reprezentowanie wielu, nakładających się hierarchii

Wzorzec pośrednika **Accountability** (Odpowiedzialność)



Composite - reprezentowanie wielu, nakładających się hierarchii

Wzorzec pośrednika **Accountability** (Odpowiedzialność)



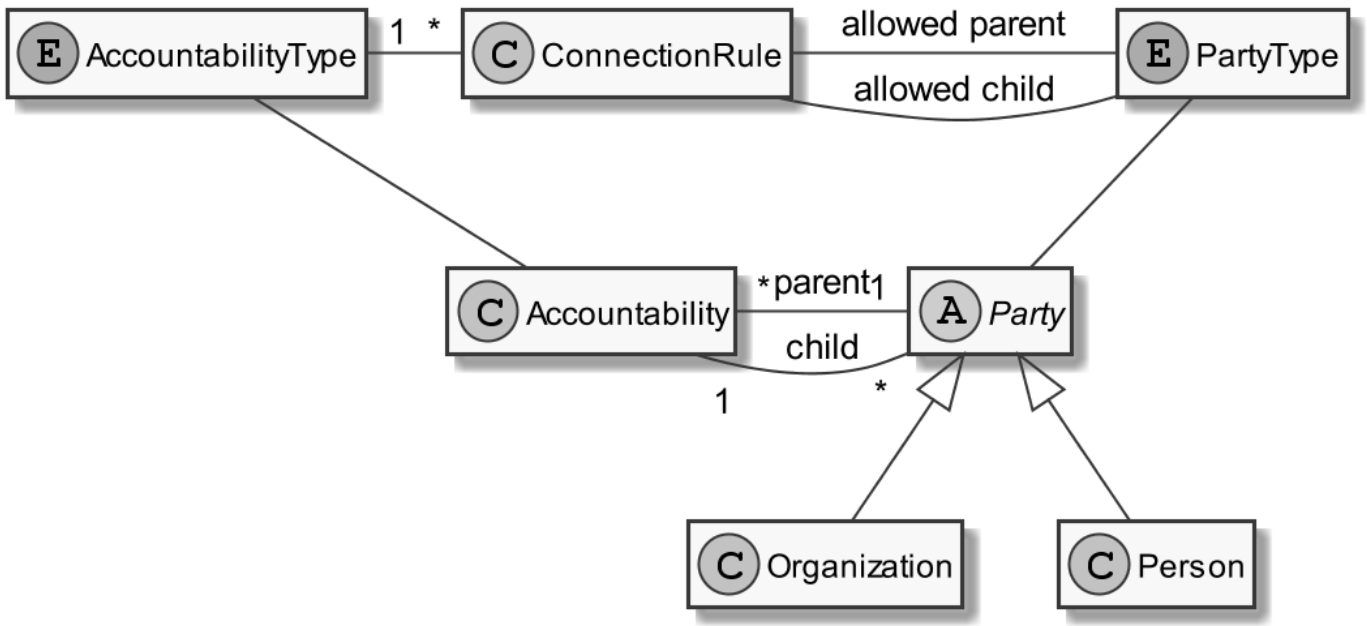
Duża elastyczność:

- **Kowalski** *pracuje* dla **IBM**
- **Kowalski** *jest menedżerem* **zespołu Zarządzania Drukarkami**
- **dr Babacki** *świadczy usługi* w **szpitalu Bielańskim**
- **Kowalski** *zgadza się* na operację przeprowadzoną przez dr Babackiego
- **szpital Bielański** *ma kontrakt* dla **IBM**
- ...

Tak, a la baza grafowa.

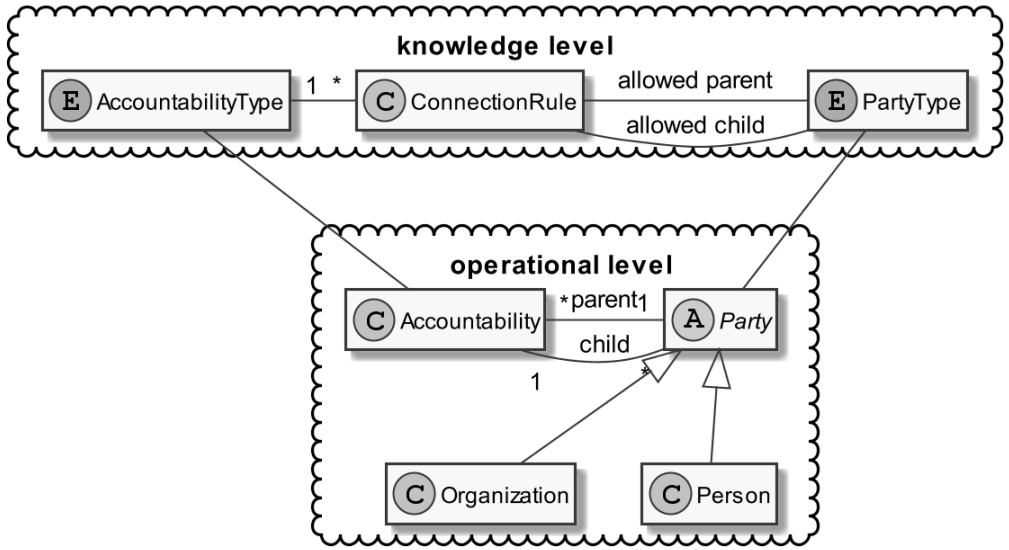
Composite - reprezentowanie wielu, nakładających się hierarchii

Wzorzec pośrednika **Accountability** oraz **Knowledge Level**



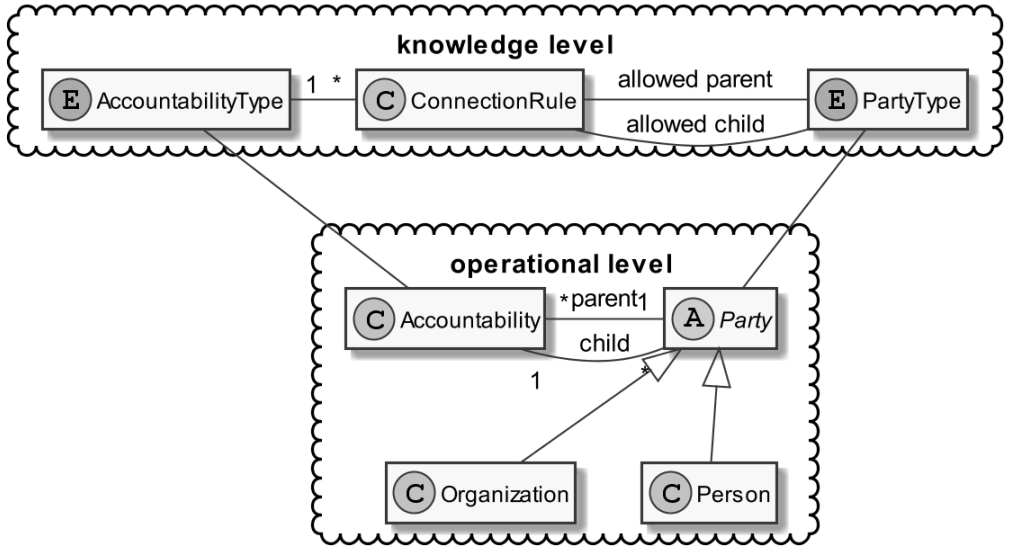
Composite - reprezentowanie wielu, nakładających się hierarchii

Wzorzec pośrednika **Accountability** oraz **Knowledge Level**



Composite - reprezentowanie wielu, nakładających się hierarchii

Wzorzec pośrednika **Accountability** oraz **Knowledge Level**



Czyli sformalizowane reguły połączeń:

- podtypy **`Party`** być może niepotrzebne (mamy **`PartyType`**)
- **`PartyType`** mogą tworzyć hierarchię

Composite - przykłady