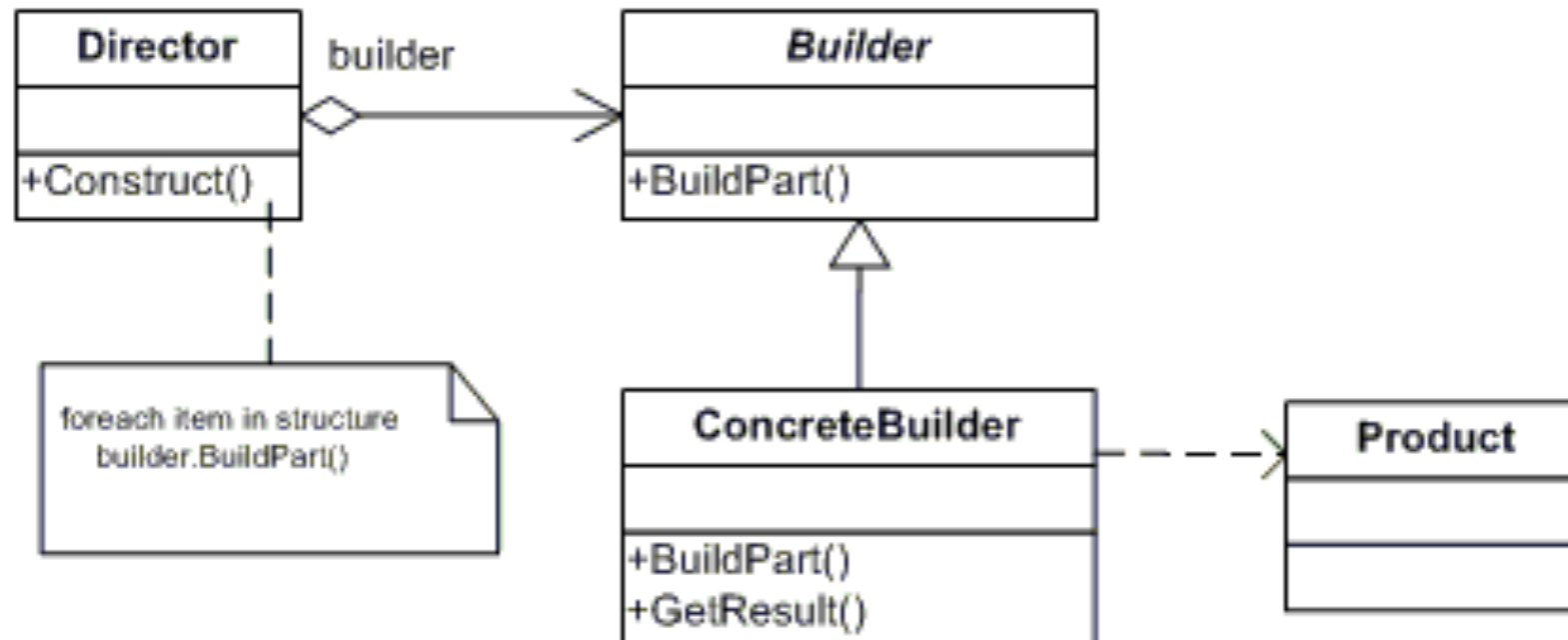


Návrhové vzory II

OMO, LS 2014/2015

Builder

Aneb: Dosazujeme do vzorečku

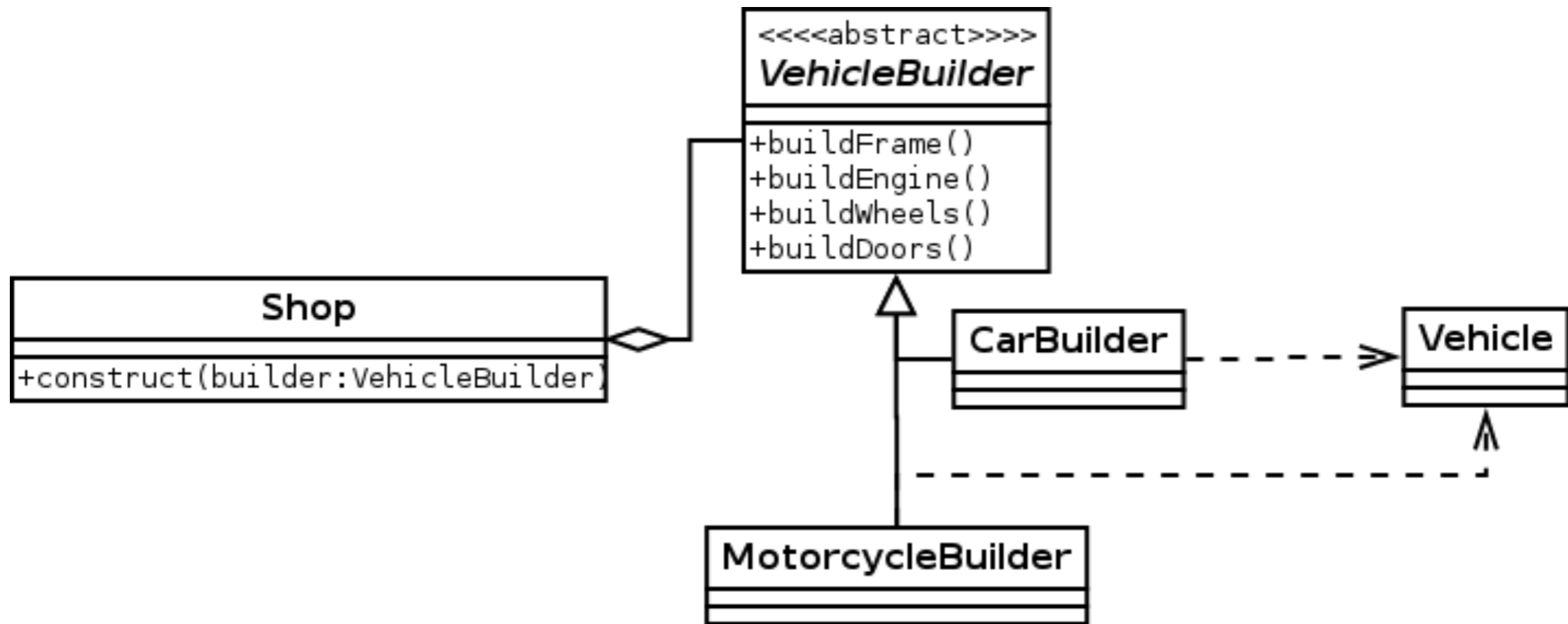


Převzato z [3]

Builder - Vlastnosti

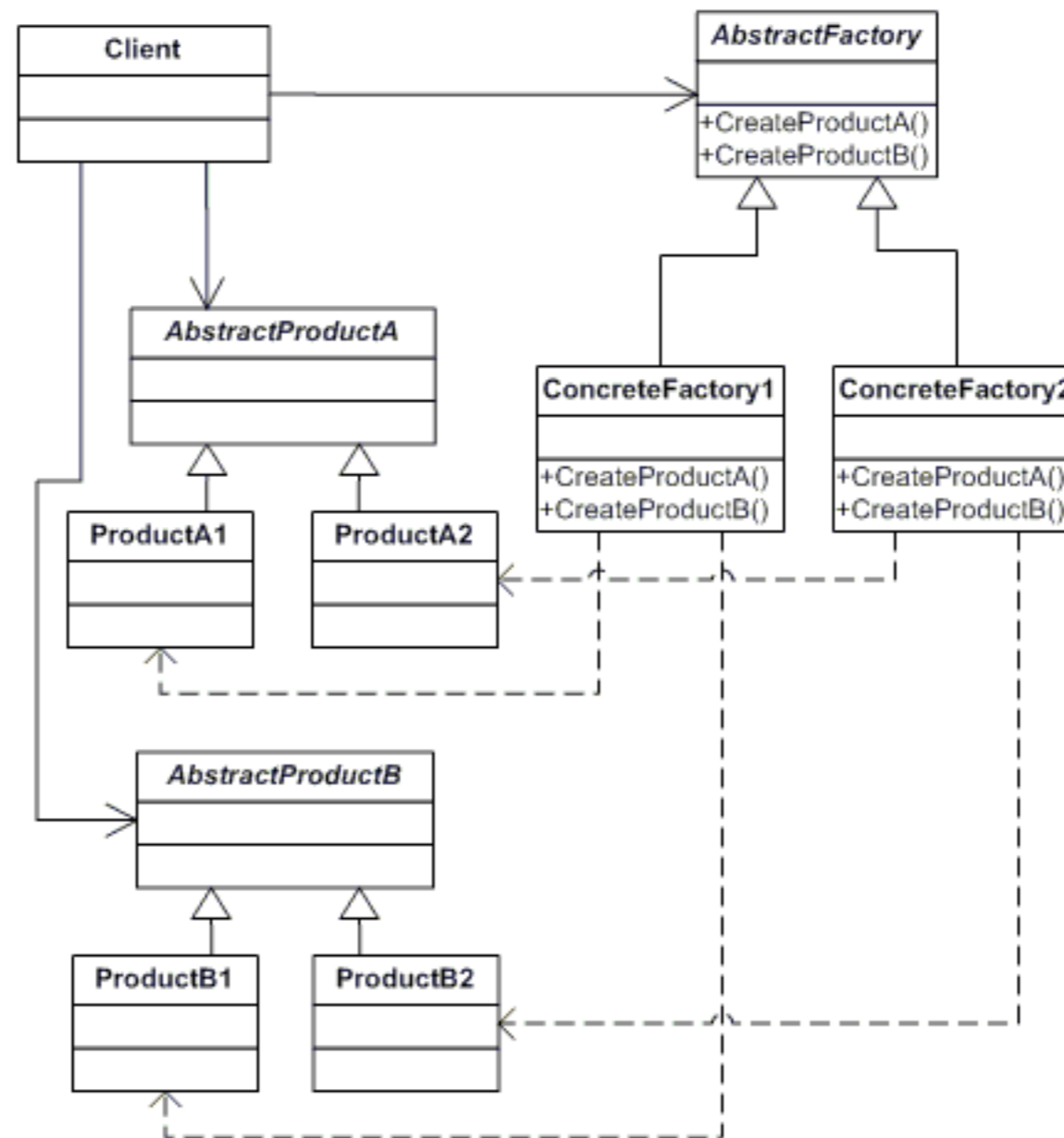
- Odděluje proces tvorby složitého objektu od jeho vnitřní reprezentace.
- Například auto se vždy bude skládat z kol, karoserie, sedaček, ale každé auto má jiné komponenty.

Builder - Příklad



Abstract Factory

Aneb: Bude toho víc



Převzato z [3]

Abstract Factory - Vlastnosti

- Poskytuje rozhraní pro vytváření příbuzných nebo závislých objektů.
- Není nutno specifikovat konkrétní třídy (realizováno až v potomcích).
- Odděluje tedy klienta od samotného procesu vytváření instancí.

Abstract Factory – Příklad

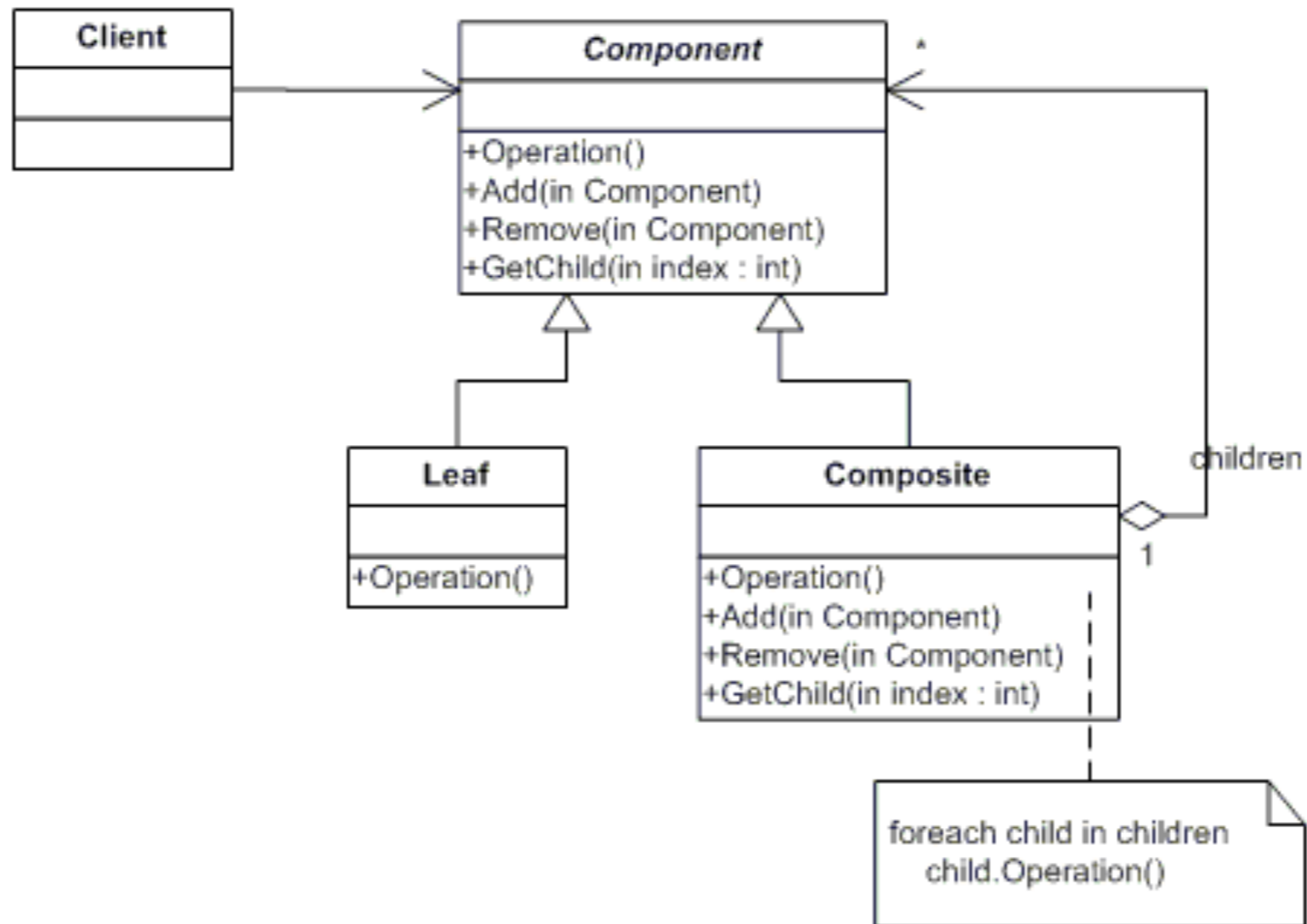
- Problém: Potřebujeme navrhnout objekt, který bude “obsahovat” následující informace na základě požadavků uživatele:
 1. Výrobce telefonu:
 - a. Nokia
 - b. Samsung
 - c. HTC
 2. Typ telefonu:
 - a. Chytrý telefon
 - b. Hloupý telefon

Abstract Factory - Příklad

- Na základě těchto informací tedy potřebujeme tři továrny výrobců (HTC, Nokia, Samsung) telefonů a dvě množiny produktů (smart, dumb).

Composite

Aneb: Stromová struktura

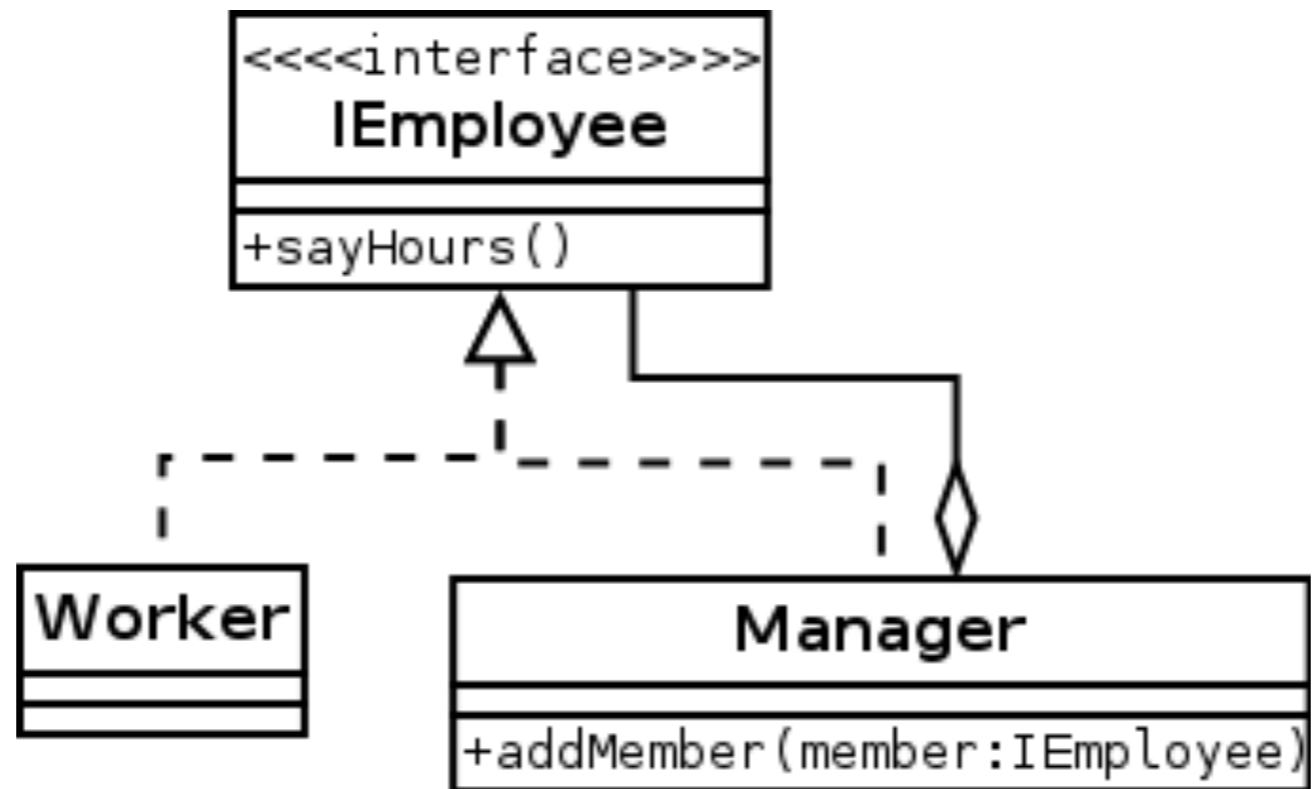


Převzato z [3]

Composite - Vlastnosti

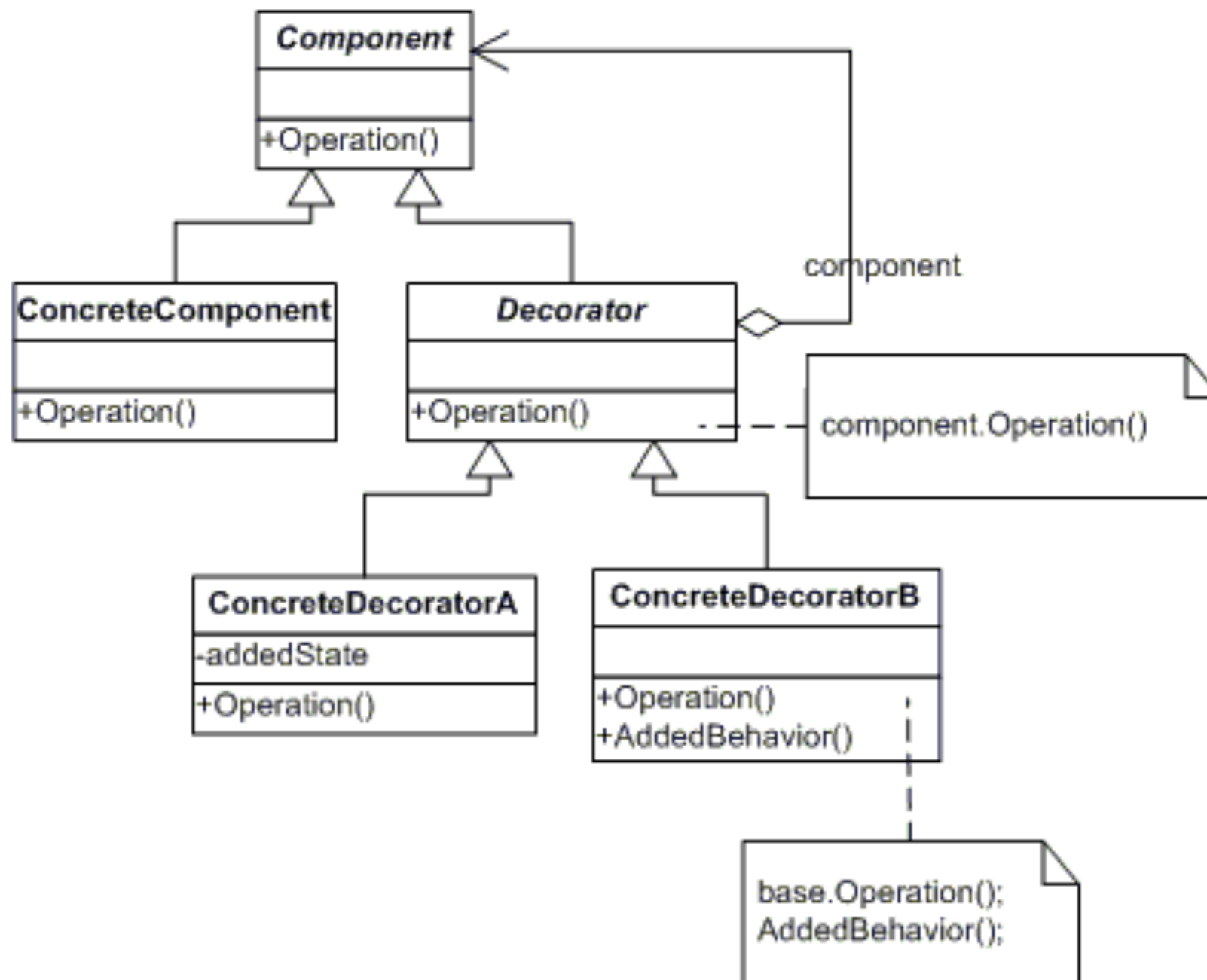
- Slouží k reprezentaci stromové struktury objektů.
- Výhodou je, že rozhraní jednotlivého uzlu i celkou jsou stejná.
- Je velice vhodný například pro reprezentaci struktury grafických objektů, souborového systému apod.

Composite - Příklad



Decorator

Aneb: Příliš mnoho druhů tříd

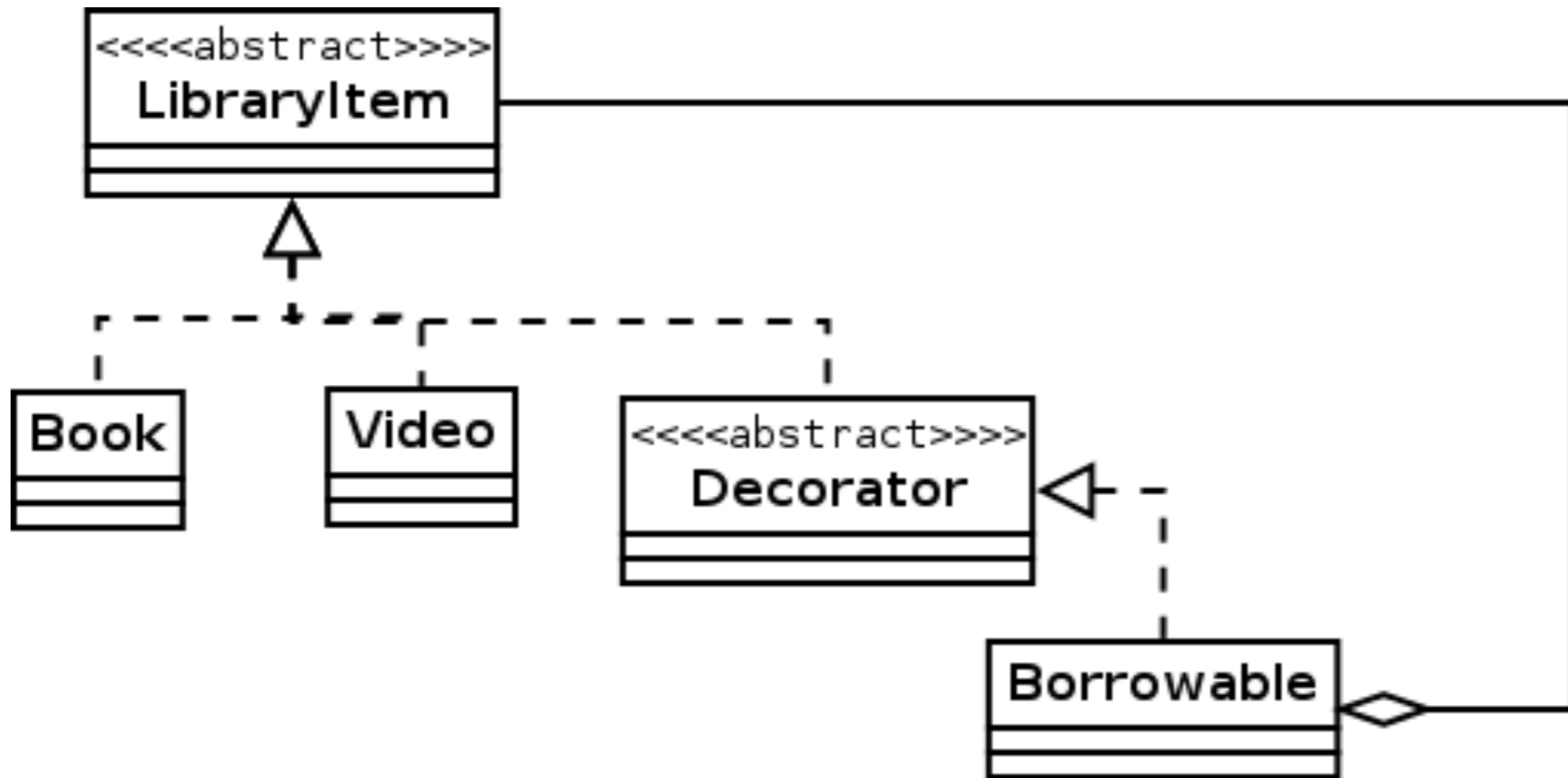


Převzato z [3]

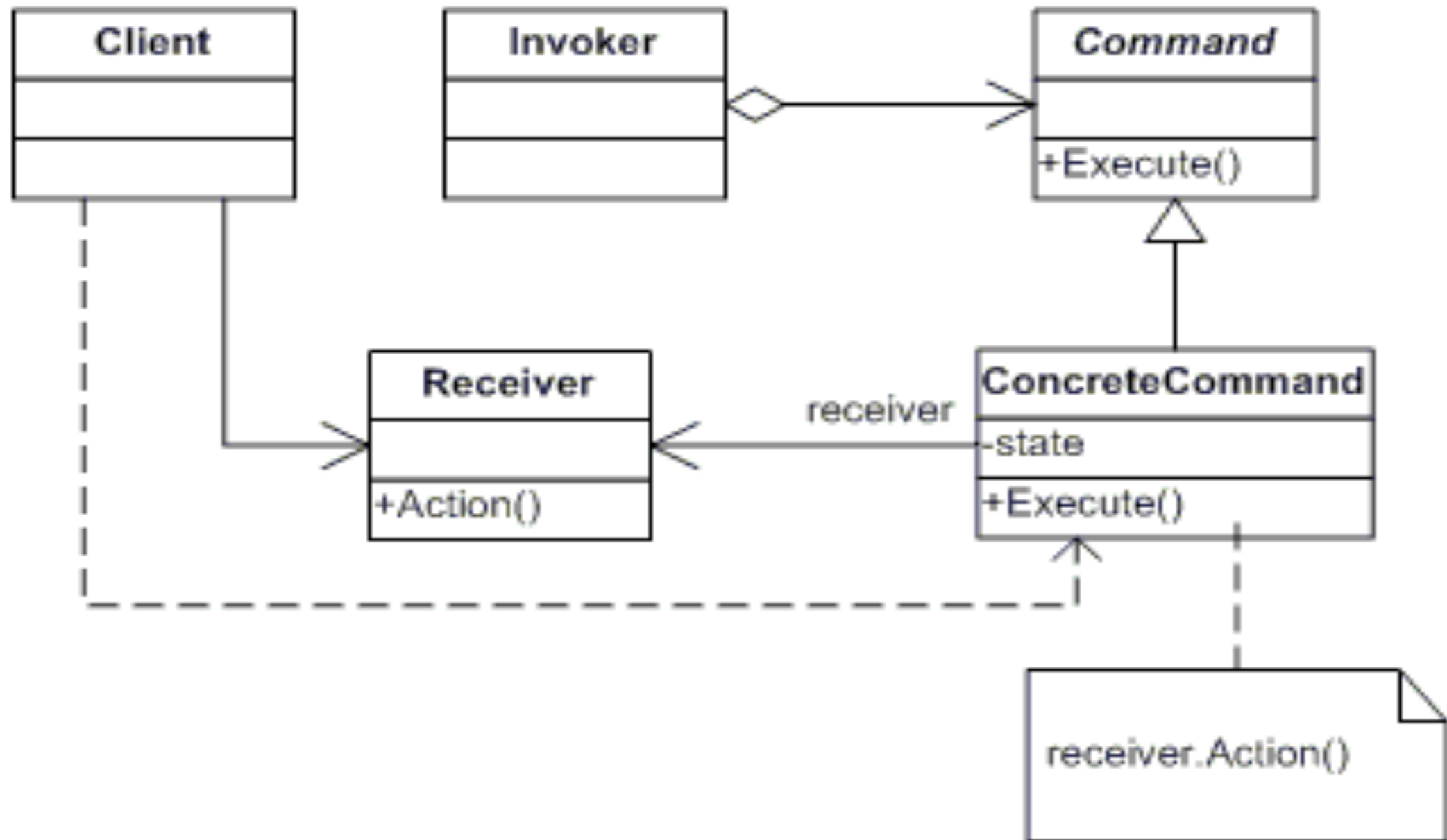
Decorator - Vlastnosti

- Umožňuje dynamicky přidávat další vlastnosti k dekorovanému objektu.
- Může poskytovat dobrou alternativu k dědění.
- Konstruktor dekorátoru převezme dekorovaný objekt jako svůj parametr a ozdobí jej funkčností.

Decorator - Příklad



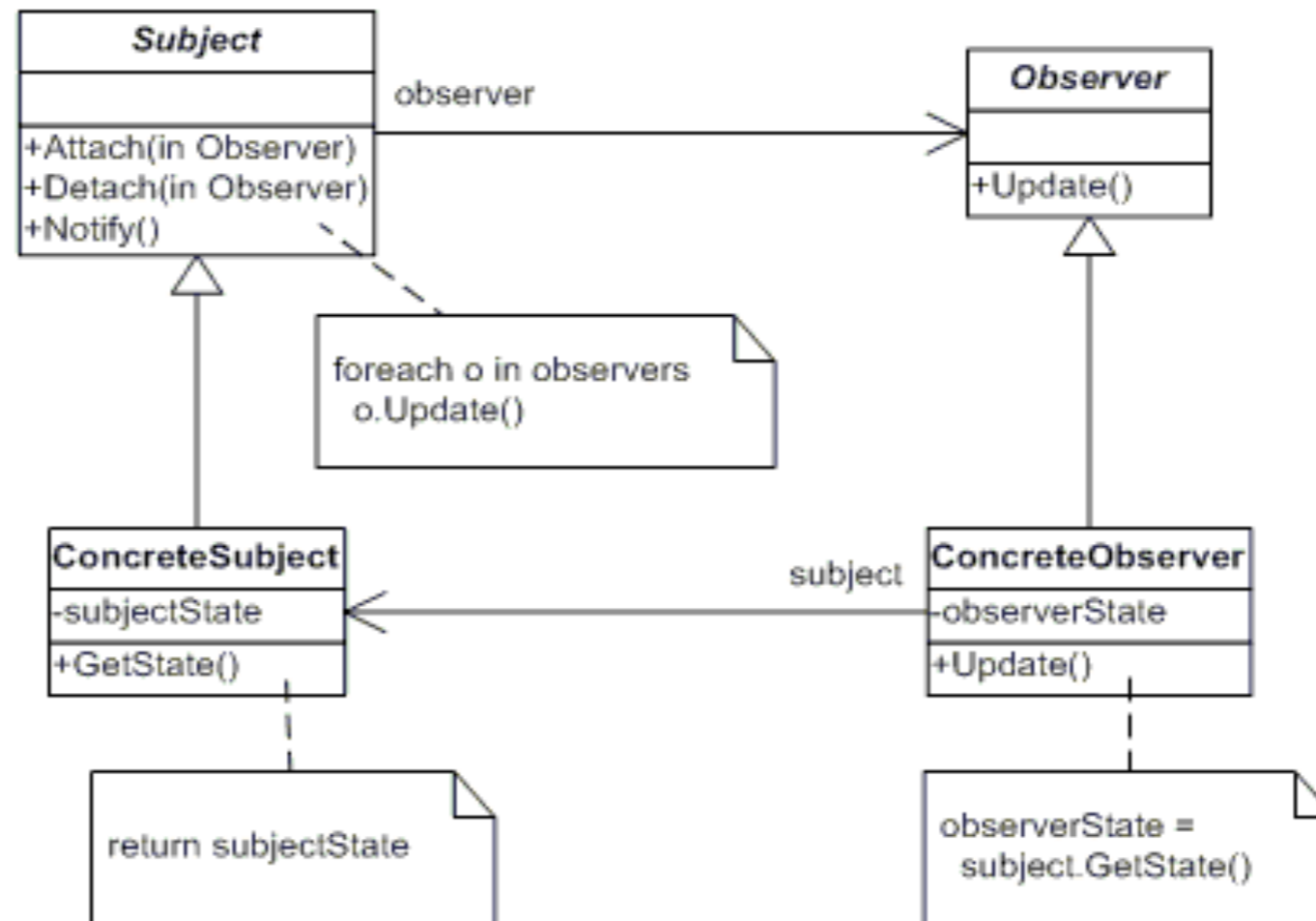
Command



Command

- Představuje zapouzdření požadavku.
- Velmi vhodné k použití při vícevláknovém zpracování pomocí fronty.
- Příkladem může být např. `EventQueue` v Javě.

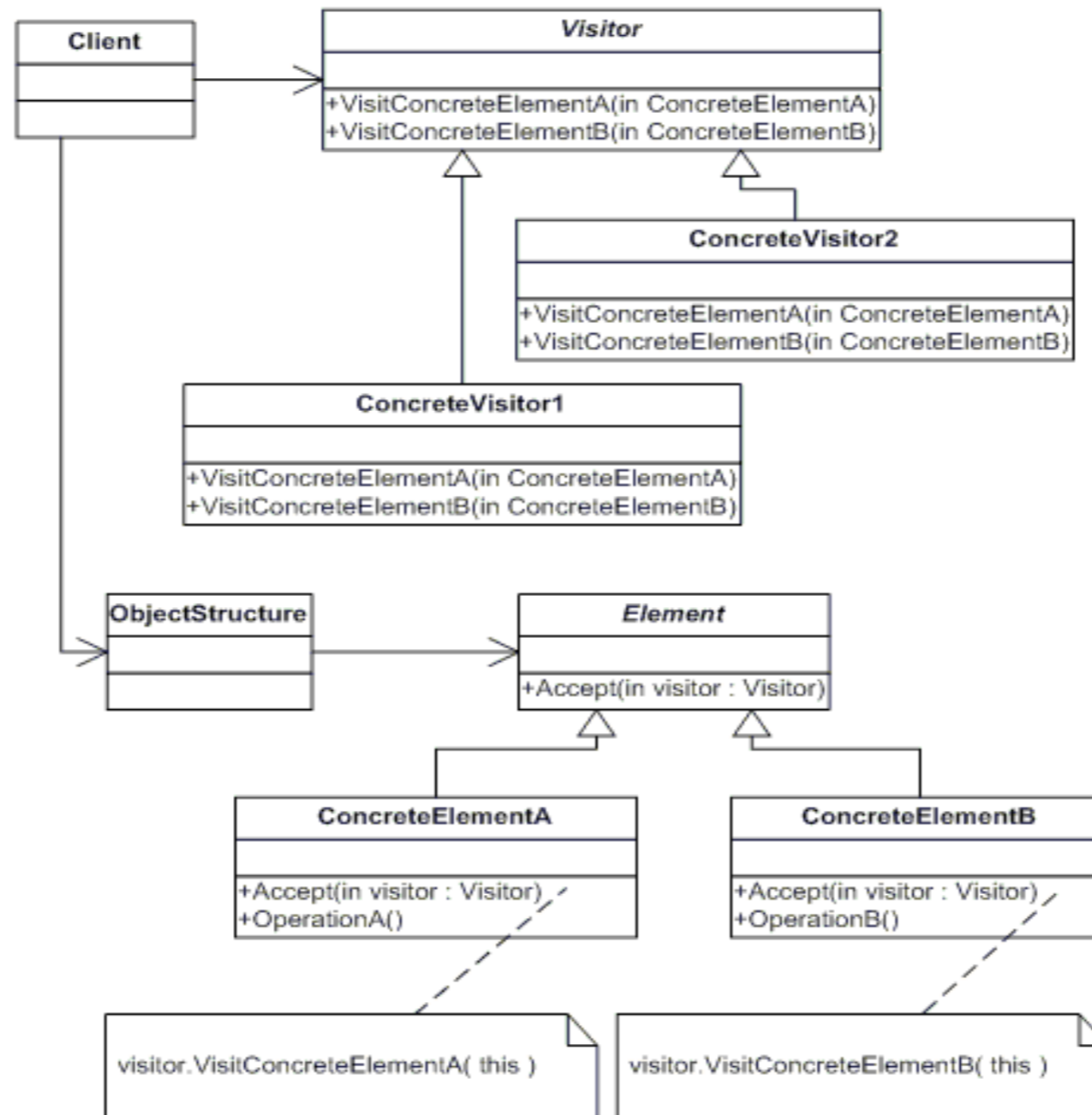
Observer



Observer

- Často slouží k poslání události o změně registrovaným pozorovatelům.
- Jedná se v podstatě o implementaci callbacku.
- Lze použít například i k logování daných událostí v systému.

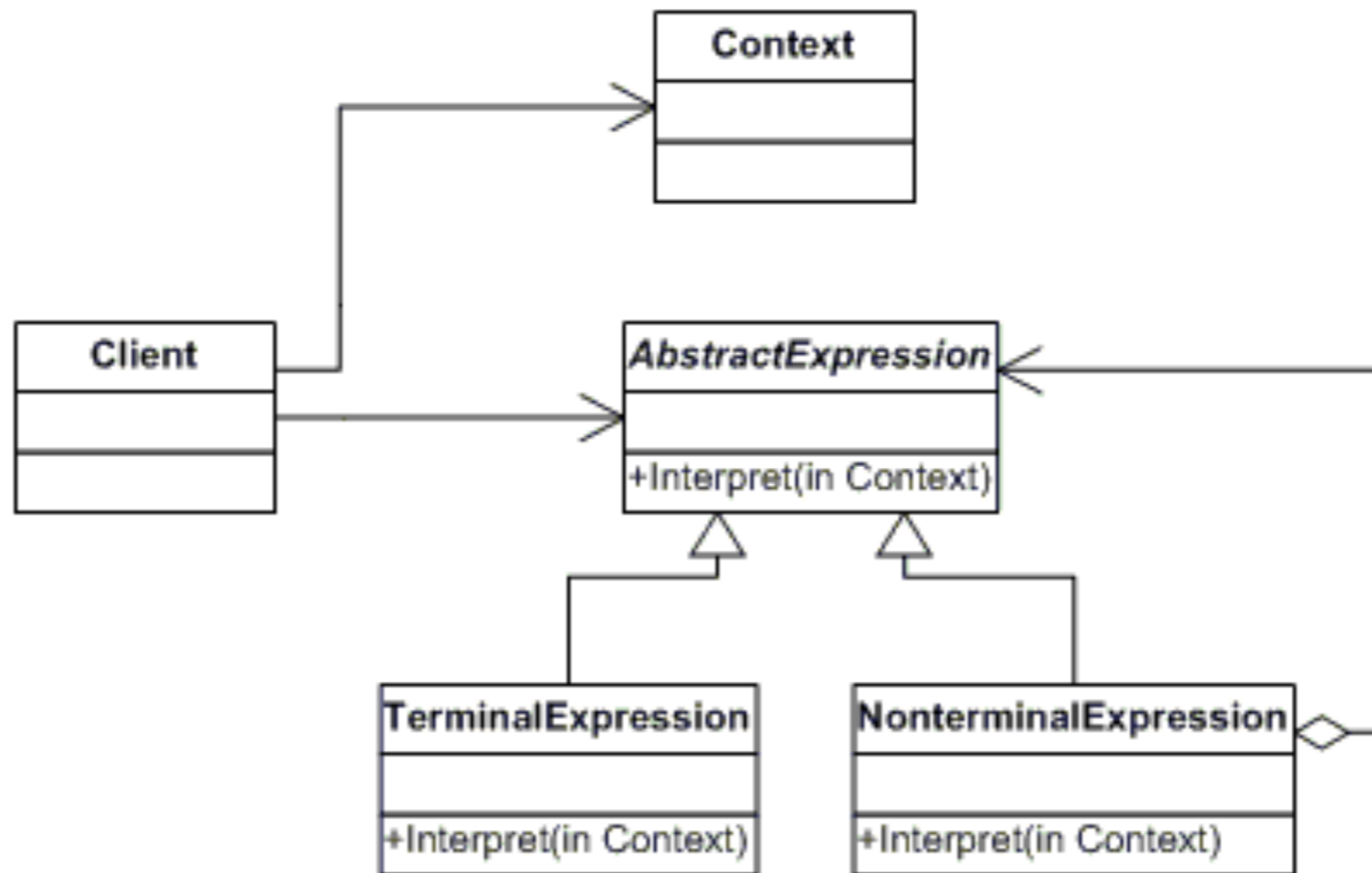
Visitor



Visitor

- Slouží k implementaci operace nad strukturou objektů, aniž bychom museli strukturu kvůli tomu měnit.
- Přidání další operace nad strukturou vyžaduje pouze implementaci nového visitoru.
- Používá principu double dispatch.

Interpreter



Interpreter

- Slouží k implementaci interpretru jazyka.
- Na základě gramatiky je vybudován AST a ten je interpretován.
- Velmi obecný návrhový vzor.
- Může se hodit například pokud chcete k vašemu systému poskytnout DSL.

Intepreter – Příklad

- Navrhněte interpreter matematických výrazů.
- Interpreter bude obsahovat následující operace:
 - Plus
 - Multiply
 - Minus
- Bude rozlišovat dva typy operandů – proměnné a hodnoty.
- Parametrem metody interpret bude Context v kterém budou uloženy hodnoty proměnných.

Literatura

- [1] Gamma, E., Helm, R., Johnson, R., Vlissides, J. **Design Patterns: Elements of Reusable Object-Oriented Software**, Addison-Wesley
- [2] Pecinovský, R. **Návrhové vzory**, Computer Press
- [3] <http://www.dofactory.com/Patterns/Patterns.aspx>
- [4] <https://csharpdesignpatterns.codeplex.com/>