

Chain of Responsibility

Software Architektur

Wintersemester 2019

WWI 17 SE B, Gruppe 3

Andreas Gülükoglu,

Felix Waage,

Fabio Westphal,

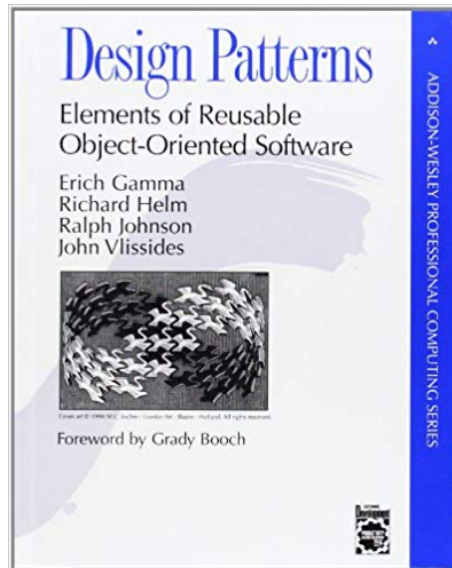
Yvonne Werner,

Milena Zahn

Agenda

- Definition und Erklärung
- Anwendung
- Vor- und Nachteile
- UML-Diagramm
- Beispiel
- Quellenangaben

Definition



Bildquelle: <https://www.amazon.de/Patterns-Elements-Reusable-Object-Oriented-Software/dp/0201633612>

„Vermeide die Kopplung des Auslösers einer Anfrage mit seinem Empfänger, indem mehr als ein Objekt die Möglichkeit enthält, die Aufgabe zu erledigen. Verkette die empfangenden Objekte und leite die Anfrage an der Kette entlang, bis ein Objekt sie erledigt.“

Erklärung

Chain of Responsibility = Verhaltensmuster; objektorientierte Version von if ... else if ... else if else

- Entkopplung des Auslösers einer Anfrage mit seinem Empfänger
- Verkettung von Objekten mit unterschiedlichen Zuständigkeiten
→ eingehende Anfrage wird bis zu bearbeitendem Objekt weitergeleitet

Akteure

- Bearbeiter: definiert Interface für Anfragen
- Konkreter Bearbeiter: bearbeitet zuständige Anfragen, ansonsten Weiterleitung
- Client: initiiert Anfrage bei bestimmten Bearbeiter

Anwendung

- Antwort auf eine Anfrage durch ein vorab unbekanntes, zur Laufzeit erst bestimmtes Objekt erfolgen können soll
- eine Anfrage an mehrere Objekte gerichtet werden soll, ohne den richtigen Empfänger zu kennen
- beantwortende Objekte erst zur Laufzeit festgelegt werden

Vor- und Nachteile

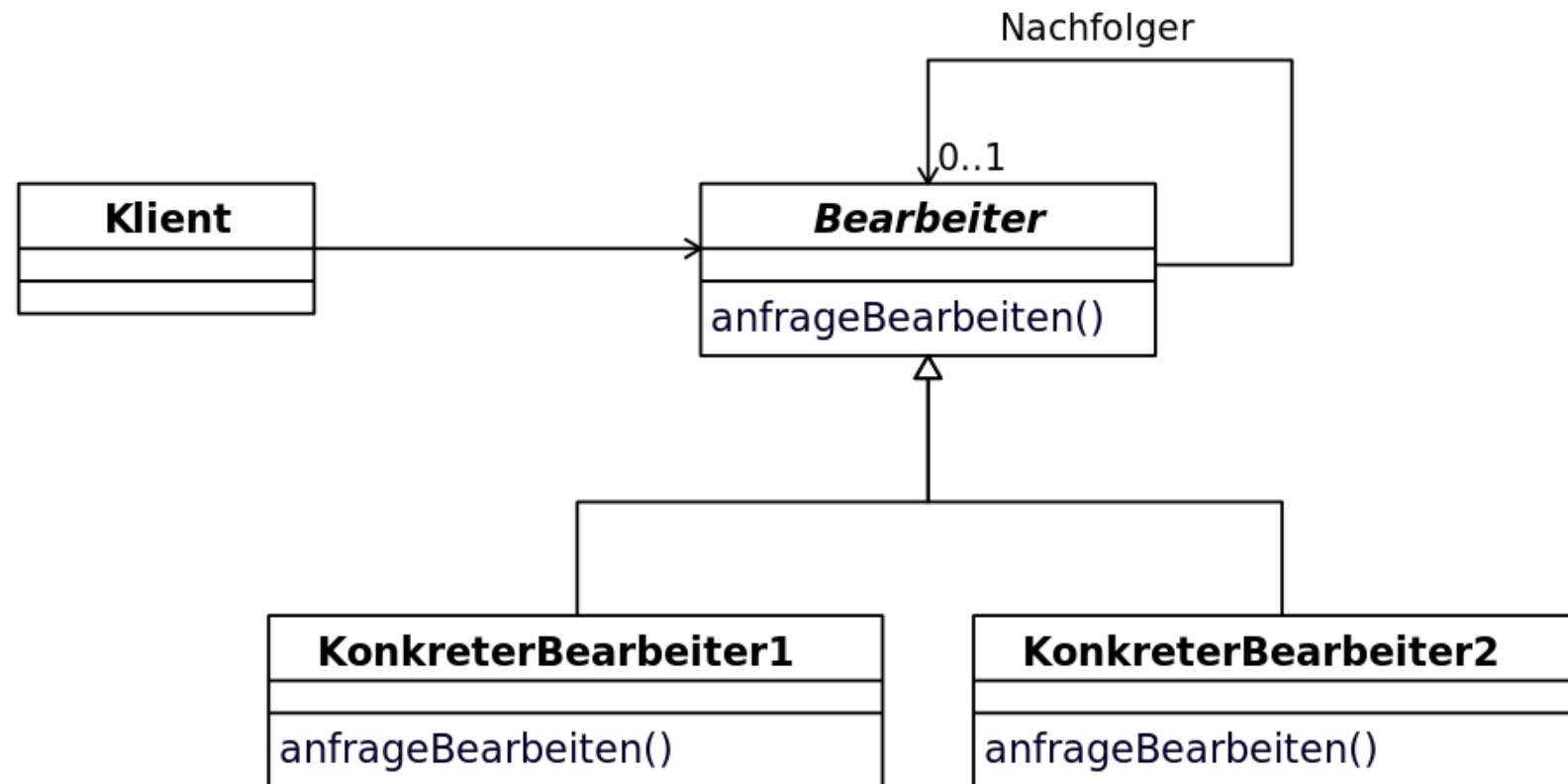
Vorteile

- Reduzierte Kopplung
 - Klient muss nicht wissen welcher zuständiger Bearbeiter antwortet
 - Kettenglieder kennen nur direkten Nachfolger
- Änderung Zuständigkeiten ohne in Kenntnis setzen potenzieller Klienten
- Zusätzliche Flexibilität
 - Zuständigkeiten können dynamisch eingefügt werden
 - Kette kann reorganisiert werden

Nachteile

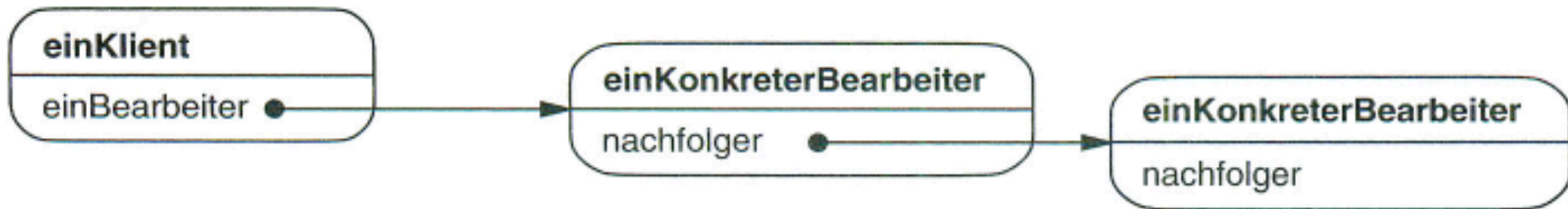
- Keine Garantie für Bearbeitung der Anfrage
- Endlosschleifen müssen aktiv vermieden werden

UML-Diagramm



UML-Diagramm

Objektstruktur



Code-Beispiel

Quellen

- <https://de.wikipedia.org/wiki/Zust%C3%A4ndigkeitskette>
- <https://refactoring.guru/design-patterns/chain-of-responsibility>
- [Uni Leipzig – Seminar Software Design Pattern](#)
- <https://examples.javacodegeeks.com/core-java/java-chain-of-responsibility-design-pattern-example/>

Vielen Dank für Eure Aufmerksamkeit!