Sommersemester 2021

Wolfgang Berger

Software Paradigmen

Behavioral Patterns

Verhaltensmuster

Verhaltensmuster

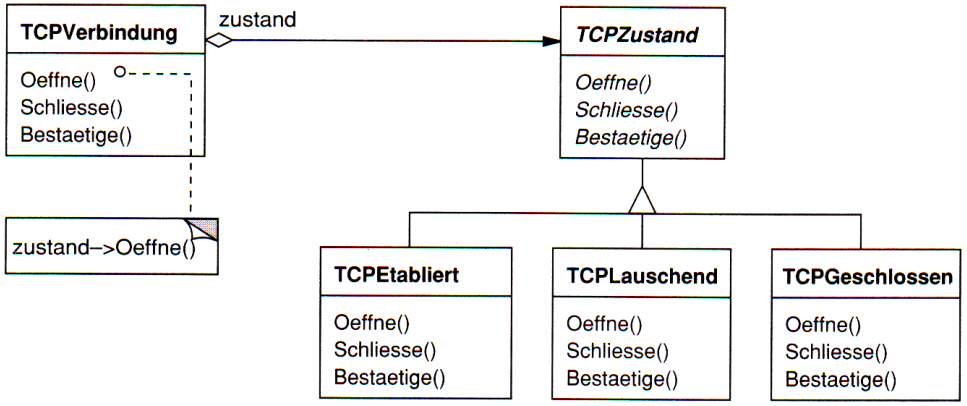
- Klassenbasierte
  - Interpreter
  - Template Method
- Objektbasierte
  - Observer
  - State
  - Command
  - Visitor
  - Iterator
  - Strategy
  - Mediator
  - Chain of Responsibility

State

- Zweck
  - Ermögliche es einem Objekt, sein Verhalten zu ändern, wenn sein interner Zustand sich ändert. Es wird so aussehen, als ob das Objekt seine Klasse gewechselt hat

State

- Motivation
  - Klasse TCPVerbindung
  - Zustände: Etabliert, Bereit, Beendet.
  - Das State Pattern beschreibt, wie die TCP Verbindung für jeden Zustand unterschiedliches Verhalten ermöglicht
  - Lösung:

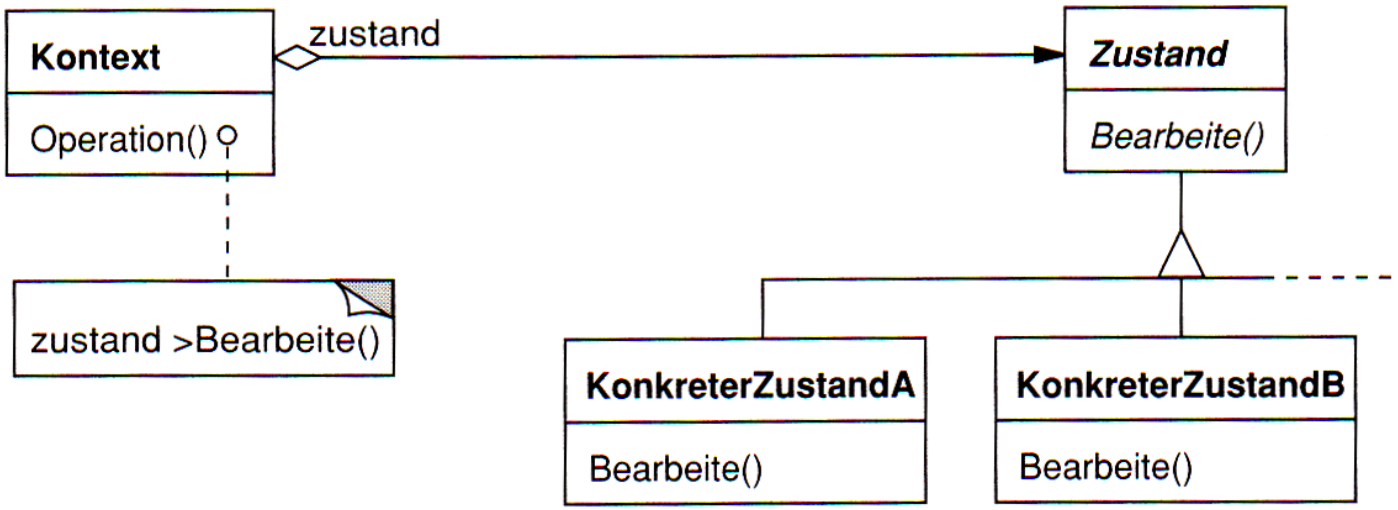


State

- Anwendbarkeit
  - Verwenden Sie das State Pattern in einem der folgenden Fälle:
    - Das Verhalten eines Objekts hängt von seinem Zustand ab, und es muss sein Verhalten zur Laufzeit und in Abhängigkeit von diesem Zustand ändern.
    - Die Operationen der Klasse besitzen große mehrteilige Bedingungsanweisungen, die vom Objektzustand abhängen.

State

- Struktur



State

- Teilnehmer
  - Kontext
    - Definiert die Klienten interessierende Schnittstelle
    - Verwaltet ein Exemplar einer KonkreterZustand-Unterklasse, welche den aktuellen Zustand definiert
  - Zustand
    - Definiert eine Schnittstelle zur Kapselung des mit einem bestimmten Zustand des Kontextobjekts verbundenen Verhaltens.
  - KonkreterZustand-Unterklasse
    - Jede Unterklasse implementiert ein Verhalten, das mit einem Zustand des Kontextobjekts verbunden ist.

State

- Interaktionen
  - Das Kontextobjekt delegiert zustandsspezifische Anfragen an das aktuelle KonkreterZustandObjekt
  - Ein Kontext kann sich selbst als ein Argument an das die Anfrage bearbeitende Zustandsobjekt mitgeben.
  - Das Kontextobjekt bietet die für Klienten hauptsächlich interessanten Schnittstelle
  - Sowohl die Kontext- als auch die KonkreterZustand-Unterklassen können entscheiden, welche Zustände aufeinander folgen und unter welchen Bedingungen sie dies tun.

State

- Konsequenzen
  - Zustandsspezifischer Code in Unterklassen verlagert – es ist leicht neue Zustände und Übergänge zu definieren.
  - Vermeidet das Warten vieler Codestücke, da keine Bedingungsabfragen der unterschiedlichen Zustände mehr nötig ist.
  - Zustandsmuster fördert eine bessere Strukturierung des Codes

State

- Konsequenzen
  - Explizite Zustandsübergänge – es wird nicht nur eine Eigenschaft geändert sondern der Typ des Objekts ändert sich.
  - Wenn Zustandsobjekte keine Objektvariablen besitzen können sie sogar gemeinsam genutzt werden. Es darf keinen intrinsischen Zustand geben sondern nur Verhalten.

State

- Implementierung
  - Definition der Zustandsübergänge
    - Kontext: Nachteil: bei Hinzufügen neuer Zustände muss der Kontext angepasst werden.
    - Besser: Zustandsklassen sind selbst für die Definition der Nachfolgezustände zuständig.
  - Erzeugen und Löschen von Zustandsobjekten
    - Bei Bedarf erzeugen und löschen oder
    - Im Voraus erzeugen und nie mehr löschen
  - Verwenden dynamischer Vererbung
    - Delegation (Bridge!)

State

- Übung 3!