Programovanie v jazyku C#

Behaviorálne návrhové vzory

prednáška 12
Ing. Ján Magyar, PhD.
ak. rok. 2023/2024 ZS

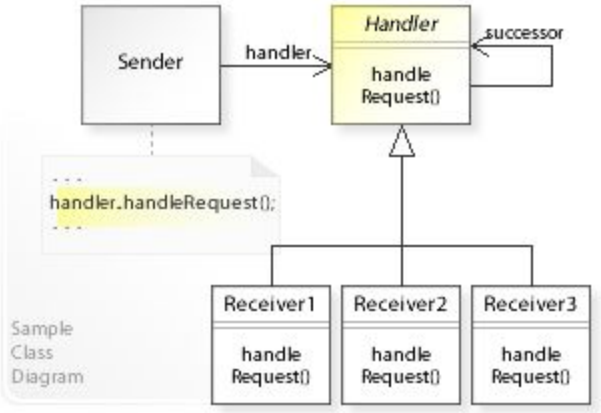
Behaviorálne vzory

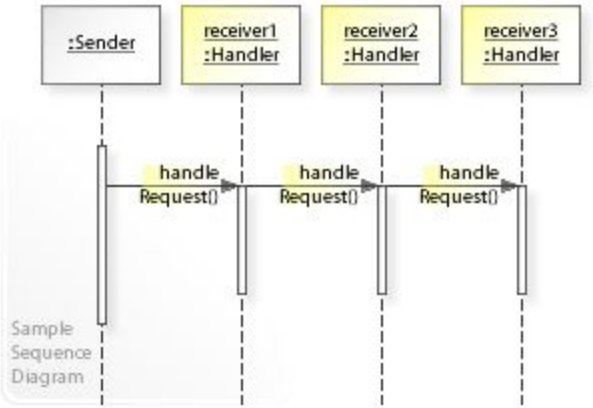
- chain of responsibility
- command
- iterator
- mediator
- observer and publish/subscribe
- strategy
- visitor

Chain of responsibility

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

Chain of responsibility

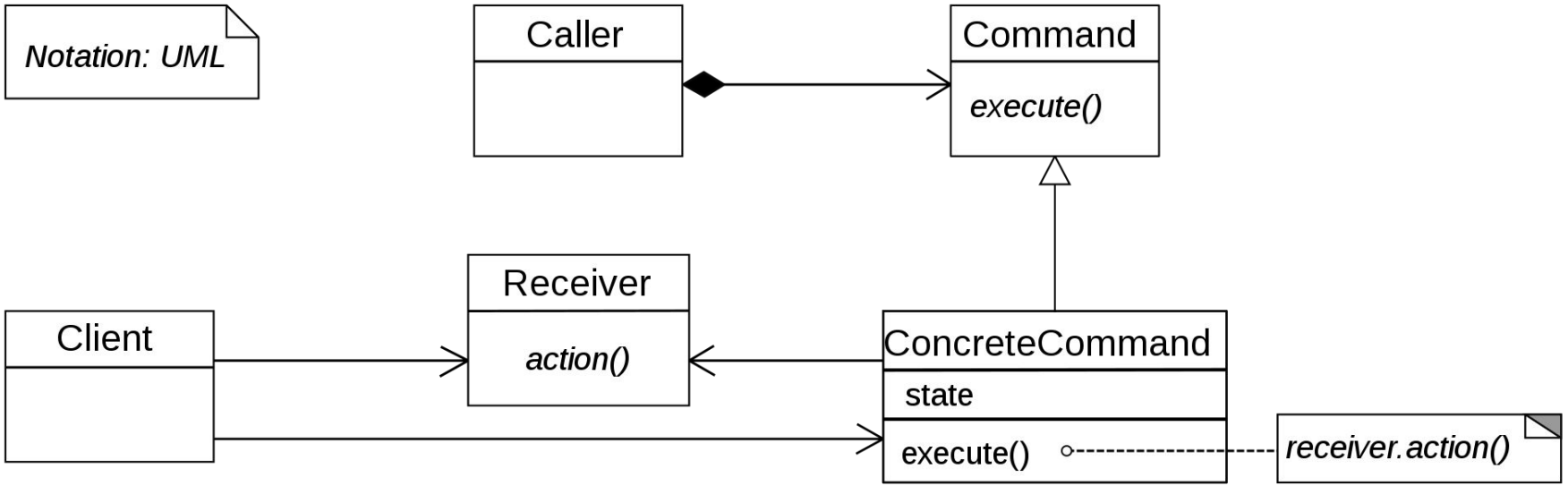




Command

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

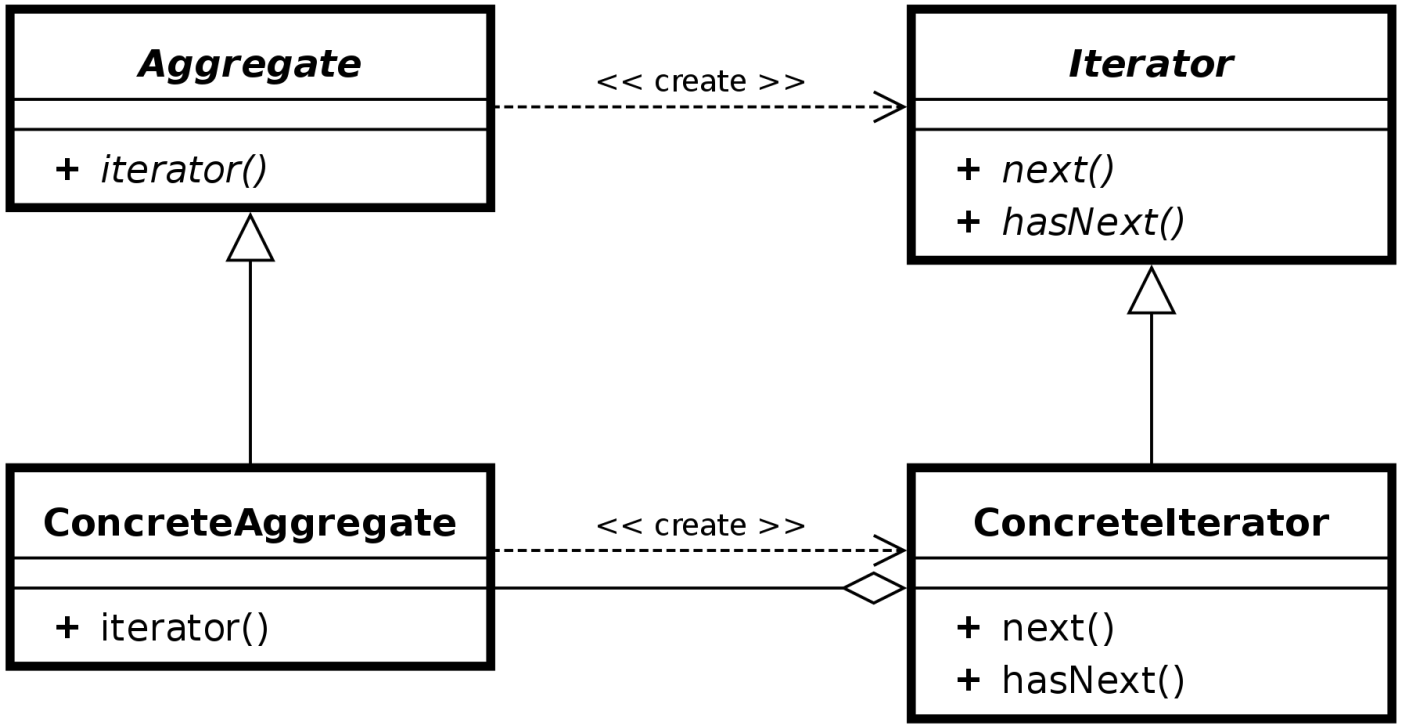
Command


Notation: UML
Caller
Command
execute()
Receiver
action()
Client
ConcreteCommand
state
execute()
receiver.action()

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

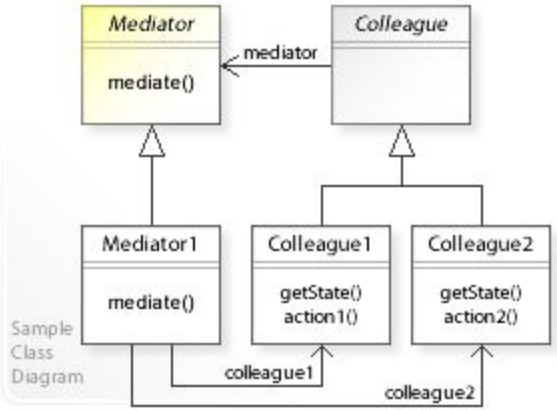
Iterator

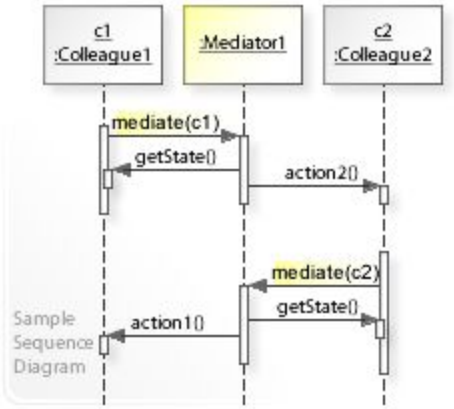

Aggregate
+ iterator()
<< create >>
Iterator
+ next()
+ hasNext()
ConcreteAggregate
+ iterator()
<< create >>
ConcreteIterator
+ next()
+ hasNext()

Mediator

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

Mediator

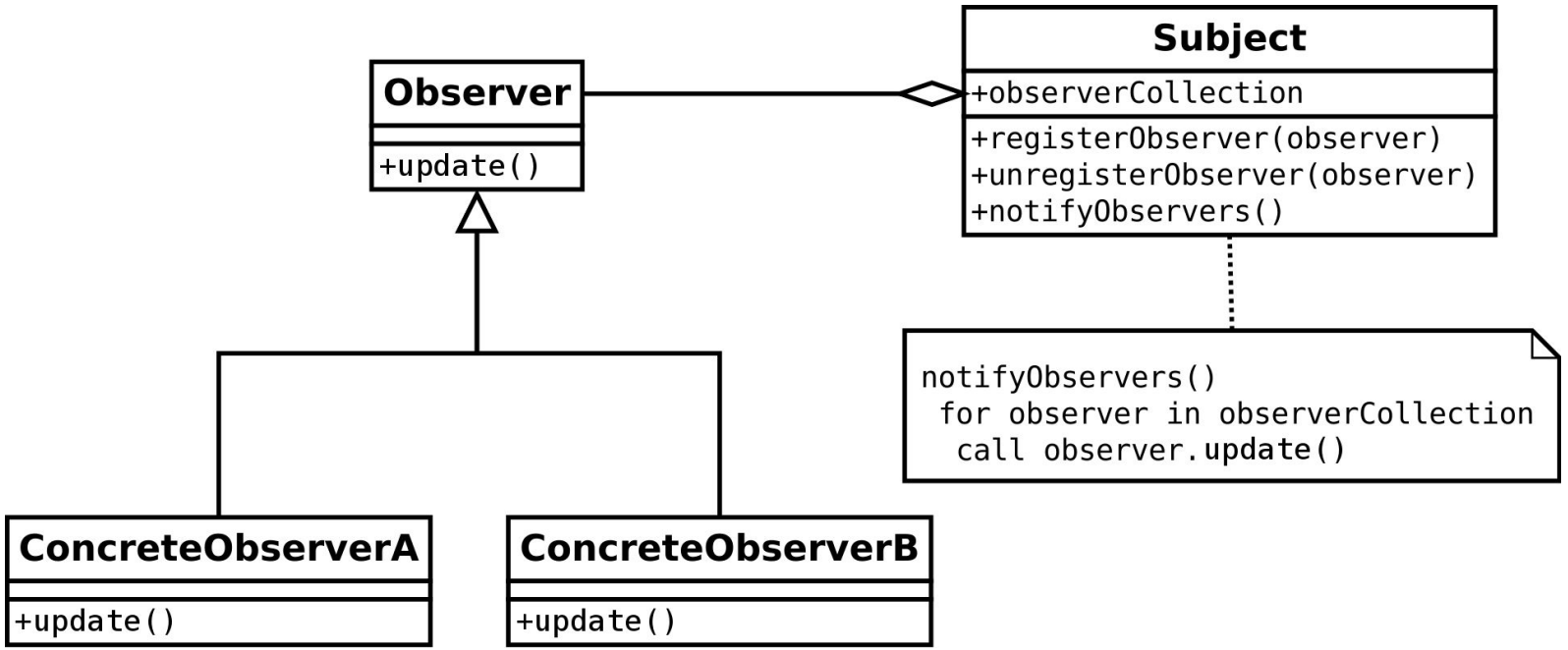

Mediator
mediate()
mediator
Colleague
Mediator1
mediate()
Colleague1
getState()
action1()
Colleague2
getState()
action2()
colleague1
colleague2
Sample
Class
Diagram


c1
:Colleague1
:Mediator1
c2
:Colleague2
mediate(c1)
getState()
action2()
mediate(c2)
getState()
action1()
Sample
Sequence
Diagram

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

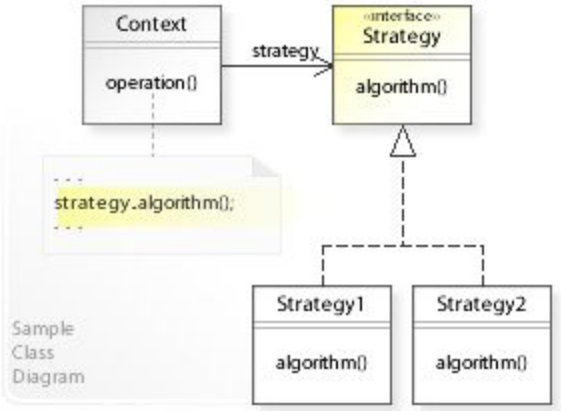
Observer

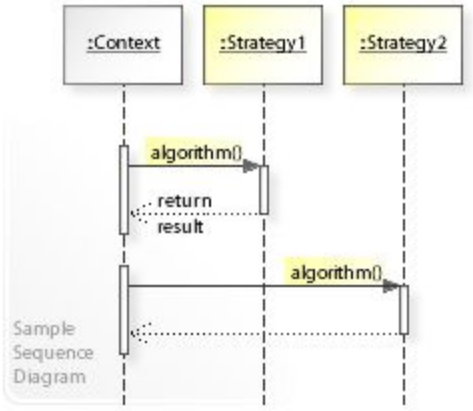

Subject
+observerCollection
+registerObserver(observer)
+unregisterObserver(observer)
+notifyObservers()
Observer
+update()
notifyObservers()
for observer in observerCollection
call observer.update()
ConcreteObserverA
+update()
ConcreteObserverB
+update()

Strategy

- zadefinujeme 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

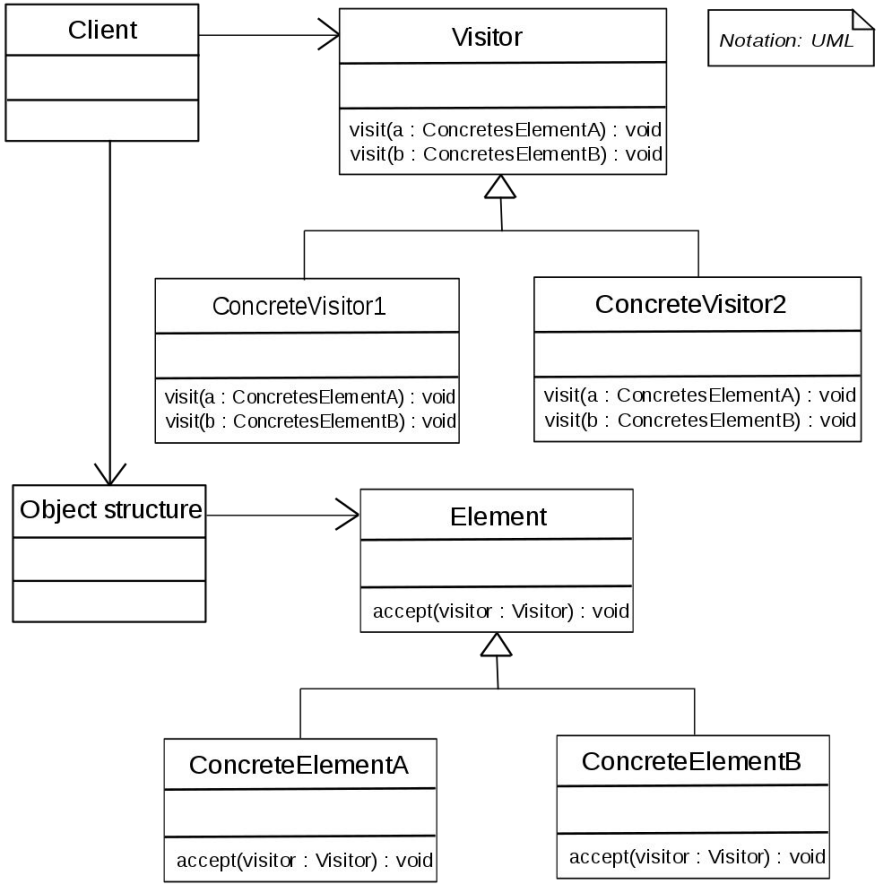

Context
operation()
strategy
«interface»
Strategy
algorithm()
strategy.algorithm();
Strategy1
algorithm()
Strategy2
algorithm()
Sample
Class
Diagram


:Context
:Strategy1
:Strategy2
algorithm()
return
result
algorithm()
Sample
Sequence
Diagram

Visitor

- pre operácie, ktoré sa vykonajú nad prvkami štruktúry
- môžeme zadefinovať 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


Client
Visitor
Notation: UML
visit(a : ConcretesElementA) : void
visit(b : ConcretesElementB) : void
ConcreteVisitor1
visit(a : ConcretesElementA) : void
visit(b : ConcretesElementB) : void
ConcreteVisitor2
visit(a : ConcretesElementA) : void
visit(b : ConcretesElementB) : void
Object structure
Element
accept(visitor : Visitor) : void
ConcreteElementA
accept(visitor : Visitor) : void
ConcreteElementB
accept(visitor : Visitor) : void

Ďalšie behaviorálne návrhové vzory

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

otázky?