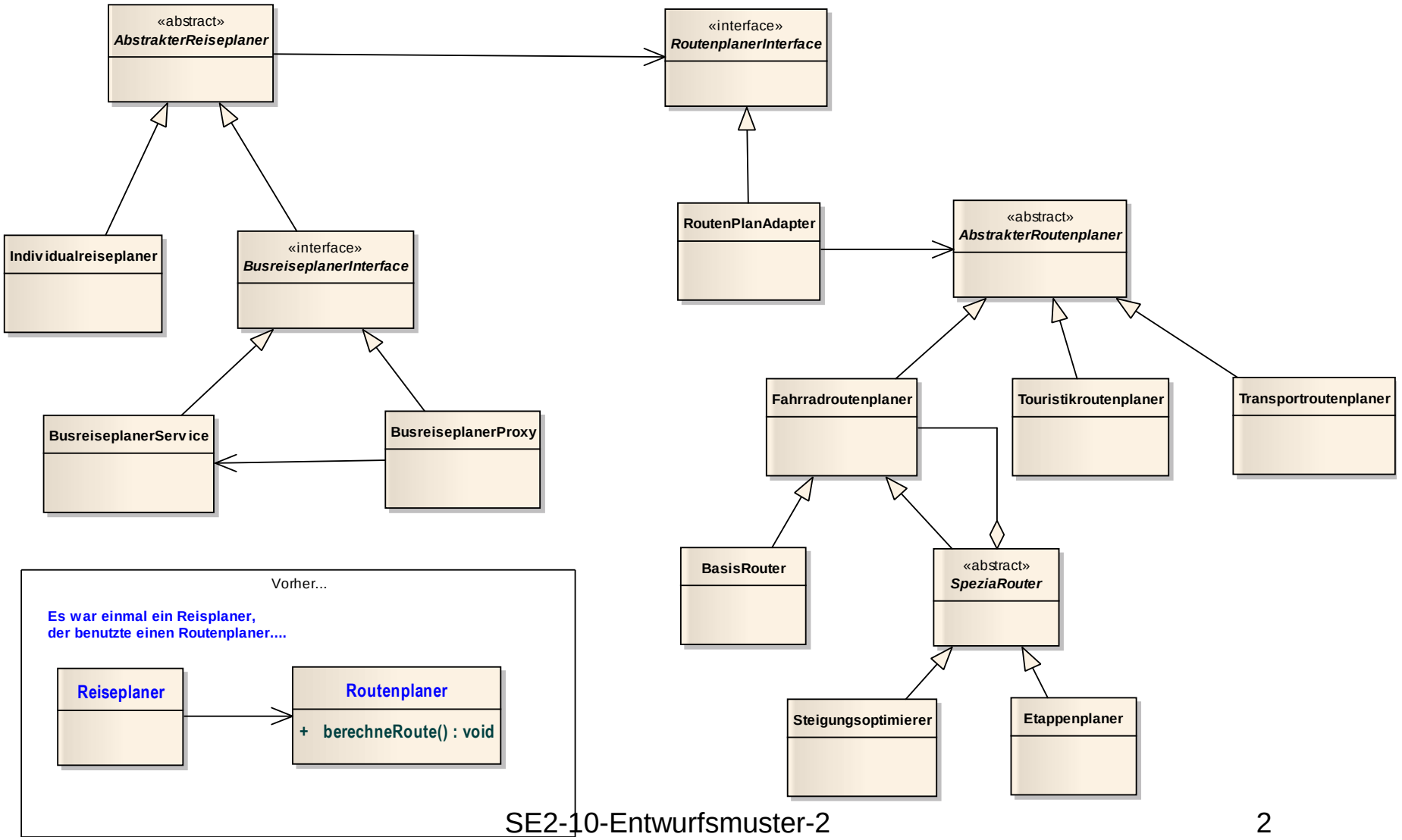
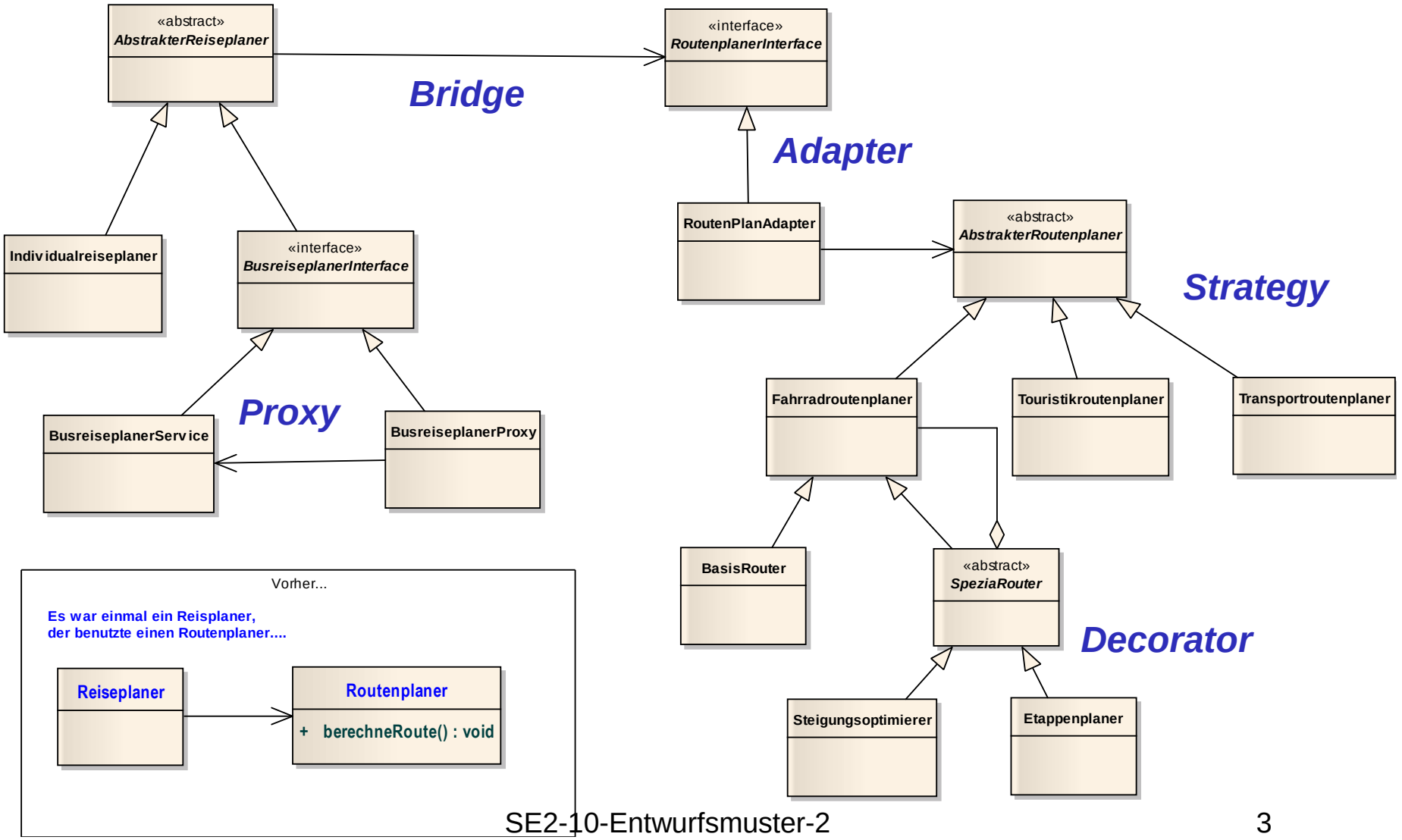
Methodik des Entwurfs - 2

- Erzeugungsmuster
- Verhaltensmuster

Strukturmuster-Puzzle
(zum Tafelbild)



Strukturmuster-Puzzle
(zum Tafelbild)



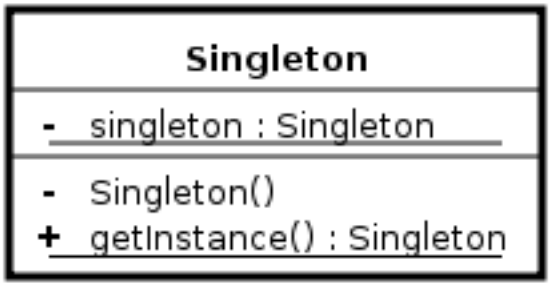
Entwurfsmuster

- Strukturmuster
  - Adapter, Strategie, Decorator, Proxy, Bridge, Facade, …
- Erzeugungsmuster
  - Singleton, (Abstract)Factory, ObjectPool, …
- Verhaltensmuster
  - Observer, Command, Iterator, TemplateMethod, State, Mediator, …
- Nebenläufigkeitsmuster
  - Active Object, Messaging, Reactor, ThreadPool, …
- Architekturmuster

Erzeugungsmuster
Singleton

Zweck:

- Erzeugung von maximal einem Objekt der Klasse, das von allen Benutzern referiert wird.

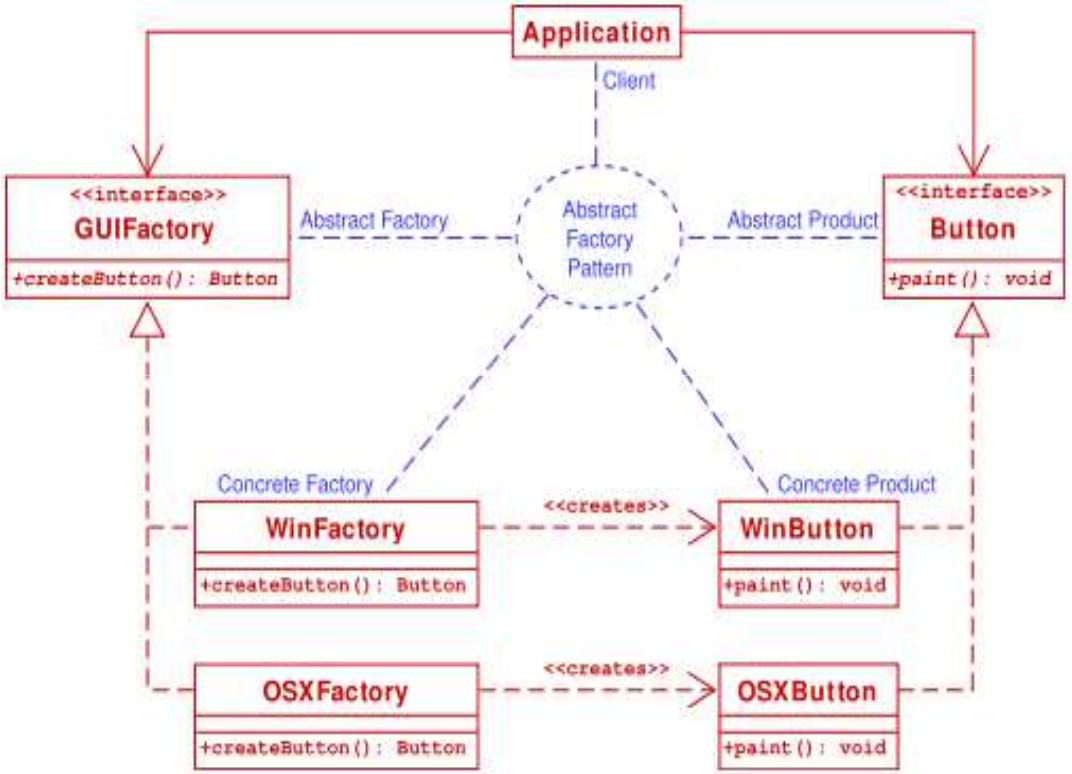


Erzeugungsmuster

Abstract Factory

Zweck:

- Gemeinsames Interface zur Erzeugung von Objekten in *verschiedenen Kontexten*
- die konkrete Erzeugung wird in die Implementierungen ausgelagert.
- Der Client muss den konkreten Kontext nicht kennen.



Quelle: Wikipedia

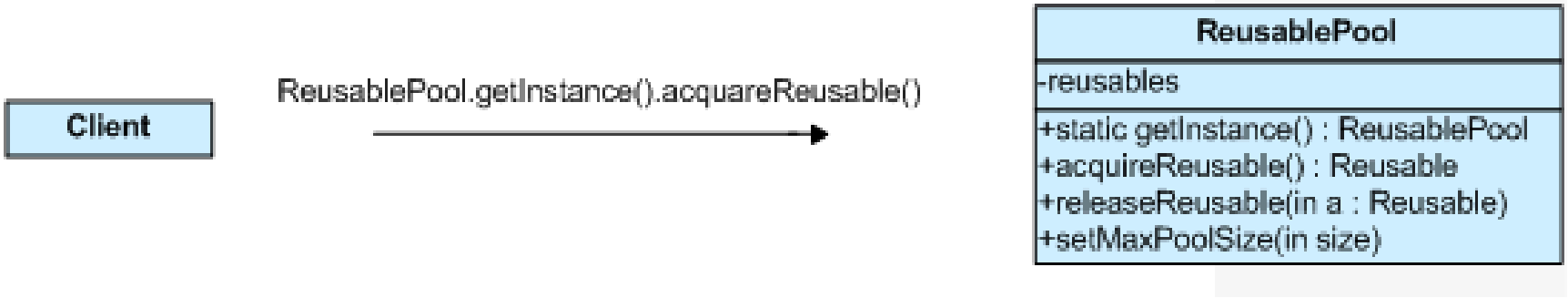
- Das Pattern *Factory Method* beschreibt den Zusammenhang der abstrakten und konkreten create-Methode.

Erzeugungsmuster

Object Pool

Zweck:

- Erzeugung und Verwaltung wiederverwendbarer Objekte
- Aufwand zum Erzeugen und Löschen von Objekten wird gespart
- Sinnvoll bei häufig benutzten Objekten mit kurzer Lebensdauer, insbesondere Netzwerk-Verbindungsobjekte

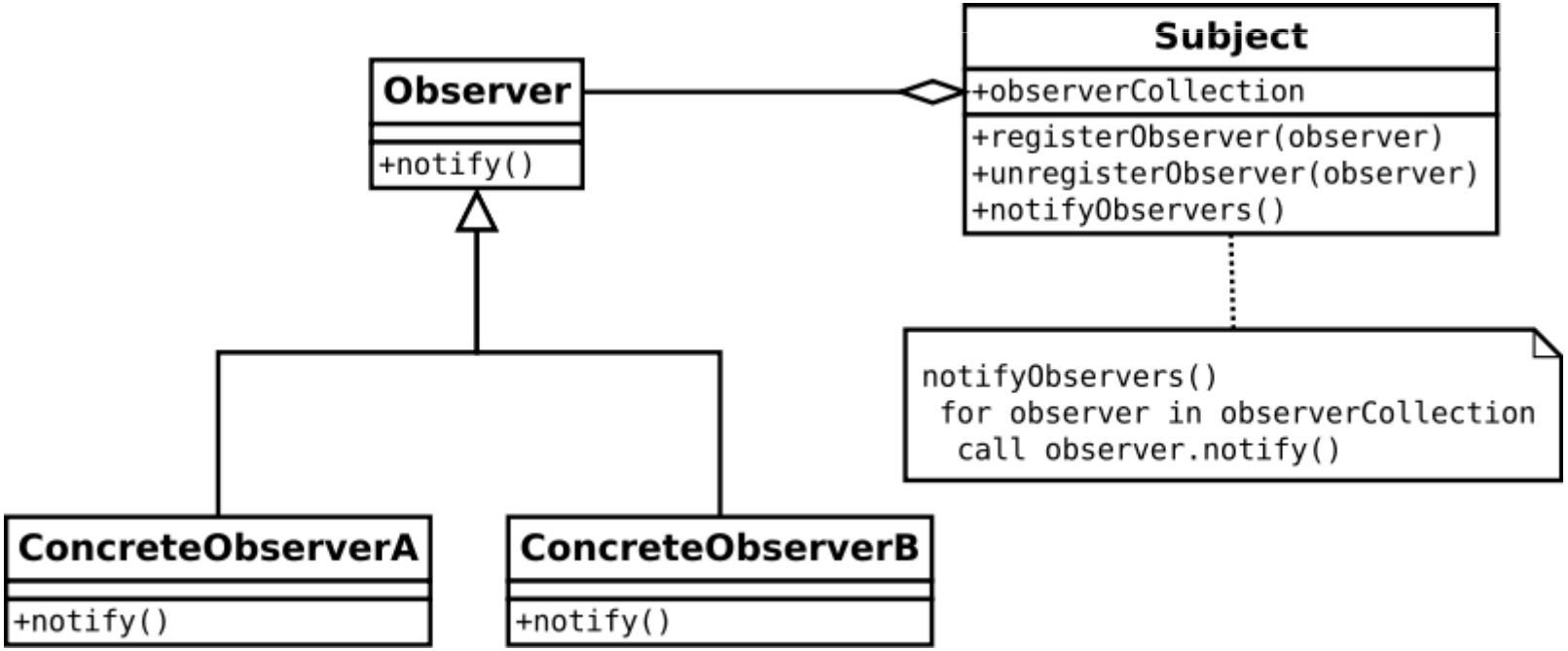


Verhaltensmuster

Observer

Zweck:

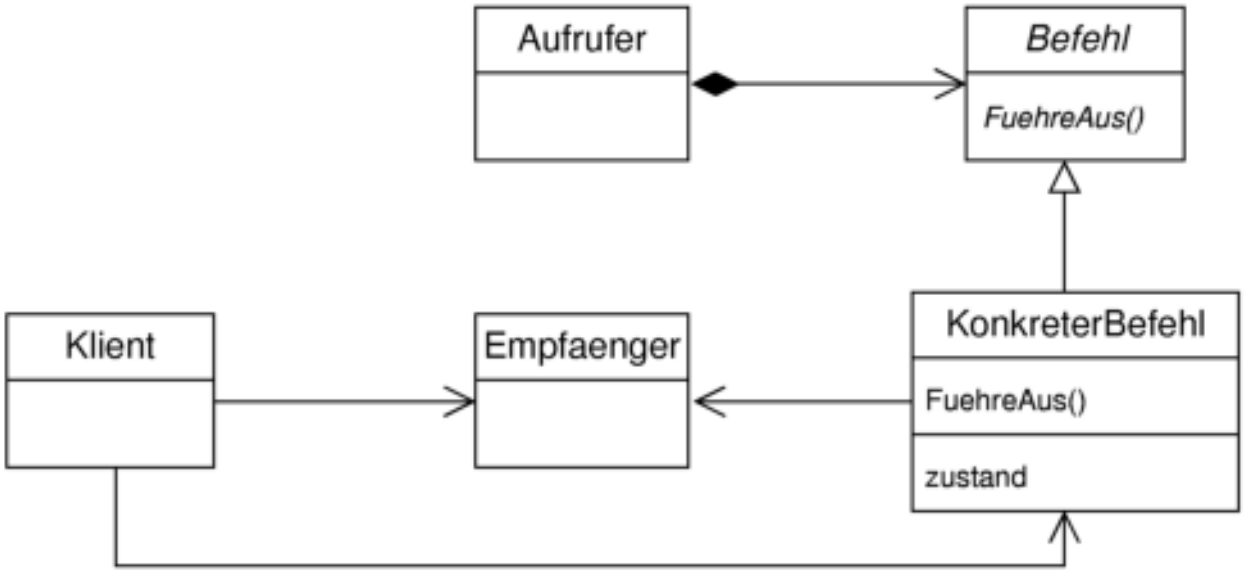
- konfigurierbare Synchronisation von Objekten mit einem veränderlichen Objekt ("Subjekt")
- Publish-Subscribe-Prinzip:
  - Observer registrieren sich beim Subjekt der Beobachtung
  - Das Subjekt benachrichtigt alle registrierten Objekte bei Veränderung
  - Die registrierten Objekte lesen die Veränderung aktiv ab.



Verhaltensmuster
Command

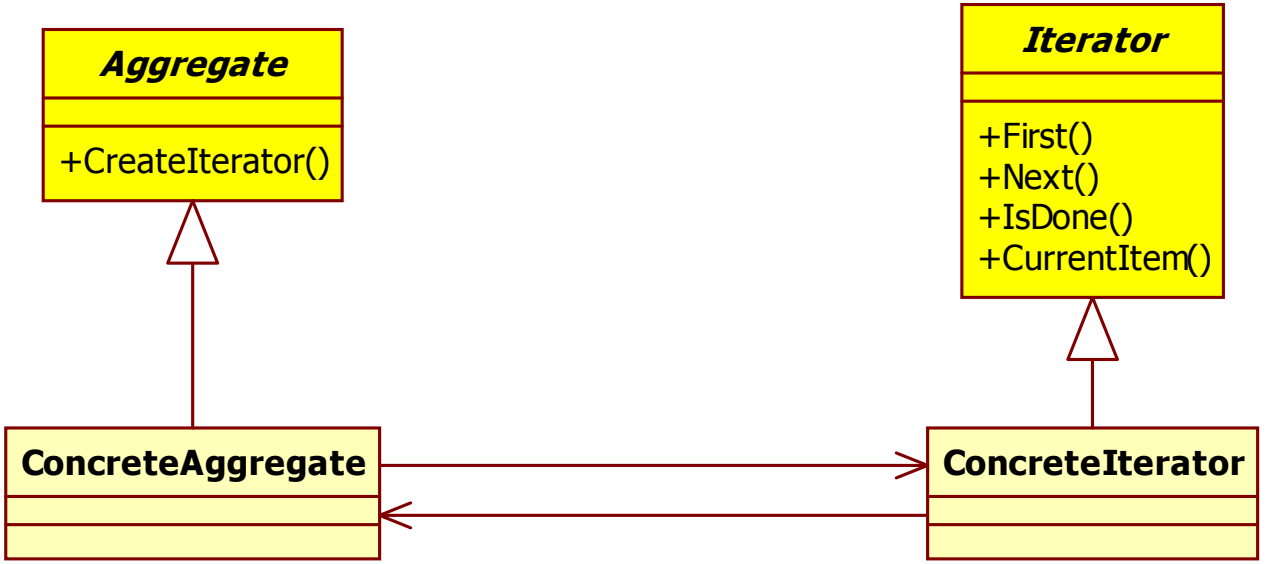
Zweck:

- Kapselung eines Befehls
- Trennung von Befehlserzeugung und Befehlsaufruf
- erleichtert Einreihen in Warteschlangen, Logging, Undo



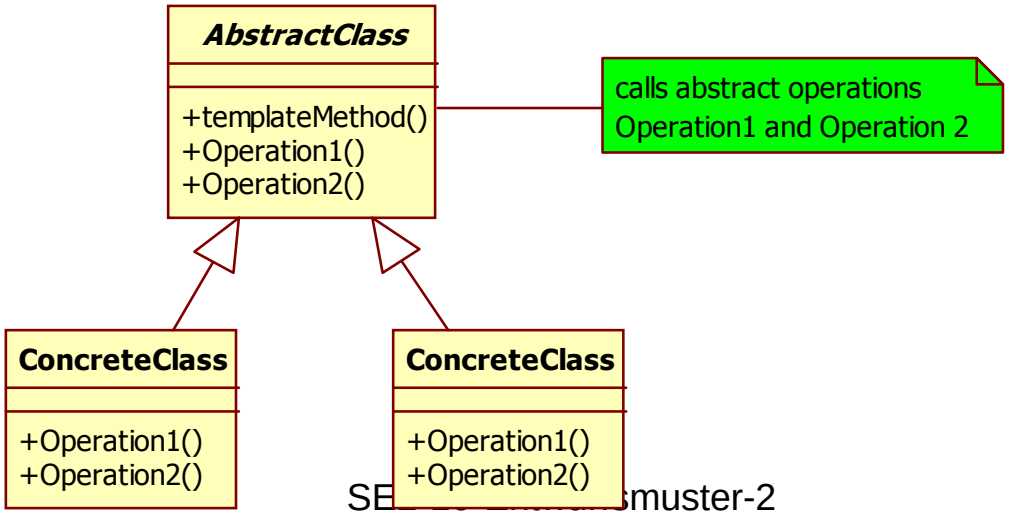
Verhaltensmuster
Iterator

- Zweck:
  - Ursprünglich in Kombination mit Compositum
  - Iteration über die Struktur ohne Kenntnis der Implementierung mit der Möglichkeit, die in der Schnittstelle angegebene (für Knoten und Blätter implementierte) Methode an beliebiger Stelle auszuführen.
  - Verallgemeinert für traversierbare rekursive Strukturen.



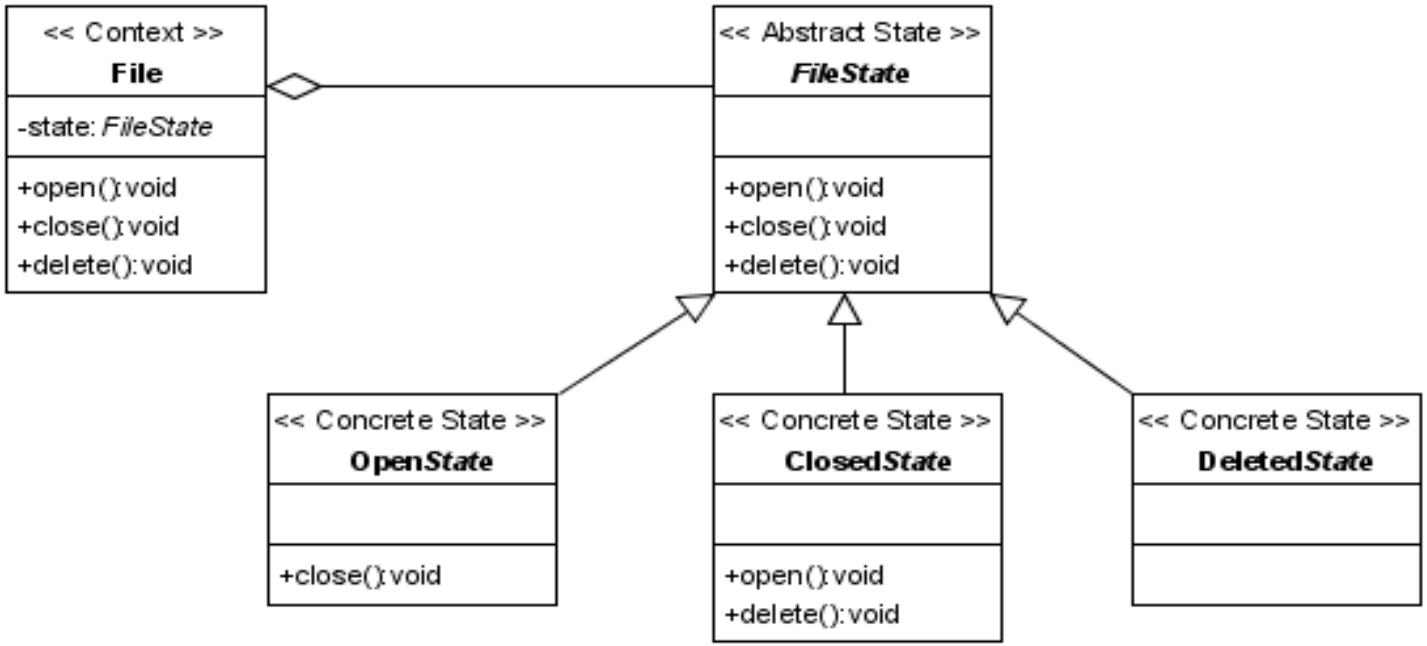
Verhaltensmuster
Template Method

- Zweck:
  - Verwendung der Struktur eines Algorithmus mit verschiedenen Einzeloperationen
  - "Generischer" Algorithmus
- Kontext:
  - Ein Algorithmus besteht aus einer globalen Ablaufstruktur (z.B. Iterieren über eine Liste)
  - und innerhalb dieser Struktur Detailoperationen (z.B. Verdoppeln des Elementwerts)
  - Die Ablaufstruktur kann *unabhängig von den Detailoperationen implementiert* werden
  - Die Detailoperationen werden dabei *abstrakt benutzt*, quasi als "Platzhalter" für die konkreten Implementierungen



Verhaltensmuster
State

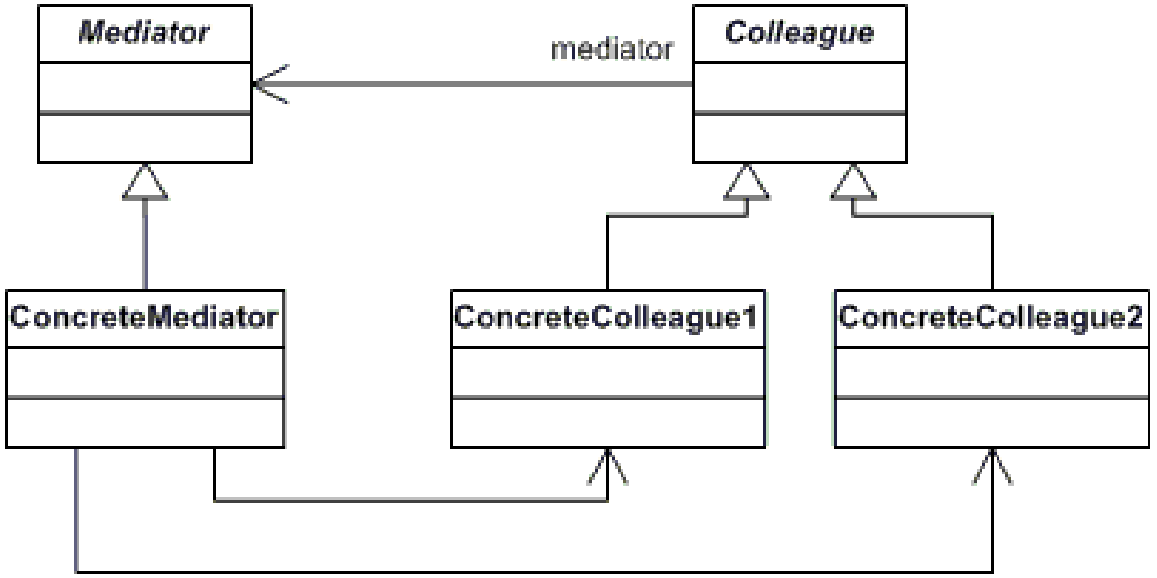
- Zweck:
  - Verhaltensänderung abhängig vom Objektzustand
  - Das zustandsabhängige Verhalten wird in extra Objekte gekapselt
  - Bei Zustandsänderung tauscht der Kontext das Zustandsobjekt aus
- *Das Objekt scheint zu wechseln (gutes Beispiel: Mauszeiger)*.



Verhaltensmuster
Mediator

Zweck:

- Lose Kopplung durch Auslagerung der Kommunikation zwischen Klassen
- Reduziert die Abhängigkeiten bei veränderlichen Schnittstellen
- Oft nachträglich bei Systemwartung und Erweiterung eingeführt.



Das waren Entwurfsmuster im Überblick - im Einzelfall *nachlesen!*

Zum Abschluss der Modellarbeit geht es noch um Skalierbarkeit.