



Programovanie v jazyku C#

Behaviorálne návrhové vzory

prednáška 12
Ing. Ján Magyar, PhD.
ak. rok. 2024/2025 ZS

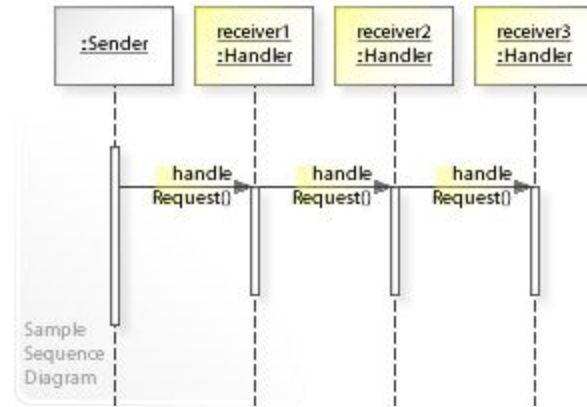
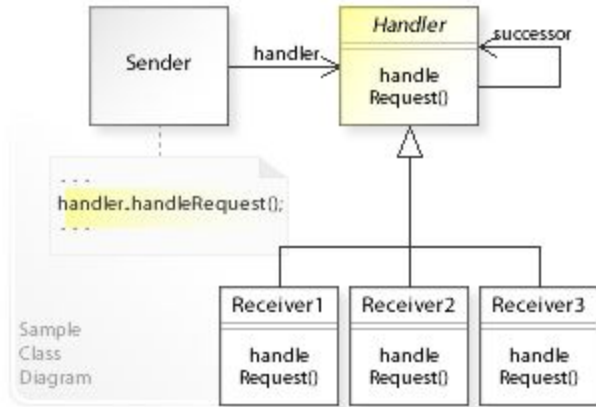
Behaviorálne vzory

- chain of responsibility
- command
- iterator
- mediator
- observer a publish/subscribe
- strategy
- visitor

Chain of responsibility

- viac objektov môže spracovať požiadavku
- vytvoríme zreteženie procesorov a správa sa posielá ďalej kým nie je spracovaná (alebo nie je spracovaná vôbec)
- máme zdroj command objektov a sériu procesorov
- každý procesor buď spracuje požiadavku alebo ju prepošle na základe podmienok počas behu

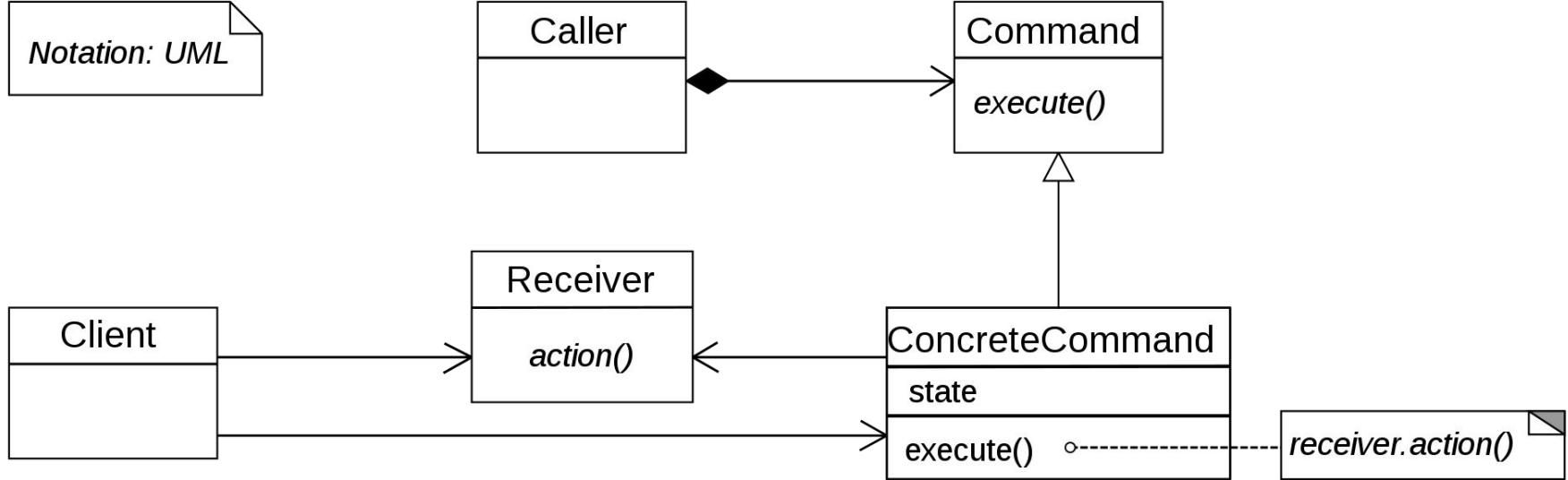
Chain of responsibility



Command

- enkapsulácia požiadavky do objektu
- umožňuje parametrizáciu klientov s rôznymi požiadavkami
- pre queueing, logging, a nezvratiteľné operácie
- požiadavka je delegovaná do command objektu namiesto priameho spracovania
- napr. GUI buttons, progress bars, transactions, wizards

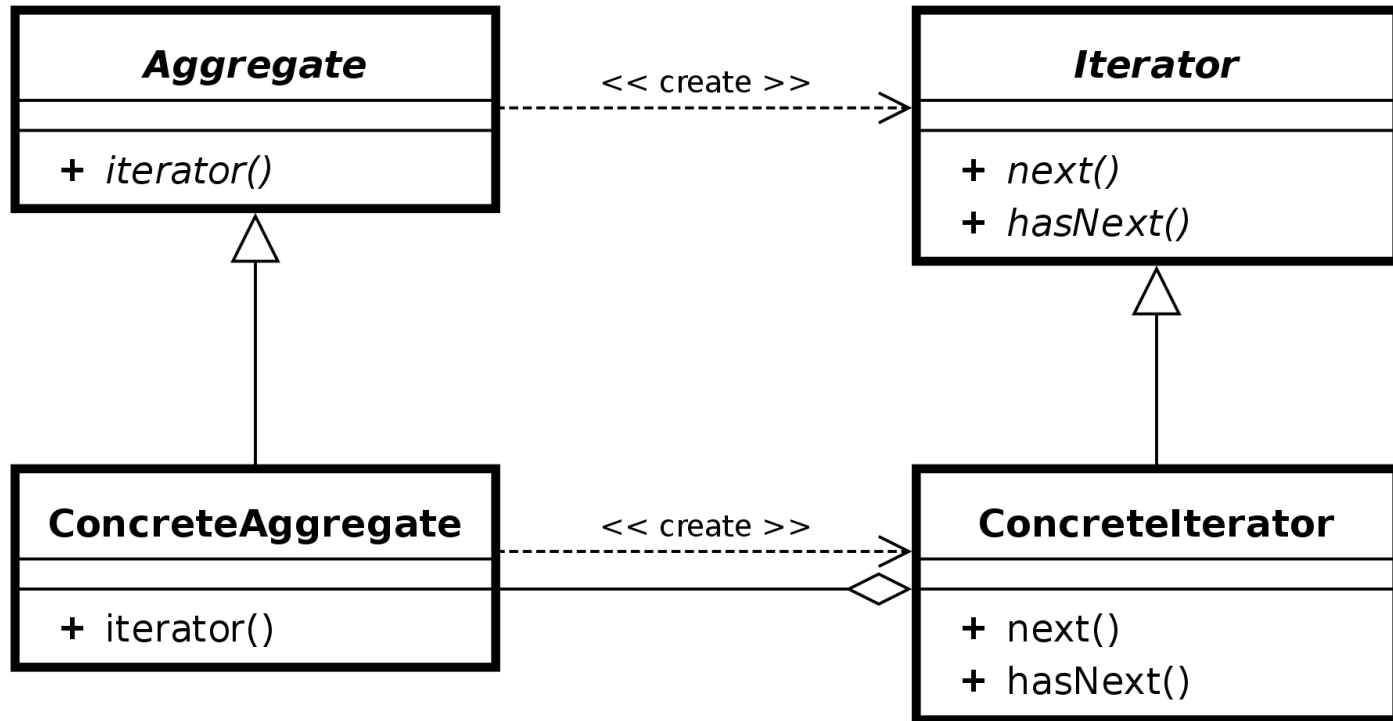
Command



Iterator

- poskytuje spôsob prístupu k prvkom agregátu sekvenčne bez odhalenia vnútornej reprezentácie
- oddelí algoritmus od kontajnera
- aj keď poskytuje jednotný prístup, nie vždy je optimálny
- trieda iterátor zapuzdrí prístup a prechádzanie agregátom, klient používa iterátor

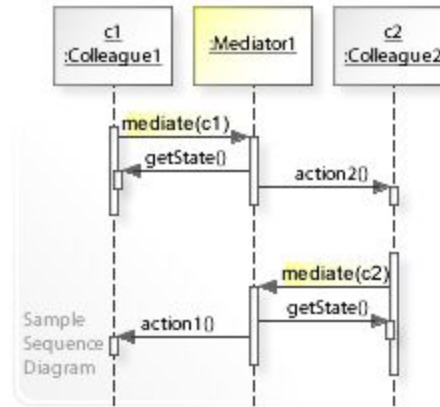
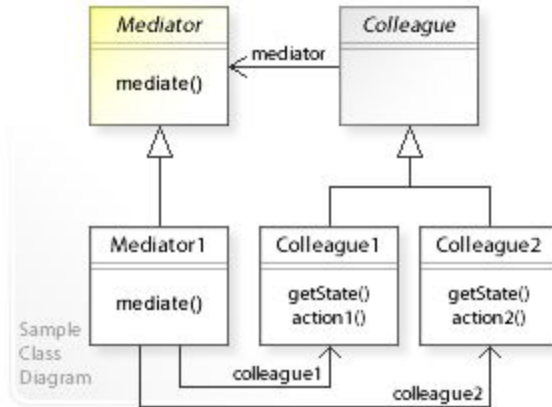
Iterator



Mediator

- enkapsulácia interakcie medzi sadou objektov
- umožňuje loose coupling medzi objektmi, keďže neodkazujú jeden na druhý explicitne
- môžeme meniť interakciu objektov
- zdefinujeme triedu mediátor, ktorá sa potom používa pre komunikáciu medzi objektmi

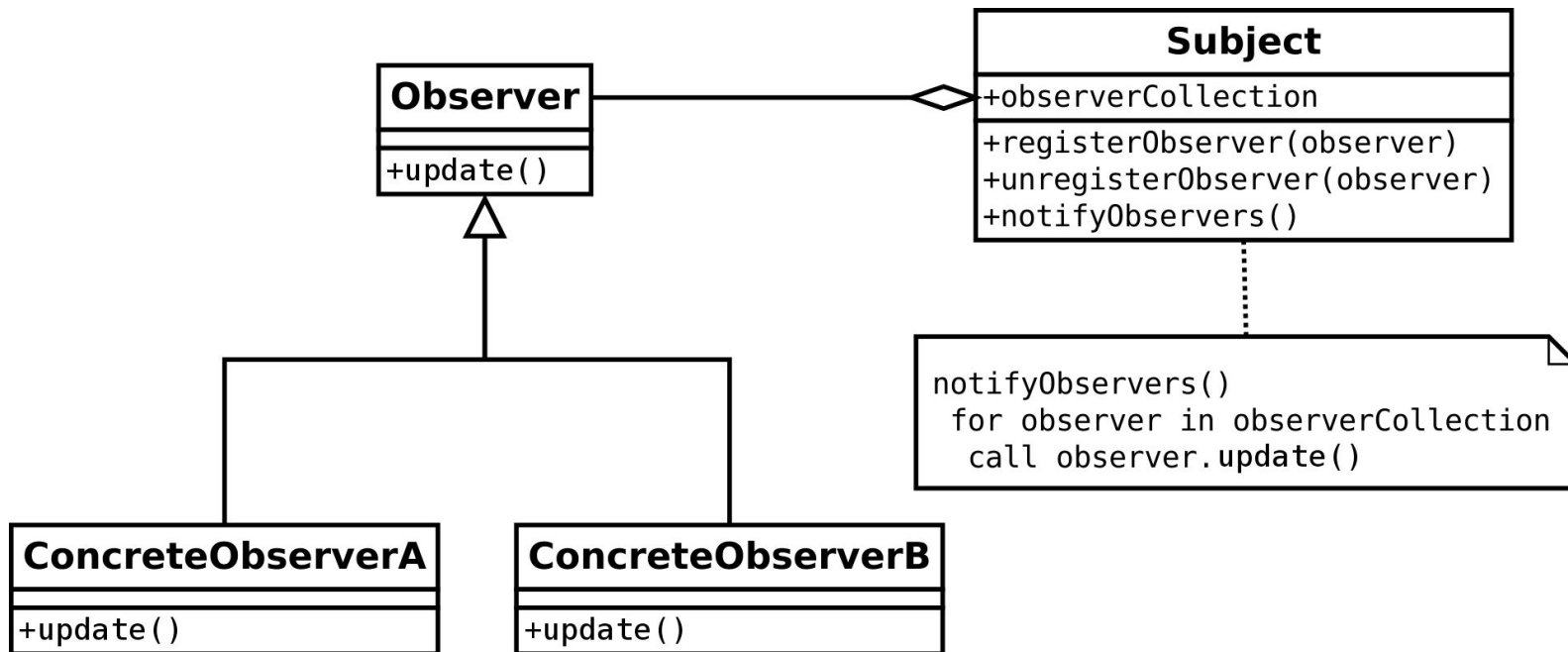
Mediator



Observer

- definícia závislosti one-to-many
- zmena jedného objektu vyžaduje notifikáciu ďalších závislých objektov
- implementuje vzor publisher/subscriber
- **subject** udržiava zoznam svojich **observerov** a notifikuje ich automaticky, tie následne zmenu spracujú vhodným spôsobom

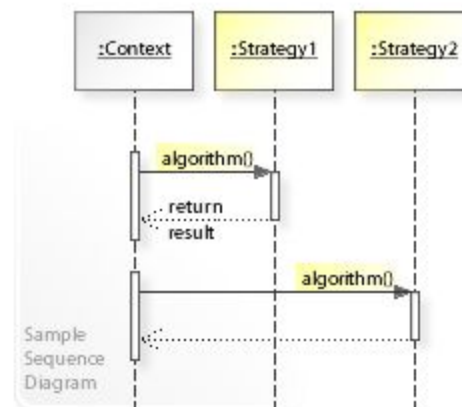
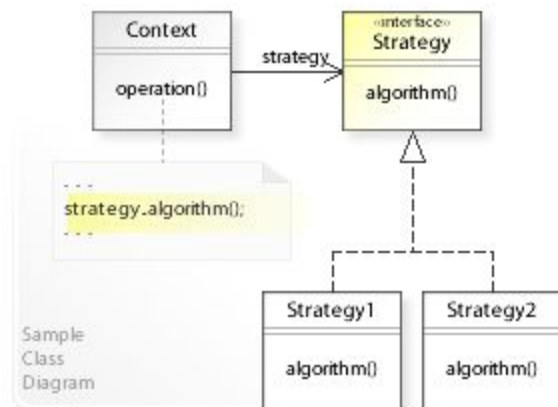
Observer



Strategy

- zdefinujeme skupinu podobných algoritmov s rovnakým rozhraním
- algoritmy potom vieme používať zameniteľne
- vyberieme algoritmus počas behu na základe podmienky
- pred zavolaním algoritmu objekt dostane informáciu o tom, ktoré riešenie má použiť a následne zavolá zodpovedajúcu metódu

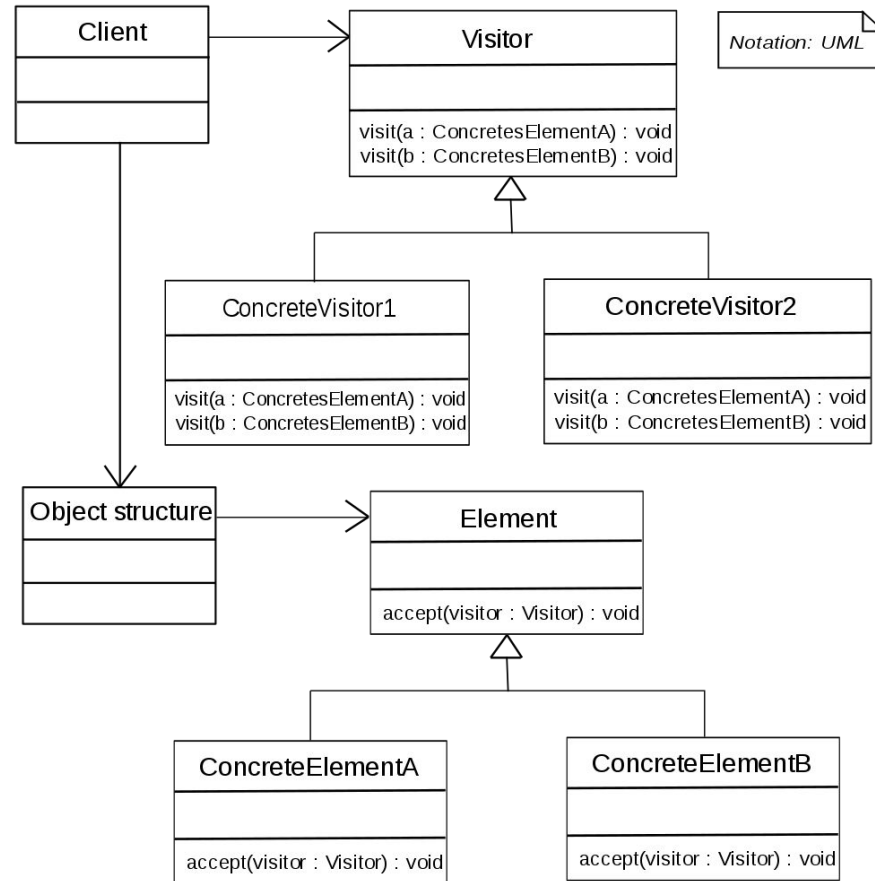
Strategy



Visitor

- pre operácie, ktoré sa vykonávajú nad prvkami štruktúry
- môžeme zadať novú operáciu bez toho, aby sme museli zmeniť triedu prvku
- oddelíme algoritmus od objektovej štruktúry ktorou pracuje
- klient neinteraguje priamo s prvkom, namiesto toho pošle visitor objekt, ktorý následne vykoná operáciu nad prvkom

Visitor



Ďalšie behaviorálne návrhové vzory

- blackboard
- memento
- null object
- servant
- specification
- state
- template method

otázky?