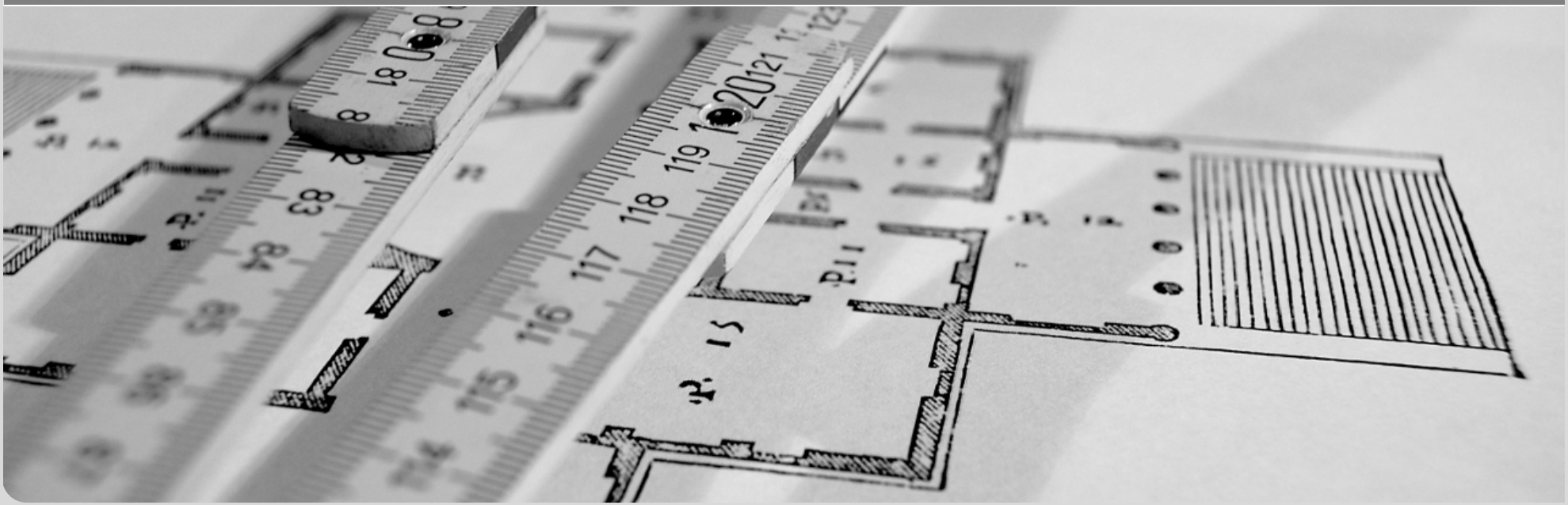
Karlsruher Institut für Technologie

Softwaretechnik 1 - 4. Tutorium

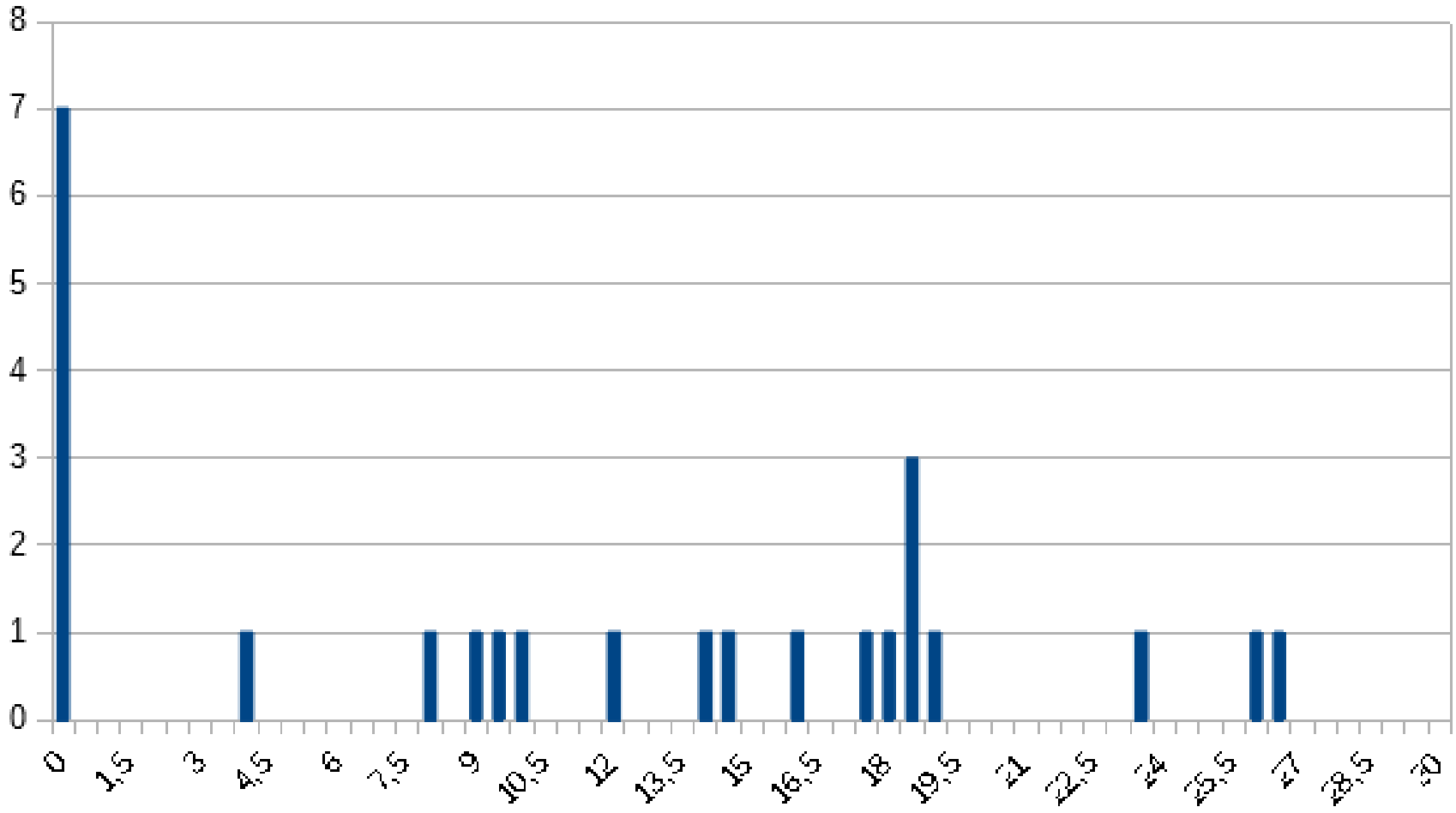
Tutorium 18

Felix Bachmann | 19.06.2018

KIT - INSTITUT FÜR PROGRAMMSTRUKTUREN UND DATENORGANISATION (IPD)



3. Übungsblatt Statistik



∅ 11,3 bzw. 15,7 von 27+3

Häufige Fehler

Aufgabe 1 (Plug-In-Architektur: PluginManager + JmjrstPlugin)

Häufige Fehler

Aufgabe 1 (Plug-In-Architektur: PluginManager + JmjrstPlugin)

- Plugins anhand **Klassen**namen vergleichen, nicht getName()

Häufige Fehler

Aufgabe 1 (Plug-In-Architektur: PluginManager + JmjrstPlugin)

- Plugins anhand **Klassen**namen vergleichen, nicht getName()
- Strings auslagern (Konstanten oder Datei)

Häufige Fehler

Aufgabe 1 (Plug-In-Architektur: PluginManager + JmjrstPlugin)

- Plugins anhand **Klassen**namen vergleichen, nicht getName()
- Strings auslagern (Konstanten oder Datei)
- PluginManager gibt euch Iterable $\Longrightarrow$ nutzt Iterator
  - kein Casten, Kopieren in Liste

Häufige Fehler

Aufgabe 1 (Plug-In-Architektur: PluginManager + JmjrstPlugin)

- Plugins anhand **Klassen**namen vergleichen, nicht getName()
- Strings auslagern (Konstanten oder Datei)
- PluginManager gibt euch Iterable $\Longrightarrow$ nutzt Iterator
  - kein Casten, Kopieren in Liste
- orientiert euch nicht am JMJRST-Stil

Häufige Fehler

Aufgabe 2 (Plug-In)

Häufige Fehler

Aufgabe 2 (Plug-In)

- Prüfen auf *.png/*.jpg sollte case insensitive sein

Häufige Fehler

Aufgabe 2 (Plug-In)

- Prüfen auf *.png/*.jpg sollte case insensitive sein
- Anm.: MetainfServices tut manchmal nicht richtig (oft hilft `mvn clean package`)

Häufige Fehler

Aufgabe 2 (Plug-In)

- Prüfen auf *.png/*.jpg sollte case insensitive sein
- Anm.: MetainfServices tut manchmal nicht richtig (oft hilft `mvn clean package`)

Aufgabe 3 (iMage-Bundle)

- keine :D

Häufige Fehler

Aufgabe 4 (Aktivitätsdiagramm)

Häufige Fehler

Aufgabe 4 (Aktivitätsdiagramm)

- keine Partition verwendet

Häufige Fehler

Aufgabe 4 (Aktivitätsdiagramm)

- keine Partition verwendet
- Aktivitiäten = runde Ecken, Objekte = spitze Ecken

Häufige Fehler

Aufgabe 4 (Aktivitätsdiagramm)

- keine Partition verwendet
- Aktivitiäten = runde Ecken, Objekte = spitze Ecken
- denkt an die Rauten!

Häufige Fehler

Aufgabe 4 (Aktivitätsdiagramm)

- keine Partition verwendet
- Aktivitiäten = runde Ecken, Objekte = spitze Ecken
- denkt an die Rauten!
- [Bedingung]

Häufige Fehler

Aufgabe 4 (Aktivitätsdiagramm)

- keine Partition verwendet
- Aktivitiäten = runde Ecken, Objekte = spitze Ecken
- denkt an die Rauten!
- [Bedingung]
- verschachtelte Aktivitäten $\Longrightarrow$ irgendwo passender Kasten dazu

Häufige Fehler

Aufgabe 5 (Zustandsdiagramm)

Häufige Fehler

Aufgabe 5 (Zustandsdiagramm)

- Übergänge,Zustände vergessen

Häufige Fehler

Aufgabe 5 (Zustandsdiagramm)

- Übergänge,Zustände vergessen
- Notation parallel: DxG

Häufige Fehler

Aufgabe 5 (Zustandsdiagramm)

- Übergänge,Zustände vergessen
- Notation parallel: DxG
- komplettes Diagramm hinzeichnen für Äquivalenz

Häufige Fehler

Aufgabe 5 (Zustandsdiagramm)

- Übergänge,Zustände vergessen
- Notation parallel: DxG
- komplettes Diagramm hinzeichnen für Äquivalenz

Häufige Fehler

Aufgabe 6 (Sequenzdiagramm)

Häufige Fehler

Aufgabe 6 (Sequenzdiagramm)

- bzgl. Konstruktor sind VL-Folien etwas blöd

Häufige Fehler

Aufgabe 6 (Sequenzdiagramm)

- bzgl. Konstruktor sind VL-Folien etwas blöd
- asynchron vs. synchron (Pfeilspitzen sind wichtig!)

Häufige Fehler

Aufgabe 6 (Sequenzdiagramm)

- bzgl. Konstruktor sind VL-Folien etwas blöd
- asynchron vs. synchron (Pfeilspitzen sind wichtig!)
- nicht statische Instanzen unterstreichen

Häufige Fehler

Aufgabe 6 (Sequenzdiagramm)

- bzgl. Konstruktor sind VL-Folien etwas blöd
- asynchron vs. synchron (Pfeilspitzen sind wichtig!)
- nicht statische Instanzen unterstreichen
- Instanz-Kästen erst dann hinzeichnen, wenn Instanz auch existiert

Häufige Fehler

Aufgabe 6 (Sequenzdiagramm)

- bzgl. Konstruktor sind VL-Folien etwas blöd
- asynchron vs. synchron (Pfeilspitzen sind wichtig!)
- nicht statische Instanzen unterstreichen
- Instanz-Kästen erst dann hinzeichnen, wenn Instanz auch existiert
- Selbstaufruf-Syntax

Aufgabe 7 (Testen mit Nachahmungen)

Häufige Fehler

Aufgabe 6 (Sequenzdiagramm)

- bzgl. Konstruktor sind VL-Folien etwas blöd
- asynchron vs. synchron (Pfeilspitzen sind wichtig!)
- nicht statische Instanzen unterstreichen
- Instanz-Kästen erst dann hinzeichnen, wenn Instanz auch existiert
- Selbstaufruf-Syntax

Aufgabe 7 (Testen mit Nachahmungen)

- Substitutionsprinzip: fordert dass Objekte einer Unterklasse immer auch im Kontext der Oberklasse eingesetzt werden können

Häufige Fehler

Aufgabe 6 (Sequenzdiagramm)

- bzgl. Konstruktor sind VL-Folien etwas blöd
- asynchron vs. synchron (Pfeilspitzen sind wichtig!)
- nicht statische Instanzen unterstreichen
- Instanz-Kästen erst dann hinzeichnen, wenn Instanz auch existiert
- Selbstaufruf-Syntax

Aufgabe 7 (Testen mit Nachahmungen)

- Substitutionsprinzip: fordert dass Objekte einer Unterklasse immer auch im Kontext der Oberklasse eingesetzt werden können
- Varianz war kein Problem, da Signatur gleich

Häufige Fehler

Aufgabe 6 (Sequenzdiagramm)

- bzgl. Konstruktor sind VL-Folien etwas blöd
- asynchron vs. synchron (Pfeilspitzen sind wichtig!)
- nicht statische Instanzen unterstreichen
- Instanz-Kästen erst dann hinzeichnen, wenn Instanz auch existiert
- Selbstaufruf-Syntax

Aufgabe 7 (Testen mit Nachahmungen)

- Substitutionsprinzip: fordert dass Objekte einer Unterklasse immer auch im Kontext der Oberklasse eingesetzt werden können
- Varianz war kein Problem, da Signatur gleich
- Problem war Verhalten, schwächere Nachbedingung

Häufige Fehler

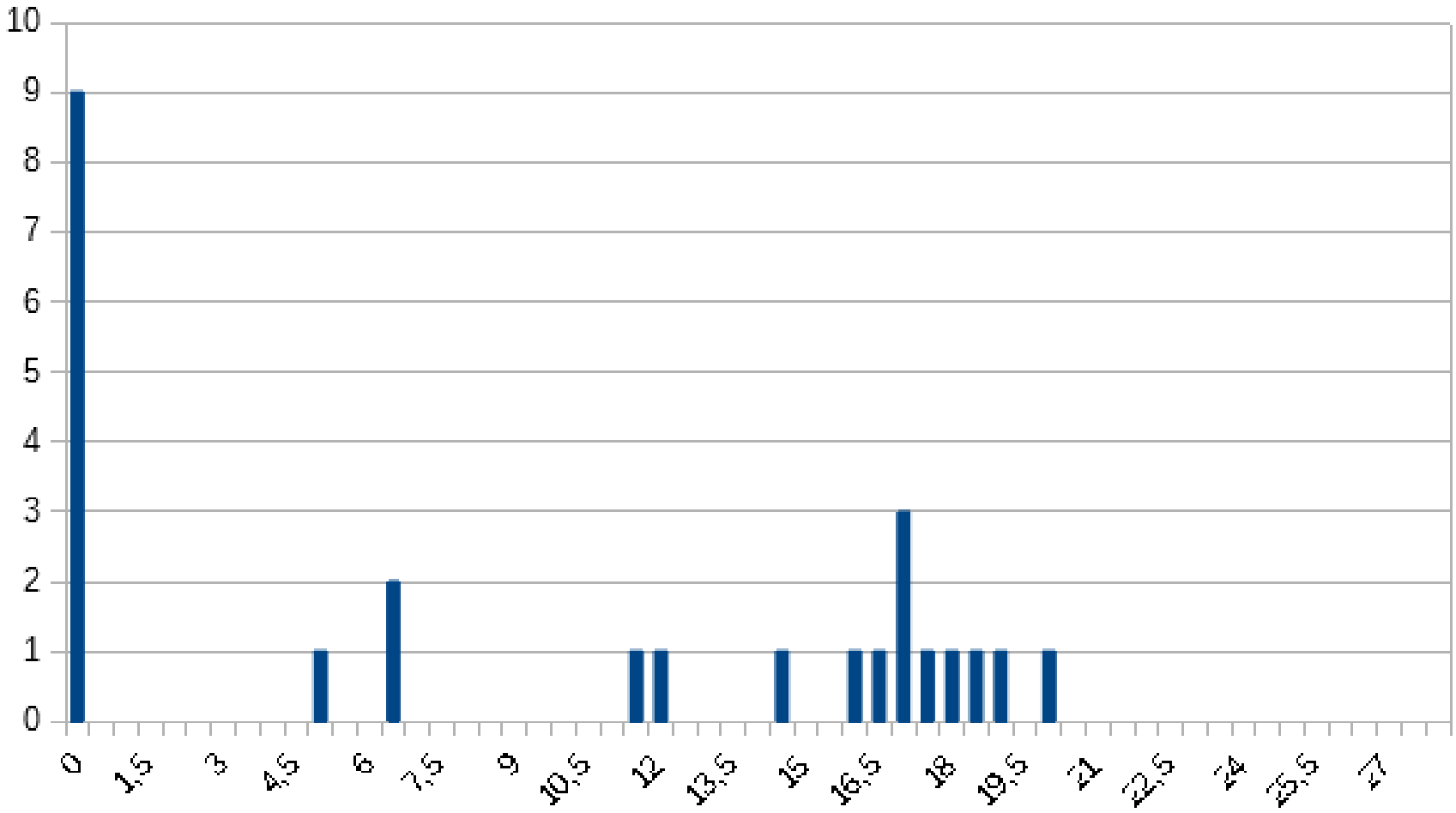
Aufgabe 6 (Sequenzdiagramm)

- bzgl. Konstruktor sind VL-Folien etwas blöd
- asynchron vs. synchron (Pfeilspitzen sind wichtig!)
- nicht statische Instanzen unterstreichen
- Instanz-Kästen erst dann hinzeichnen, wenn Instanz auch existiert
- Selbstaufruf-Syntax

Aufgabe 7 (Testen mit Nachahmungen)

- Substitutionsprinzip: fordert dass Objekte einer Unterklasse immer auch im Kontext der Oberklasse eingesetzt werden können
- Varianz war kein Problem, da Signatur gleich
- Problem war Verhalten, schwächere Nachbedingung

4. Übungsblatt Statistik



∅ 9,3 (alle), 14,5 (abgegeben) von 25+3

Häufige Fehler

Aufgabe 1: GUI für iMage

```
// won't work from the jar
File f = new File("src/main/resources/bla.png");

// use one of the following (which one depends on your needs):
this.getClass().getResource("bla.png");
Thread.currentThread().getContextClassLoader().getResource("bla.png");
System.class.getResource("bla.png");
```

Häufige Fehler

Aufgabe 1: GUI für iMage

```
// won't work from the jar
File f = new File("src/main/resources/bla.png");

// use one of the following (which one depends on your needs):
this.getClass().getResource("bla.png");
Thread.currentThread().getContextClassLoader().getResource("bla.png");
System.class.getResource("bla.png");
```

- Gottklassen, wir wollen aber sinnvolle Objektorientierung!

Häufige Fehler

Aufgabe 1: GUI für iMage

```java
// won't work from the jar
File f = new File("src/main/resources/bla.png");

// use one of the following (which one depends on your needs):
this.getClass().getResource("bla.png");
Thread.currentThread().getContextClassLoader().getResource("bla.png");
System.class.getResource("bla.png");
```

- Gottklassen, wir wollen aber sinnvolle Objektorientierung!
- `SwingUtilities.invokeLater(e -> startGui())` benutzen:
  $\Longrightarrow$ Thread-Safe (siehe nächstes Tut)

Häufige Fehler

Aufgabe 1: GUI für iMage

```java
// won't work from the jar
File f = new File("src/main/resources/bla.png");

// use one of the following (which one depends on your needs):
this.getClass().getResource("bla.png");
Thread.currentThread().getContextClassLoader().getResource("bla.png");
System.class.getResource("bla.png");
```

- Gottklassen, wir wollen aber sinnvolle Objektorientierung!
- `SwingUtilities.invokeLater(e -> startGui())` benutzen:
  $\Longrightarrow$ Thread-Safe (siehe nächstes Tut)

Häufige Fehler

Aufgabe 2: Zustandsdiagramm für Wasserzeichnen

- $a()[b] \neq [b]/a()$
  $\Longrightarrow$ beim skalieren/exception werfen

Häufige Fehler

Aufgabe 2: Zustandsdiagramm für Wasserzeichnen

- $a()[b] \neq [b]/a()$
  $\Longrightarrow$ beim skalieren/exception werfen
- sowohl „validiertes Bild“ als auch „skaliertes Bild“ ein Zustand

Häufige Fehler

Aufgabe 2: Zustandsdiagramm für Wasserzeichnen

- $a()[b] \neq [b]/a()$
  $\Longrightarrow$ beim skalieren/exception werfen
- sowohl „validiertes Bild“ als auch „skaliertes Bild“ ein Zustand

Aufgabe 3: git

- bei Umbenennung und Änderung direkt zu commiten würde beides hinzufügen

Häufige Fehler

Aufgabe 2: Zustandsdiagramm für Wasserzeichnen

- $a()[b] \neq [b]/a()$
  $\Longrightarrow$ beim skalieren/exception werfen
- sowohl „validiertes Bild“ als auch „skaliertes Bild“ ein Zustand

Aufgabe 3: git

- bei Umbenennung und Änderung direkt zu commiten würde beides hinzufügen
  $\Longrightarrow$ entweder `add -N`, `add -p`

Häufige Fehler

Aufgabe 2: Zustandsdiagramm für Wasserzeichnen

- $a()[b] \neq [b]/a()$
  $\Longrightarrow$ beim skalieren/exception werfen
- sowohl „validiertes Bild“ als auch „skaliertes Bild“ ein Zustand

Aufgabe 3: git

- bei Umbenennung und Änderung direkt zu commiten würde beides hinzufügen
  $\Longrightarrow$ entweder `add -N`, `add -p`
  $\Longrightarrow$ oder `mv neu alt`, `git mv alt neu`

Häufige Fehler

Aufgabe 2: Zustandsdiagramm für Wasserzeichnen

- $a()[b] \neq [b]/a()$
  $\Longrightarrow$ beim skalieren/exception werfen
- sowohl „validiertes Bild“ als auch „skaliertes Bild“ ein Zustand

Aufgabe 3: git

- bei Umbenennung und Änderung direkt zu commiten würde beides hinzufügen
  $\Longrightarrow$ entweder `add -N`, `add -p`
  $\Longrightarrow$ oder `mv neu alt`, `git mv alt neu`
- git rm -r löscht rekursiv Ordner (inkl. der Überordner!)
  - und fügt implizit zur Staging Area hinzu, kein add nötig

Häufige Fehler

Aufgabe 4: Architekturstile für JMJRST

- Zuordnung begründen, wenn unklar

Häufige Fehler

Aufgabe 4: Architekturstile für JMJRST

- Zuordnung begründen, wenn unklar
- Main eindeutig zugeordnet

Häufige Fehler

Aufgabe 4: Architekturstile für JMJRST

- Zuordnung begründen, wenn unklar
- Main eindeutig zugeordnet
- Änderungen zu vage beschrieben

Evaluation

leider nur 5 Teilnehmer??

Gut	Schlecht/Verbesserungswürdig
Folien (3) Beispiele und Code (3) Erklärungen (2) Tipps (2) Aufgaben (2)	zu viel Zeit für Aufgaben (2) Wahr/Falsch zu einfach (1) nicht immer Lösungen auf Folie (1) gleiche Beispiele wie in VL (1) Bewertungen zu kurz (1)

Was bisher geschah..

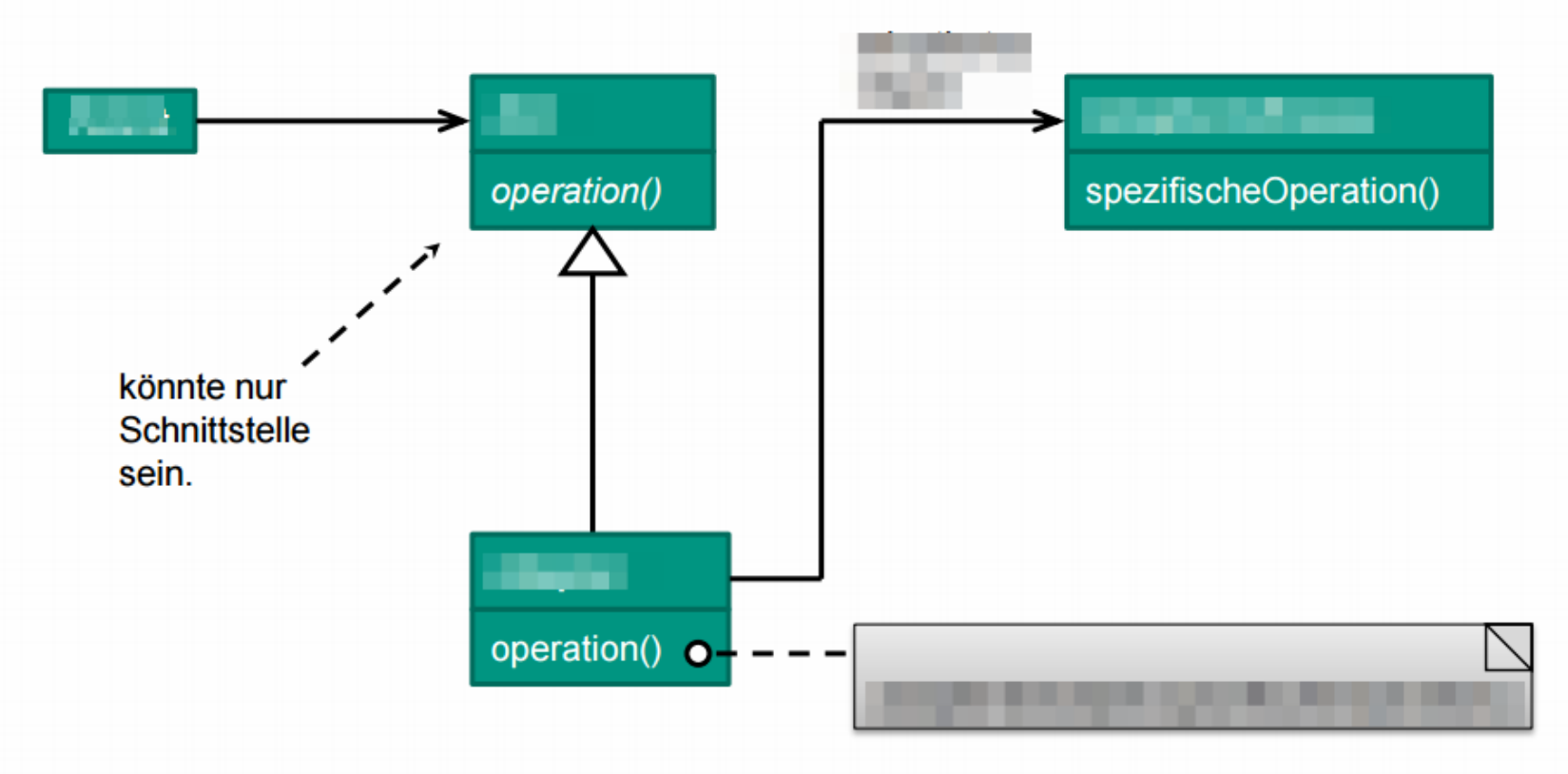
- haben uns Entkopplungmuster angeschaut

Was bisher geschah..

- haben uns Entkopplungmuster angeschaut
  $\Longrightarrow$ Beobachter, Iterator, Adapter, Stellvertreter

Was bisher geschah..

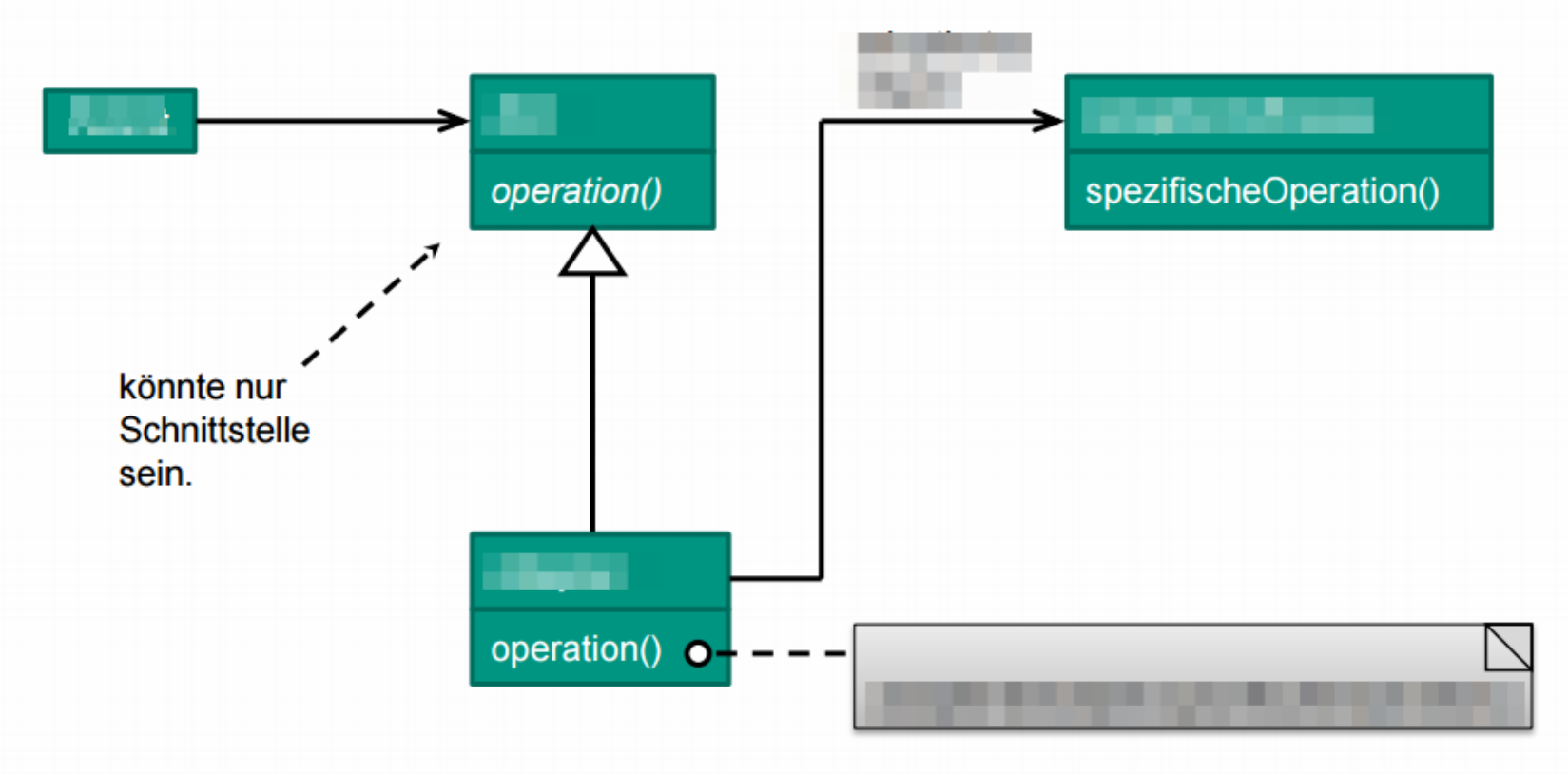
- haben uns Entkopplungmuster angeschaut
  $\Longrightarrow$ Beobachter, Iterator, Adapter, Stellvertreter



Welches Entwurfsmuster?

Was bisher geschah..

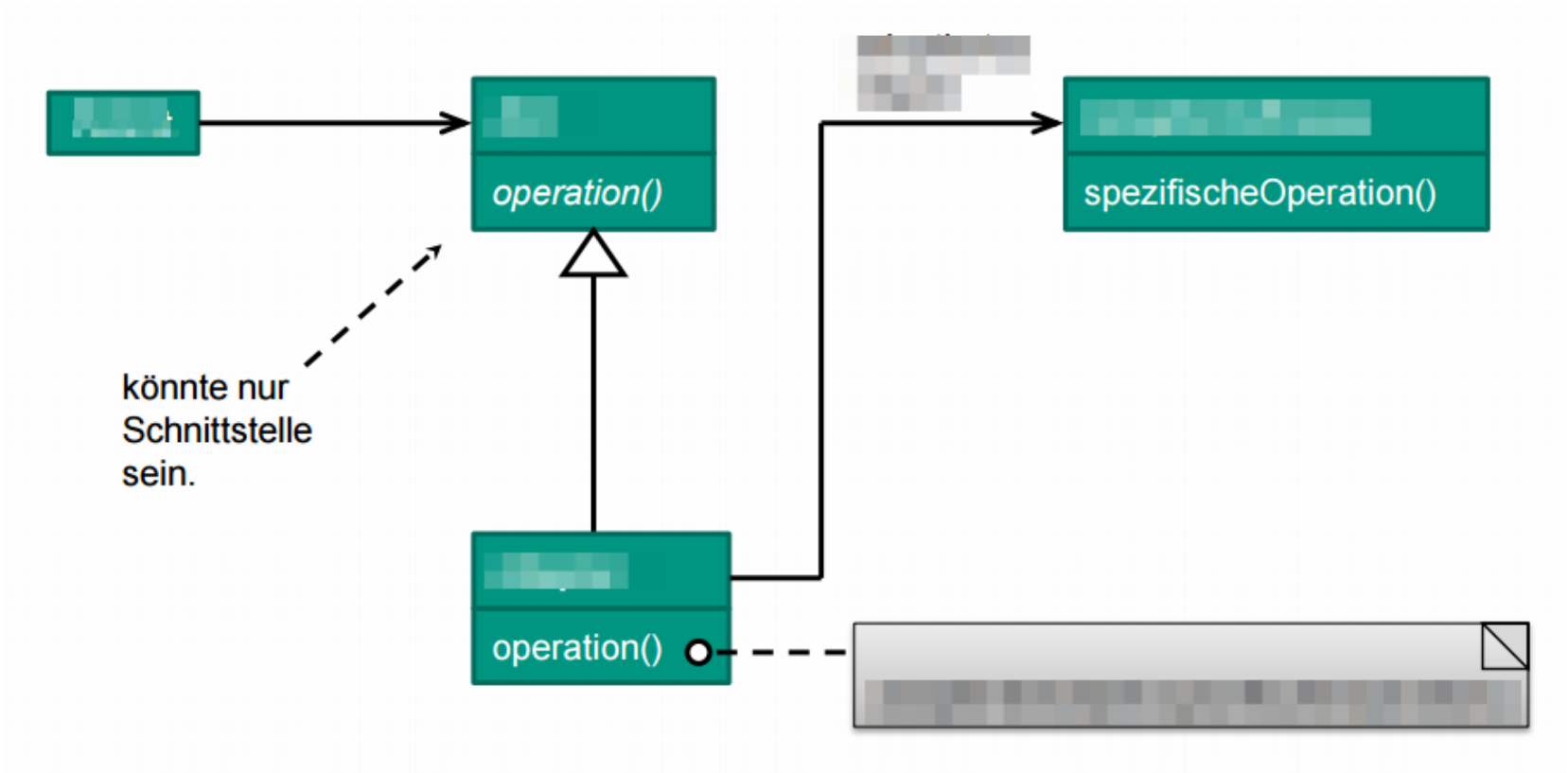
- haben uns Entkopplungmuster angeschaut
  $\Longrightarrow$ Beobachter, Iterator, Adapter, Stellvertreter



Welches Entwurfsmuster? (Objekt-)Adapter

Was bisher geschah..

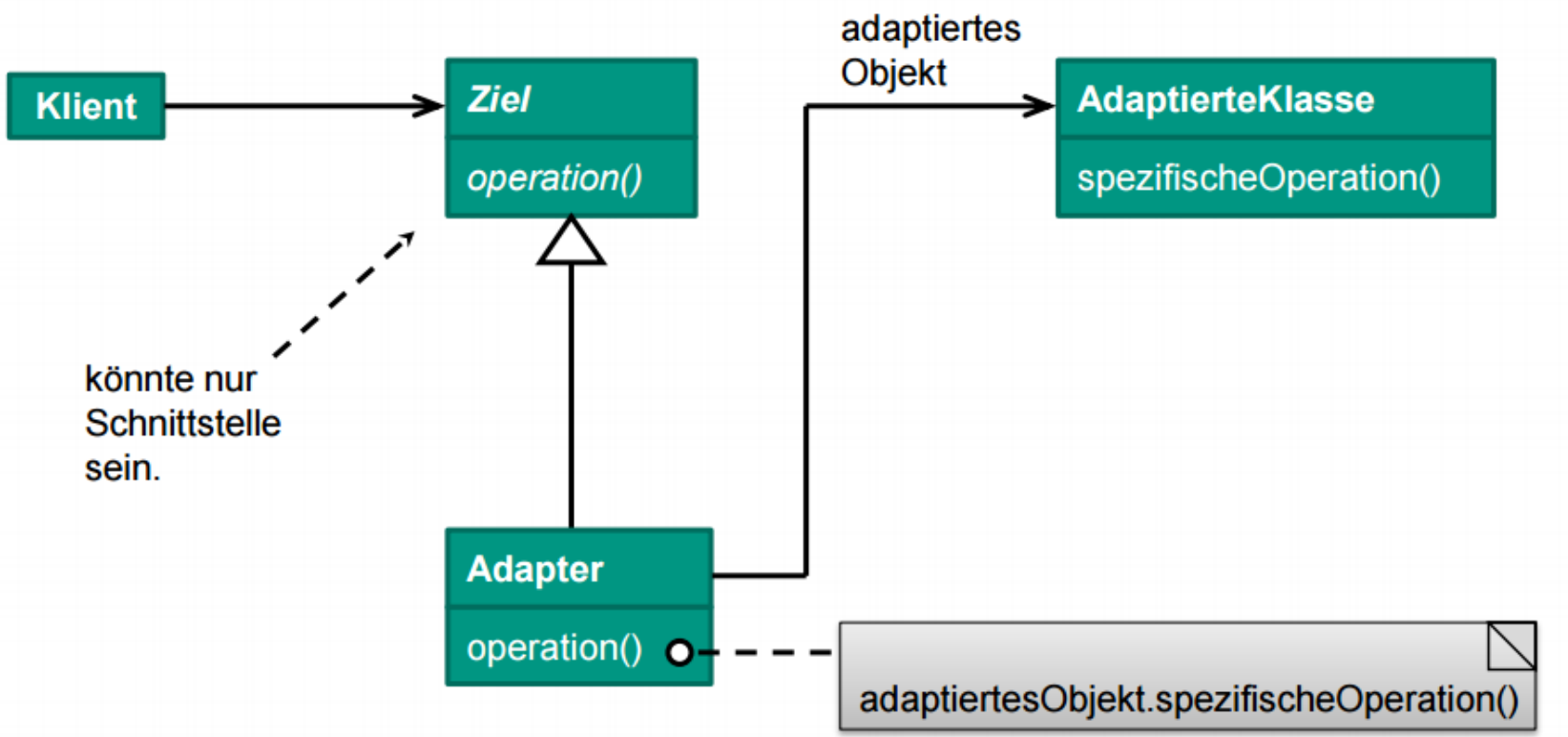
- haben uns Entkopplungmuster angeschaut
  $\Longrightarrow$ Beobachter, Iterator, Adapter, Stellvertreter



Welche Klassen?

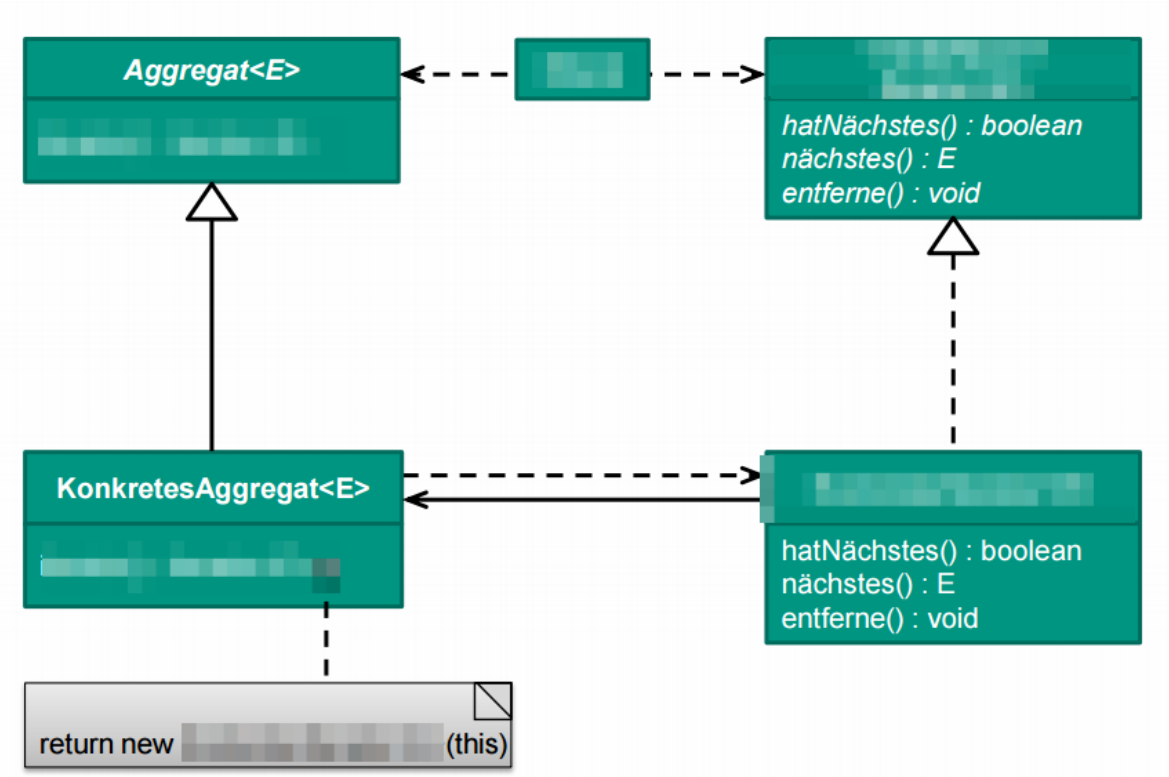
Was bisher geschah..

- haben uns Entkopplungmuster angeschaut
  $\Longrightarrow$ Beobachter, Iterator, Adapter, Stellvertreter



Was bisher geschah..

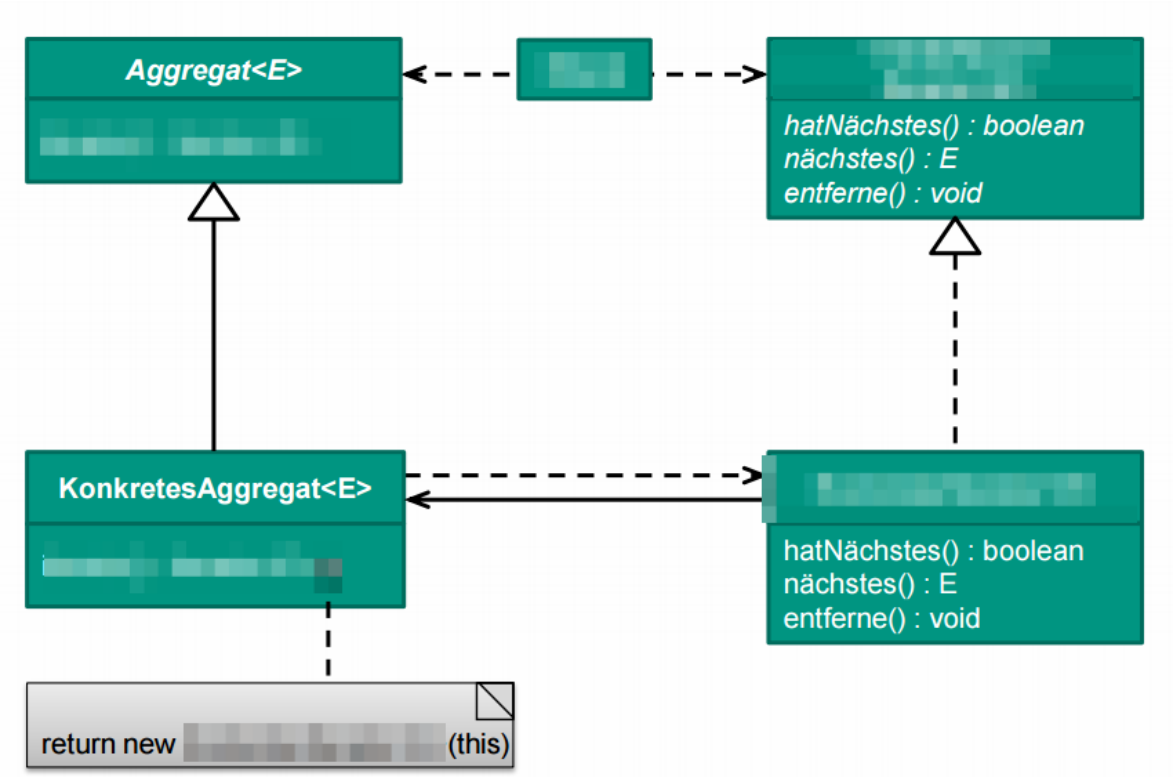
- haben uns Entkopplungmuster angeschaut
  $\Longrightarrow$ Beobachter, Iterator, Adapter, Stellvertreter



Welches Entwurfsmuster?

Was bisher geschah..

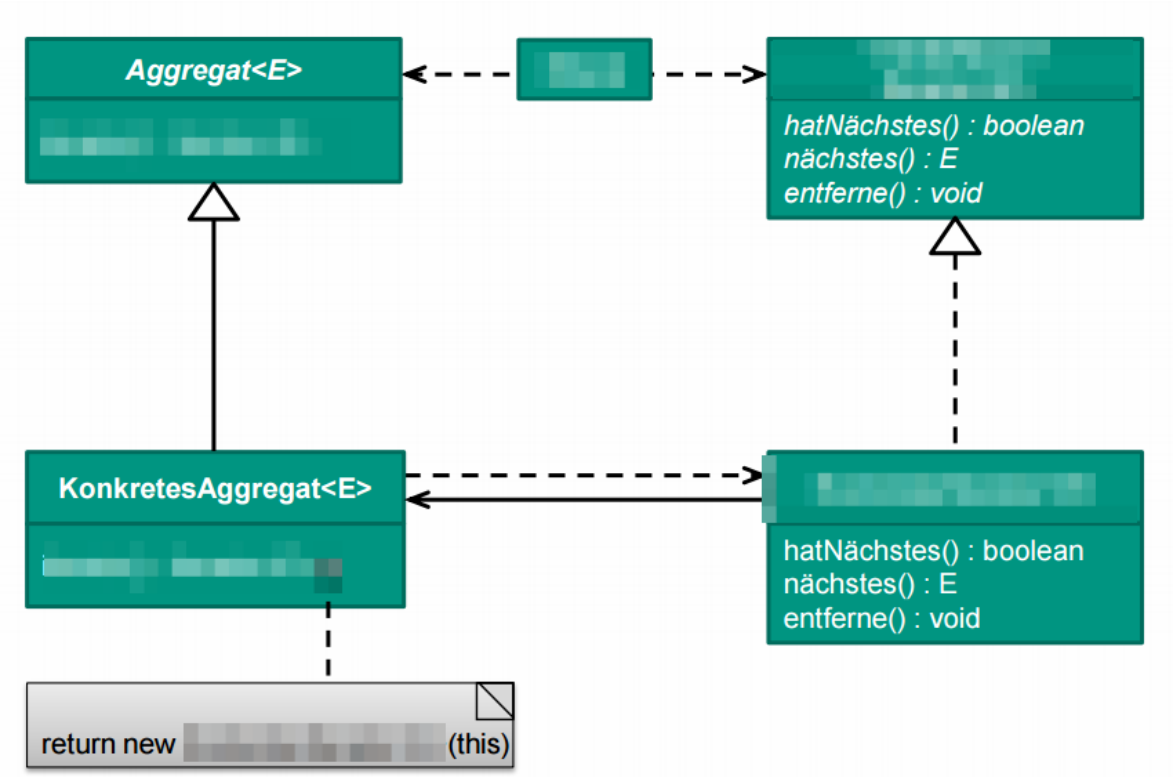
- haben uns Entkopplungmuster angeschaut
  $\Longrightarrow$ Beobachter, Iterator, Adapter, Stellvertreter



Welches Entwurfsmuster? Iterator

Was bisher geschah..

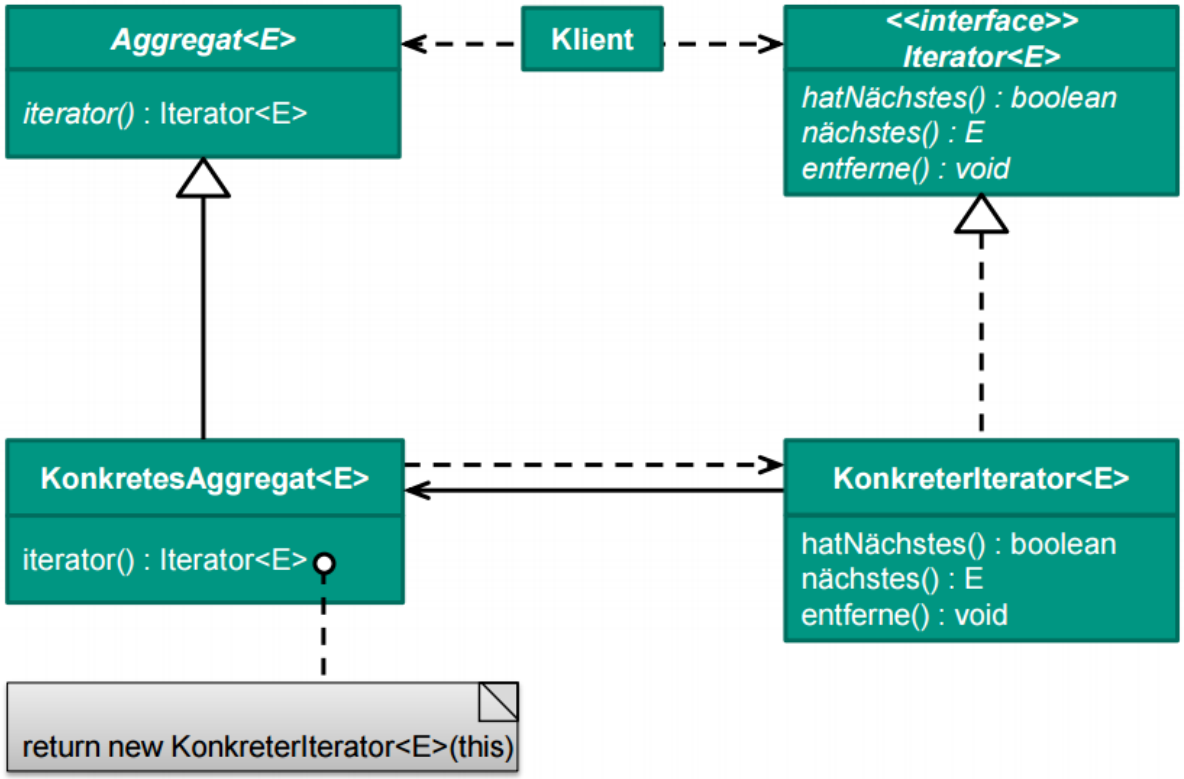
- haben uns Entkopplungmuster angeschaut
  $\Longrightarrow$ Beobachter, Iterator, Adapter, Stellvertreter



Welche Klassen und Methoden?

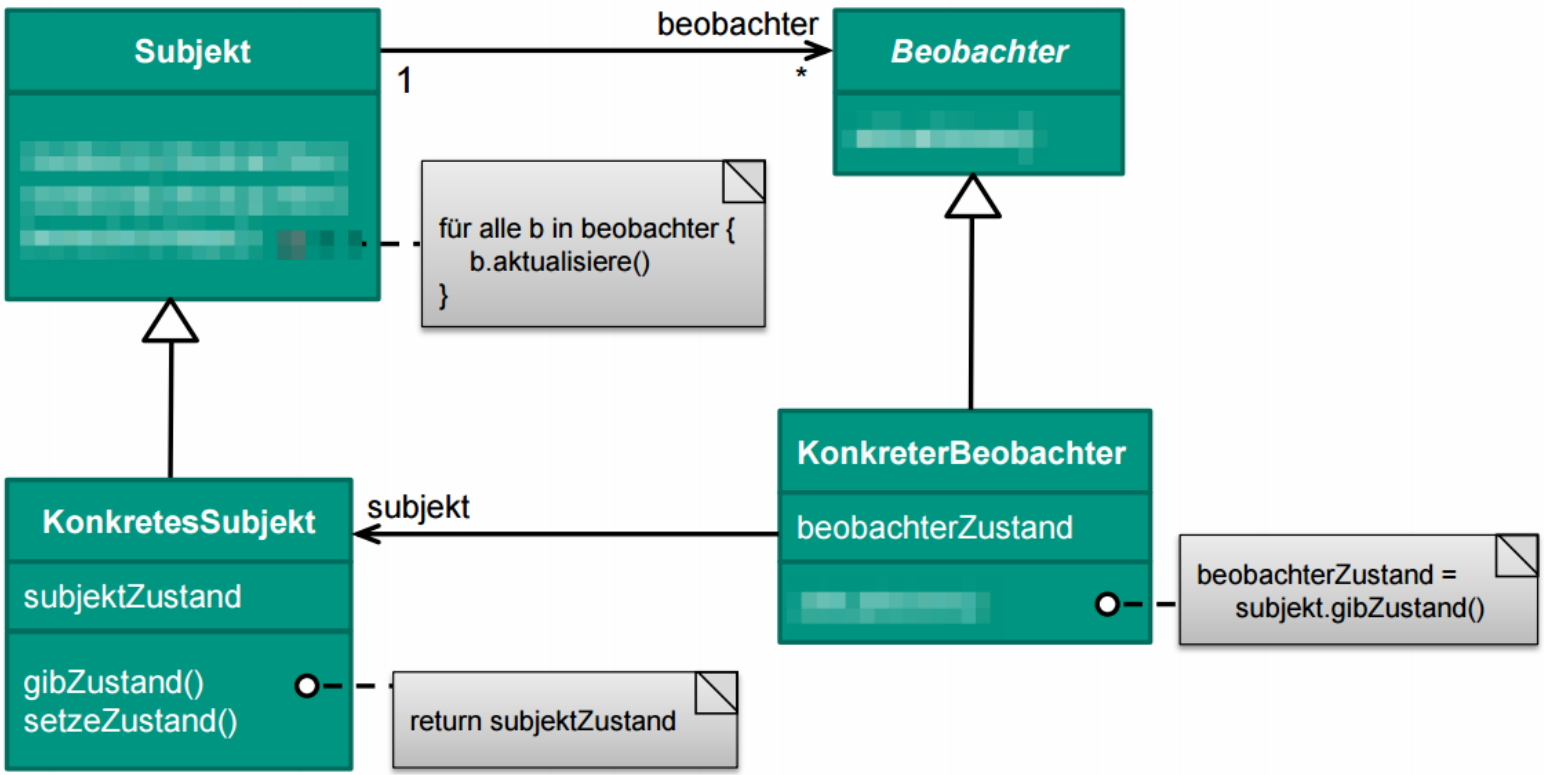
Was bisher geschah..

- haben uns Entkopplungmuster angeschaut
  $\Longrightarrow$ Beobachter, Iterator, Adapter, Stellvertreter



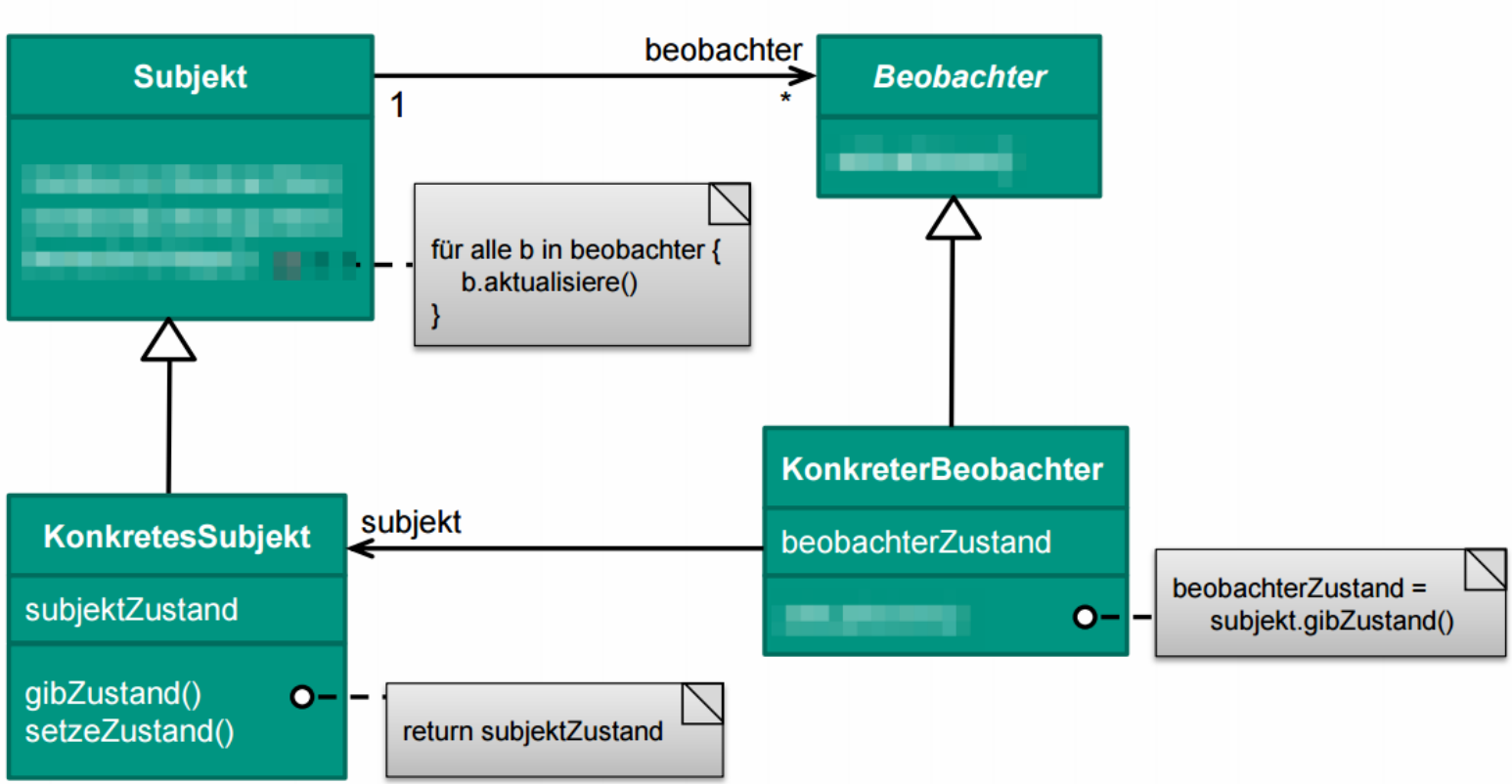
Was bisher geschah..

- haben uns Entkopplungmuster angeschaut
  $\Longrightarrow$ Beobachter, Iterator, Adapter, Stellvertreter



Was bisher geschah..

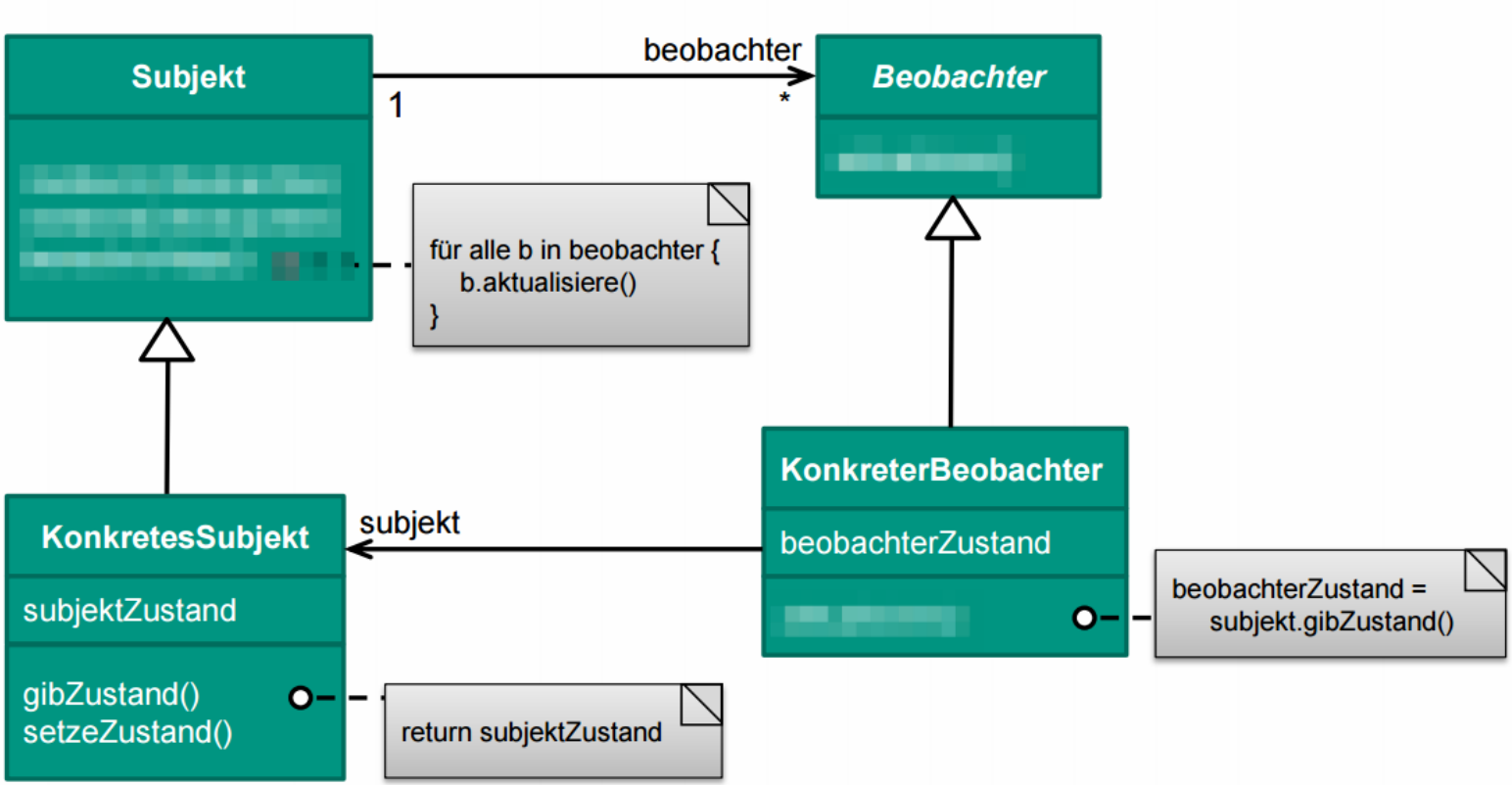
- haben uns Entkopplungmuster angeschaut
  $\Longrightarrow$ Beobachter, Iterator, Adapter, Stellvertreter



Ist wohl ein Beobachter :)

Was bisher geschah..

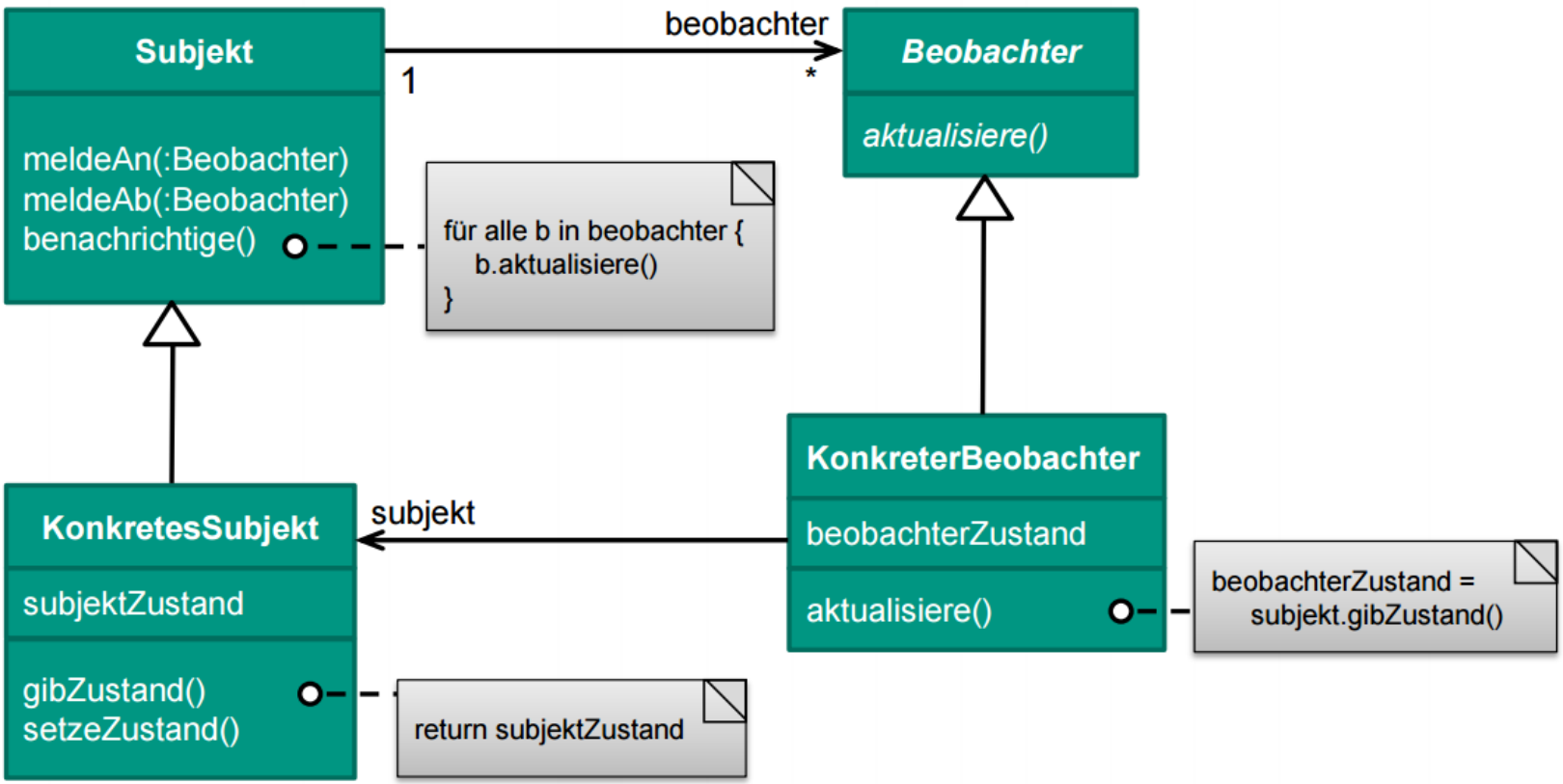
- haben uns Entkopplungmuster angeschaut
  $\Longrightarrow$ Beobachter, Iterator, Adapter, Stellvertreter



Ist wohl ein Beobachter :) Methoden?

Was bisher geschah..

- haben uns Entkopplungmuster angeschaut
  $\Longrightarrow$ Beobachter, Iterator, Adapter, Stellvertreter



Vermittler

Problem

- mehrere voneinander abhängige Objekte

Vermittler

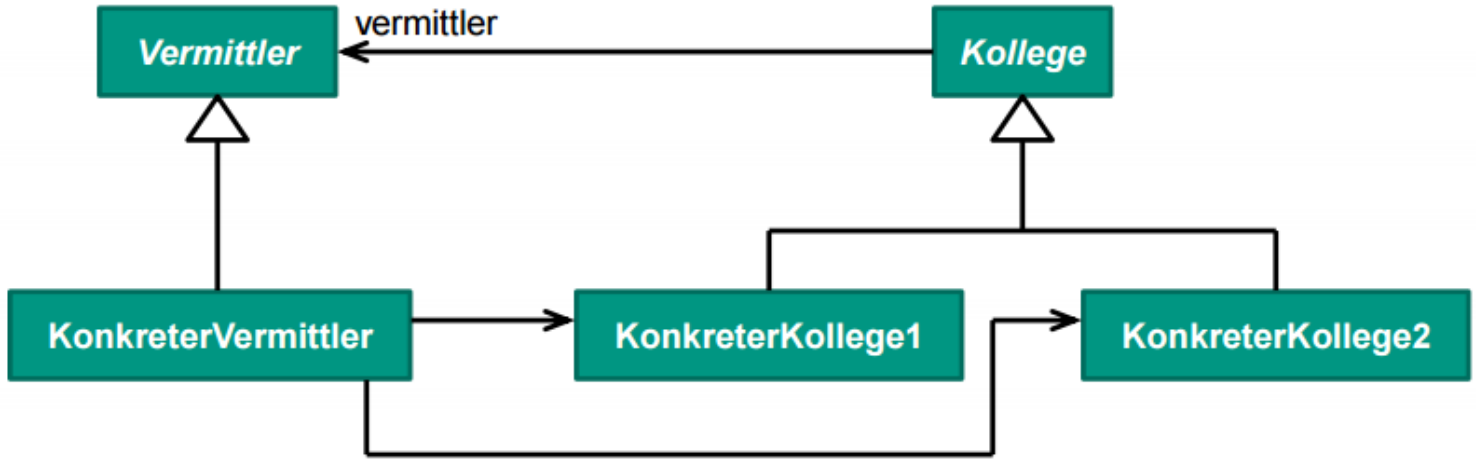
Problem

- mehrere voneinander abhängige Objekte
  $\Longrightarrow$ Zustände der Objekte von anderen Zuständen abhängig

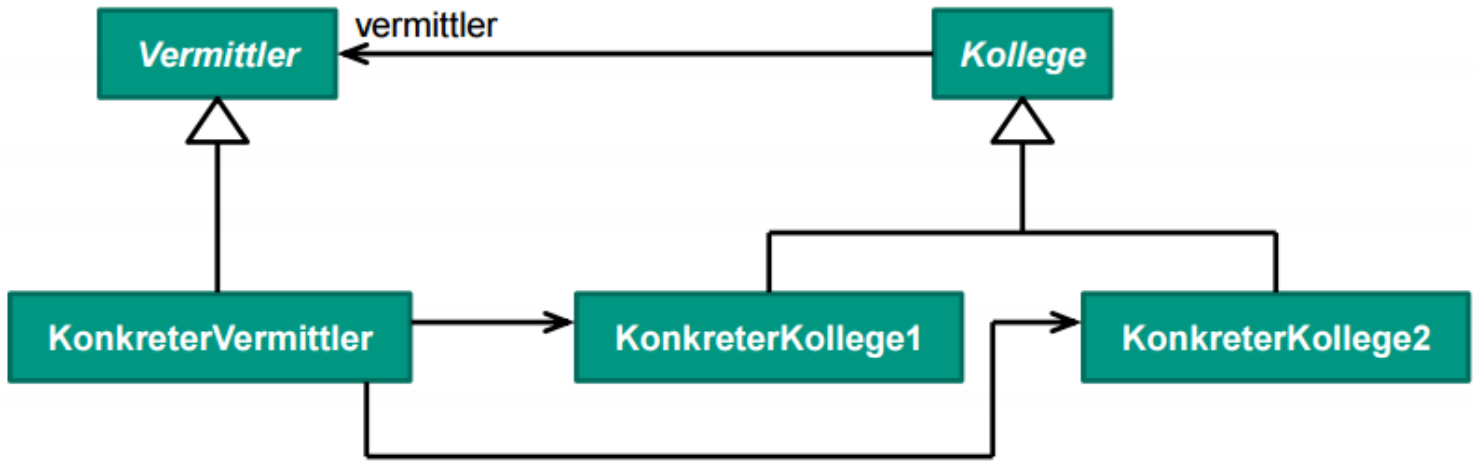
Vermittler

Problem

- mehrere voneinander abhängige Objekte
  $\Longrightarrow$ Zustände der Objekte von anderen Zuständen abhängig

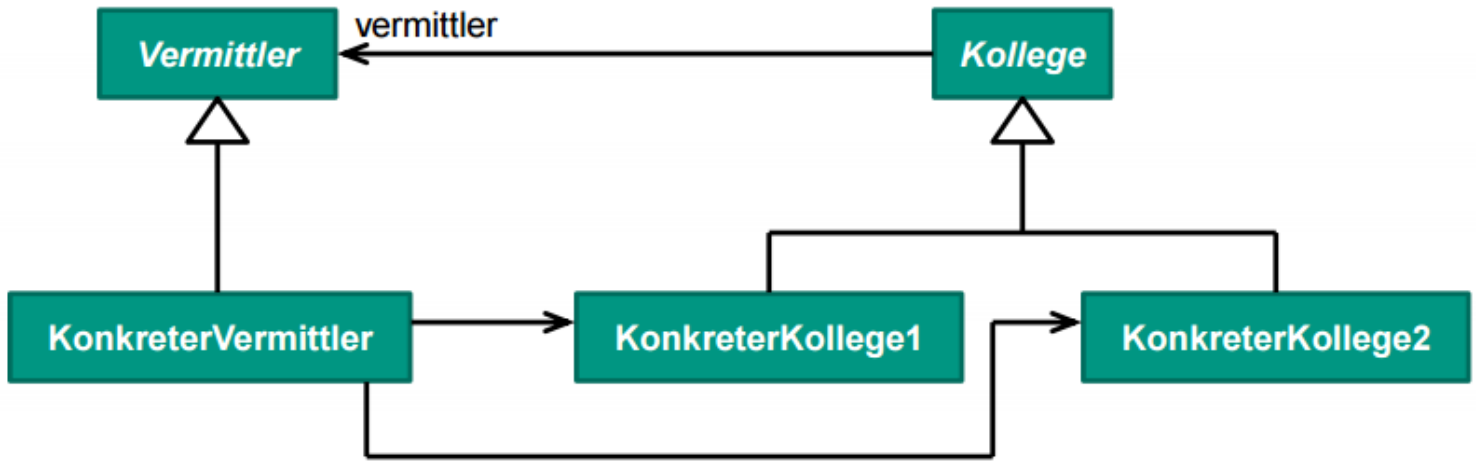


Vermittler



Entkopplung?

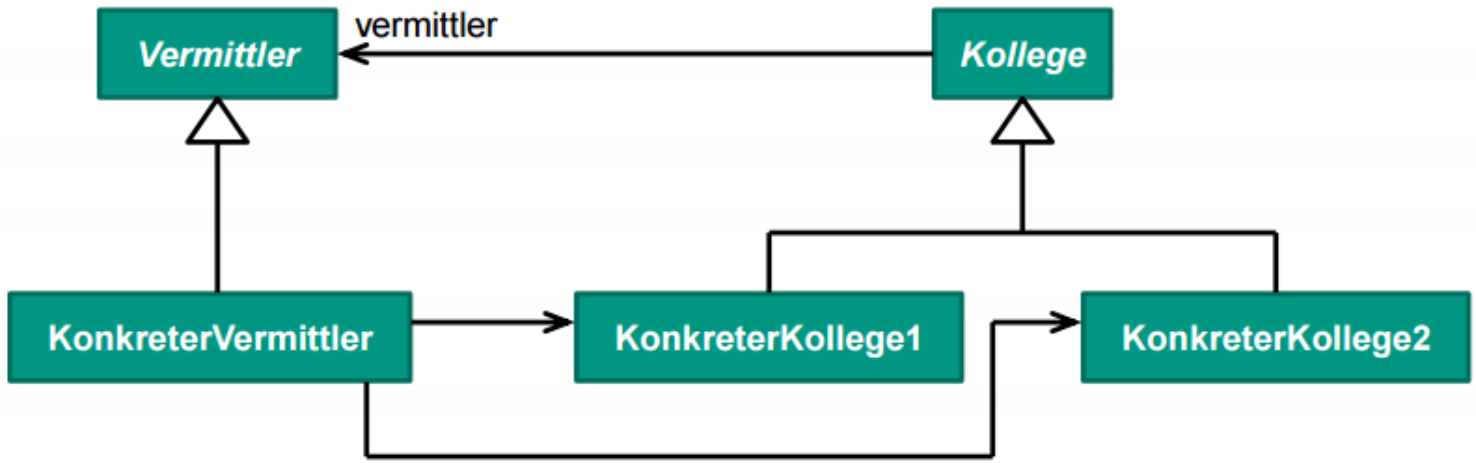
Vermittler



Entkopplung?

- Kollegen kennen sich nicht direkt

Vermittler



Entkopplung?

- Kollegen kennen sich nicht direkt
  $\Longrightarrow$ Hinzufügen eines Kollegen erfordert keine Änderung der alten Kollegen

Beobachter vs. Vermittler

- wirken ähnlich
  - ein Vermittler, viele Kollegen
  - ein Subjekt, viele Beobachter

Beobachter vs. Vermittler

- wirken ähnlich
  - ein Vermittler, viele Kollegen
  - ein Subjekt, viele Beobachter

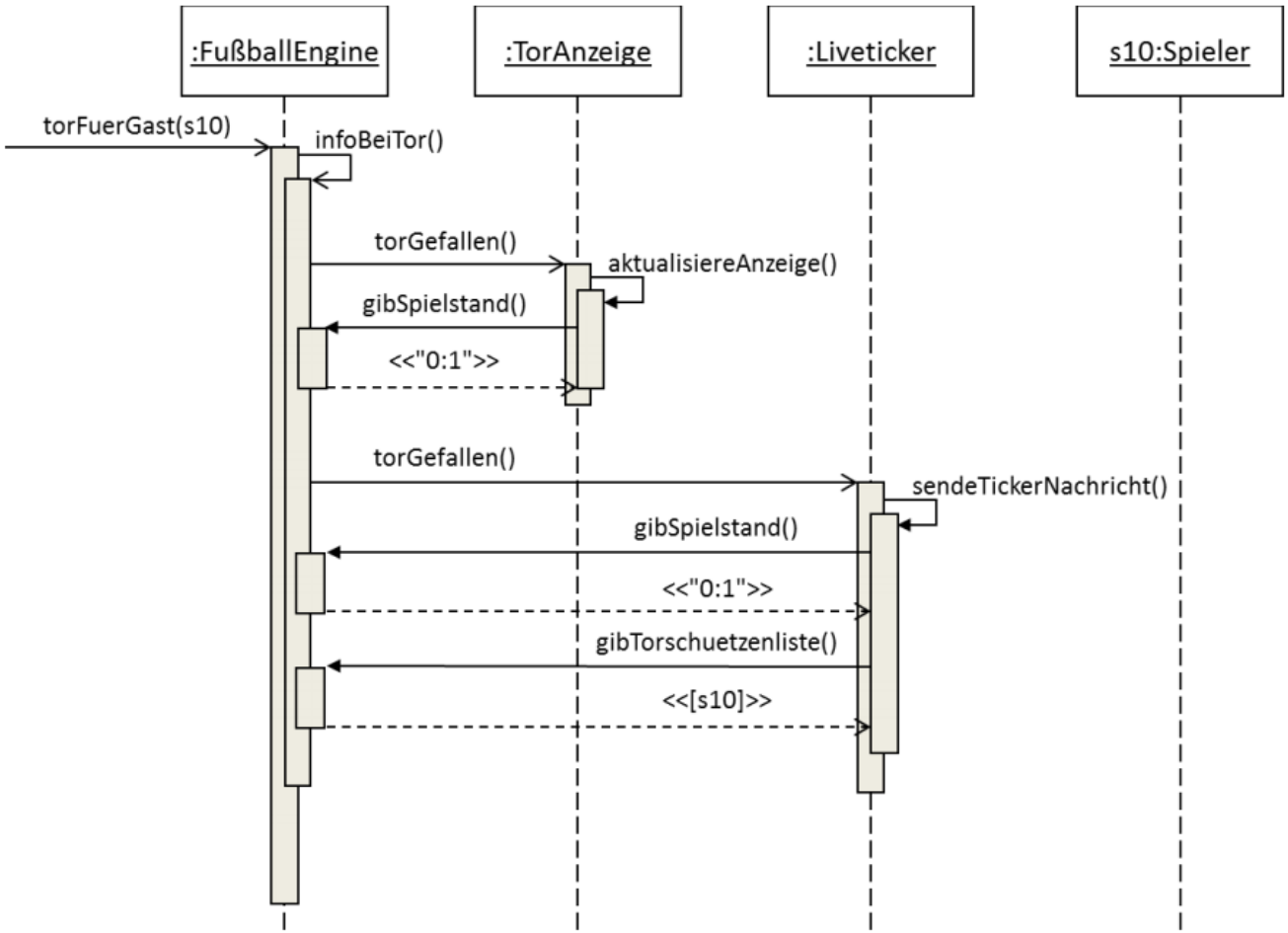
Beobachter

Beobachter interessieren sich nicht füreinander. Nur für das Subjekt

Vermittler

Kollegen interessieren sich füreinander, kommunizieren über den Vermittler miteinander.

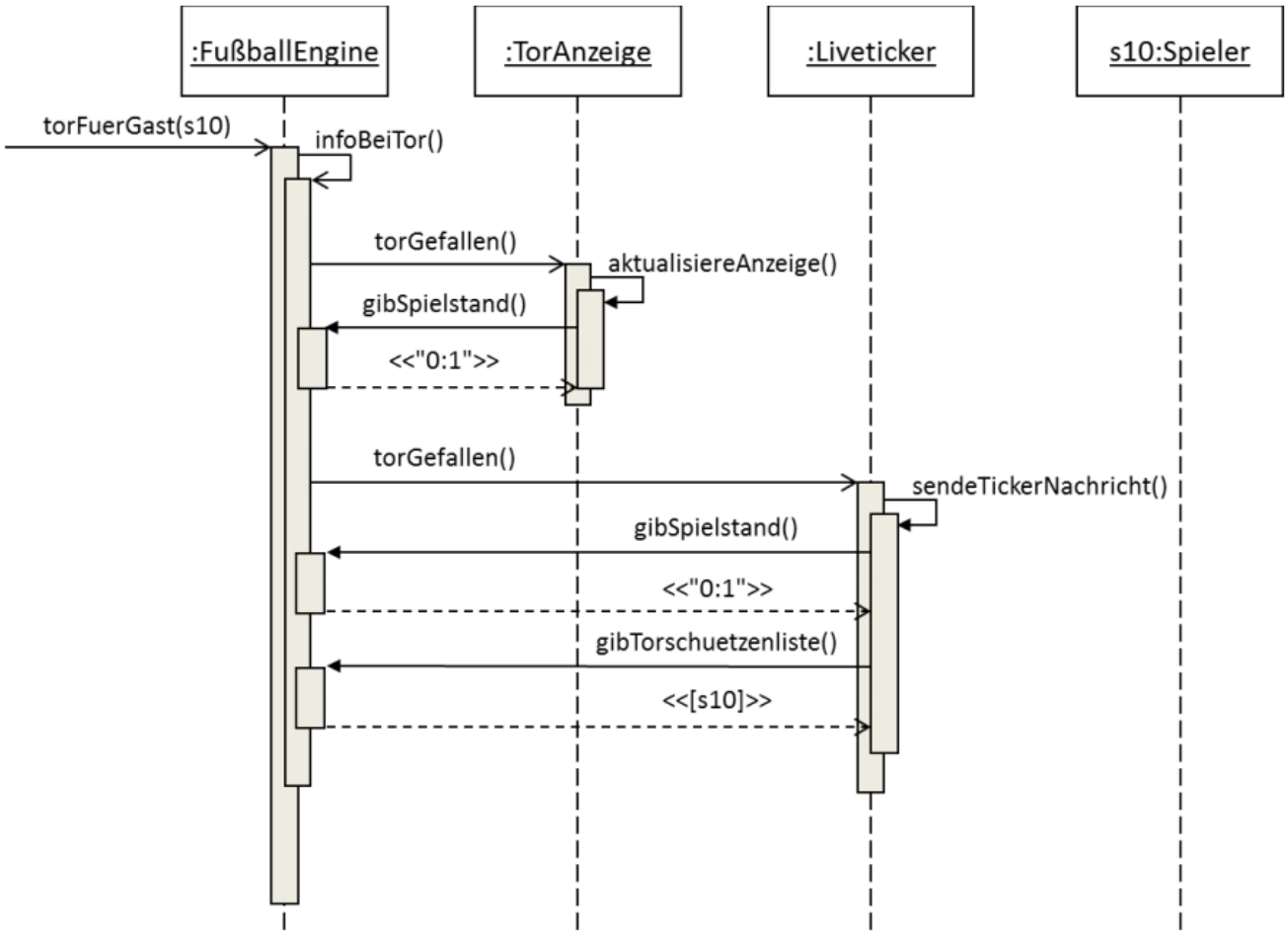
Klausuraufgabe (Hauptklausur SS 2012)



Aufgabe 1

Welches Entwurfsmuster erkennen Sie in diesem Diagramm?

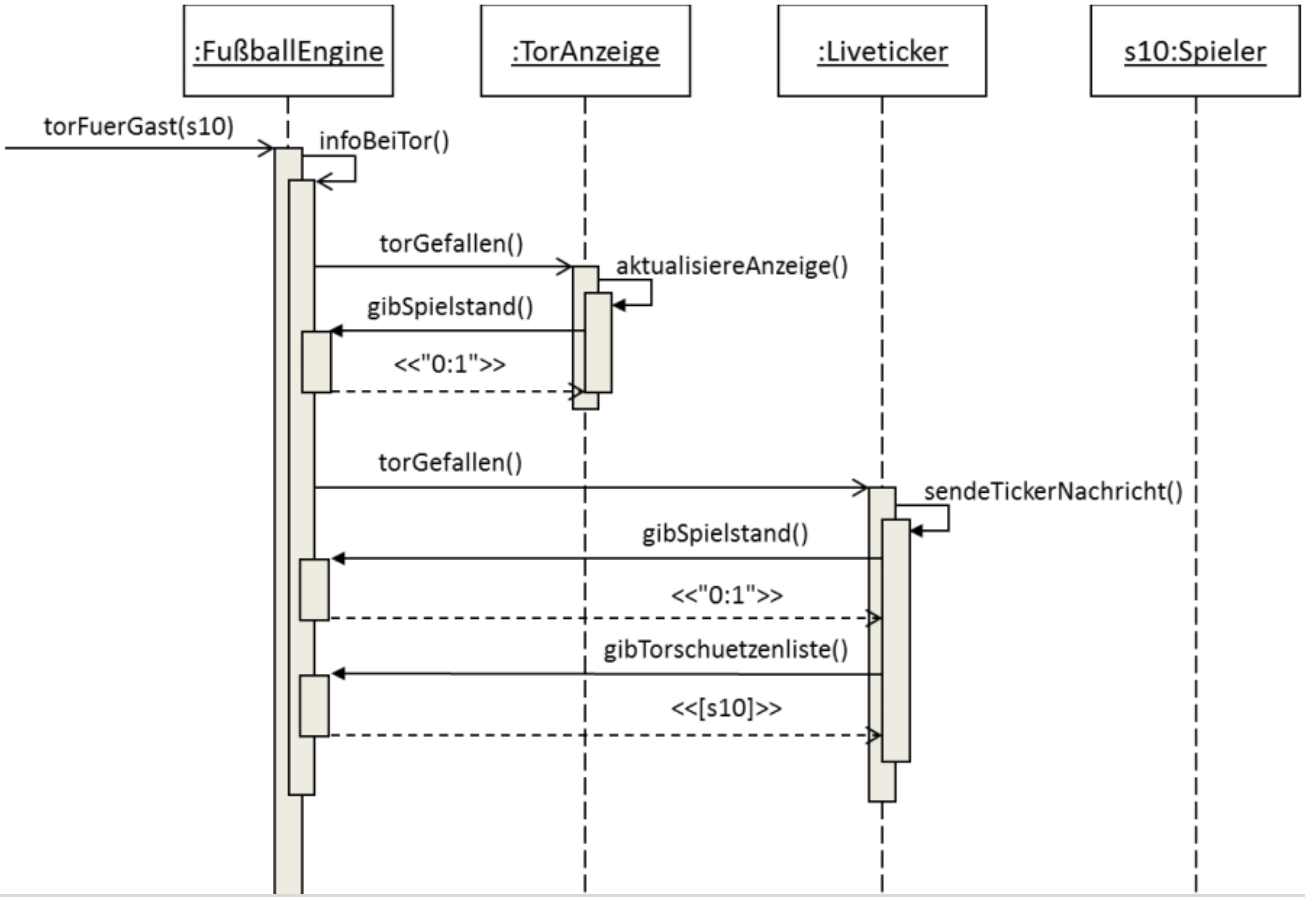
Klausuraufgabe (Hauptklausur SS 2012)



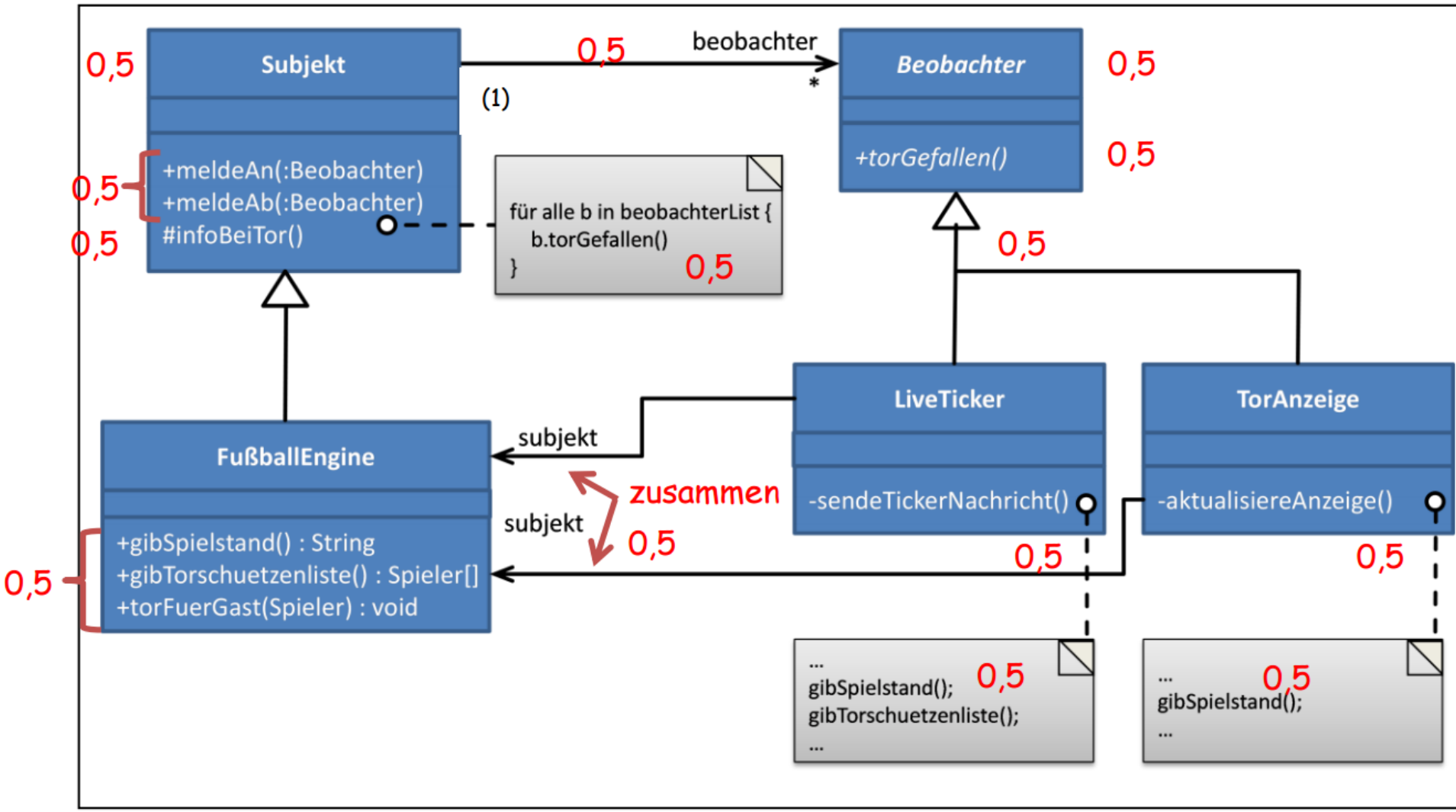
Aufgabe 1

Welches Entwurfsmuster erkennen Sie in diesem Diagramm? Beobachter.

Entwerfen Sie das folgende Klassendiagramm passend zu dem Sequenzdiagramm; es soll alle verwendeten Klassen und Methoden enthalten. Kennzeichnen Sie die Zugreifbarkeiten der Methoden mit den Symbolen +, -, #; seien Sie dabei möglichst restriktiv. Verzichten Sie auf die Modellierung von Attributen. Kennzeichnen Sie die Elemente des Entwurfsmusters und deren Funktion.



Musterlösung



Kategorien der Entwurfsmuster

- **Entkopplungs-Muster**
  - Adapter fertig
  - Beobachter fertig
  - Iterator fertig
  - Stellvertreter fertig
  - Vermittler fertig
  - (Brücke)
- Varianten-Muster
- Zustandshandhabungs-Muster
- Steuerungs-Muster
- Bequemlichkeits-Muster

Kategorien der Entwurfsmuster

- Entkopplungs-Muster fertig
- **Varianten-Muster**
  - (Abstrakte Fabrik)
  - (Besucher)
  - **Schablonenmethode**
  - **Fabrikmethode**
  - **Kompositum**
  - Strategie fertig
  - **Dekorierer**
- Zustandshandhabungs-Muster
- Steuerungs-Muster
- Bequemlichkeits-Muster

Varianten-Muster

Übergeordnetes Ziel

- Gemeinsamkeiten herausziehen und an einer Stelle beschreiben

Varianten-Muster

Übergeordnetes Ziel

- Gemeinsamkeiten herausziehen und an einer Stelle beschreiben
  $\Longrightarrow$ keine Wiederholung desselben Codes

Varianten-Muster

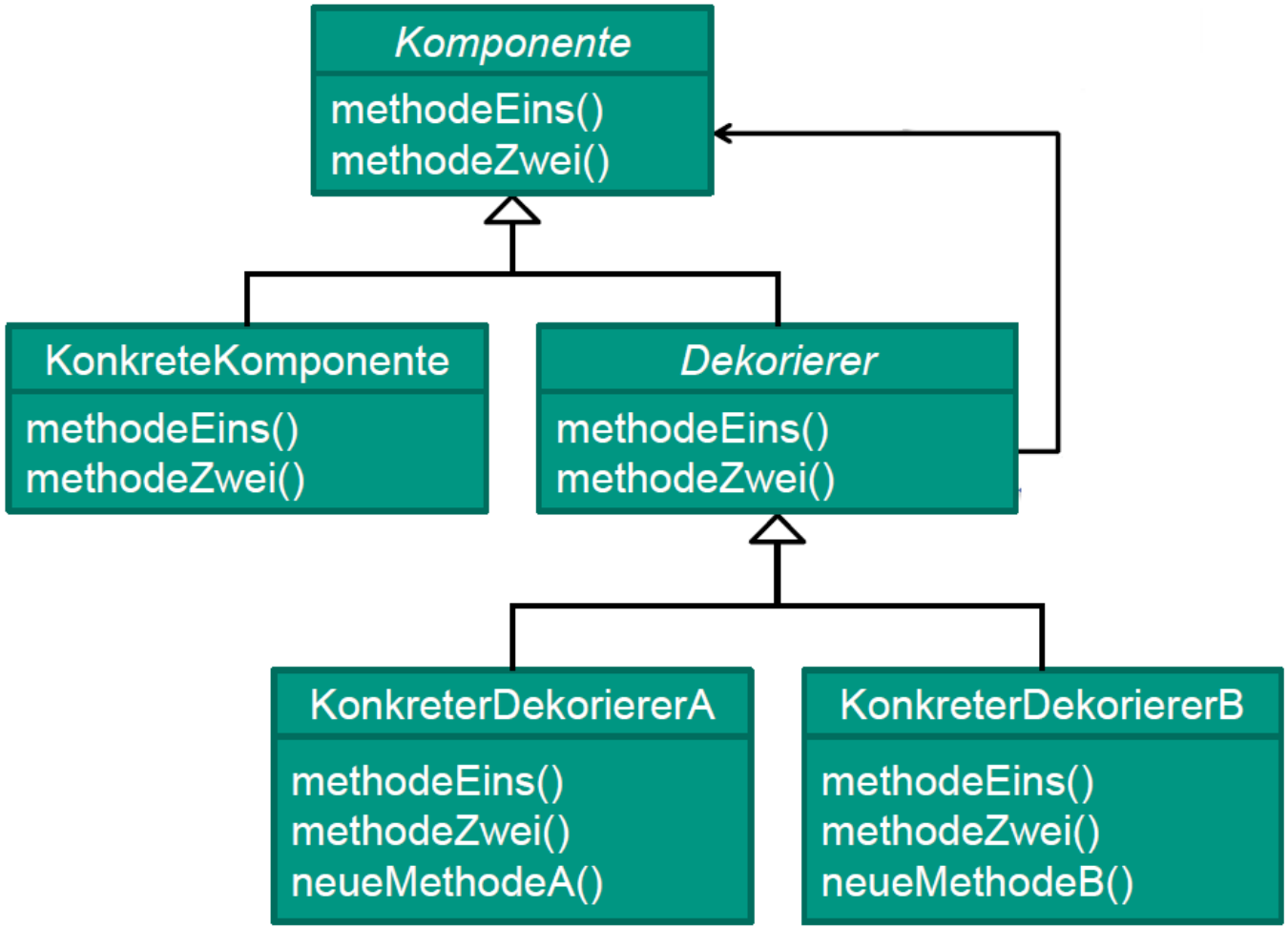
Übergeordnetes Ziel

- Gemeinsamkeiten herausziehen und an einer Stelle beschreiben
  - $\Longrightarrow$ keine Wiederholung desselben Codes
  - $\Longrightarrow$ bessere Wartbarkeit/Erweiterbarkeit

Jetzt: Gruppenarbeit

1. ihr kriegt pro Reihe eine Aufgabe
2. ihr habt Zeit zum Bearbeiten
3. Abgleichung mit Musterlösung
4. ihr stellt den anderen eure Lösung vor

Vorstellung Dekorierer



MuLö Dekorierer

Wo Gemeinsamkeiten?

Die beiden Methoden methodeEins() und methodeZwei().

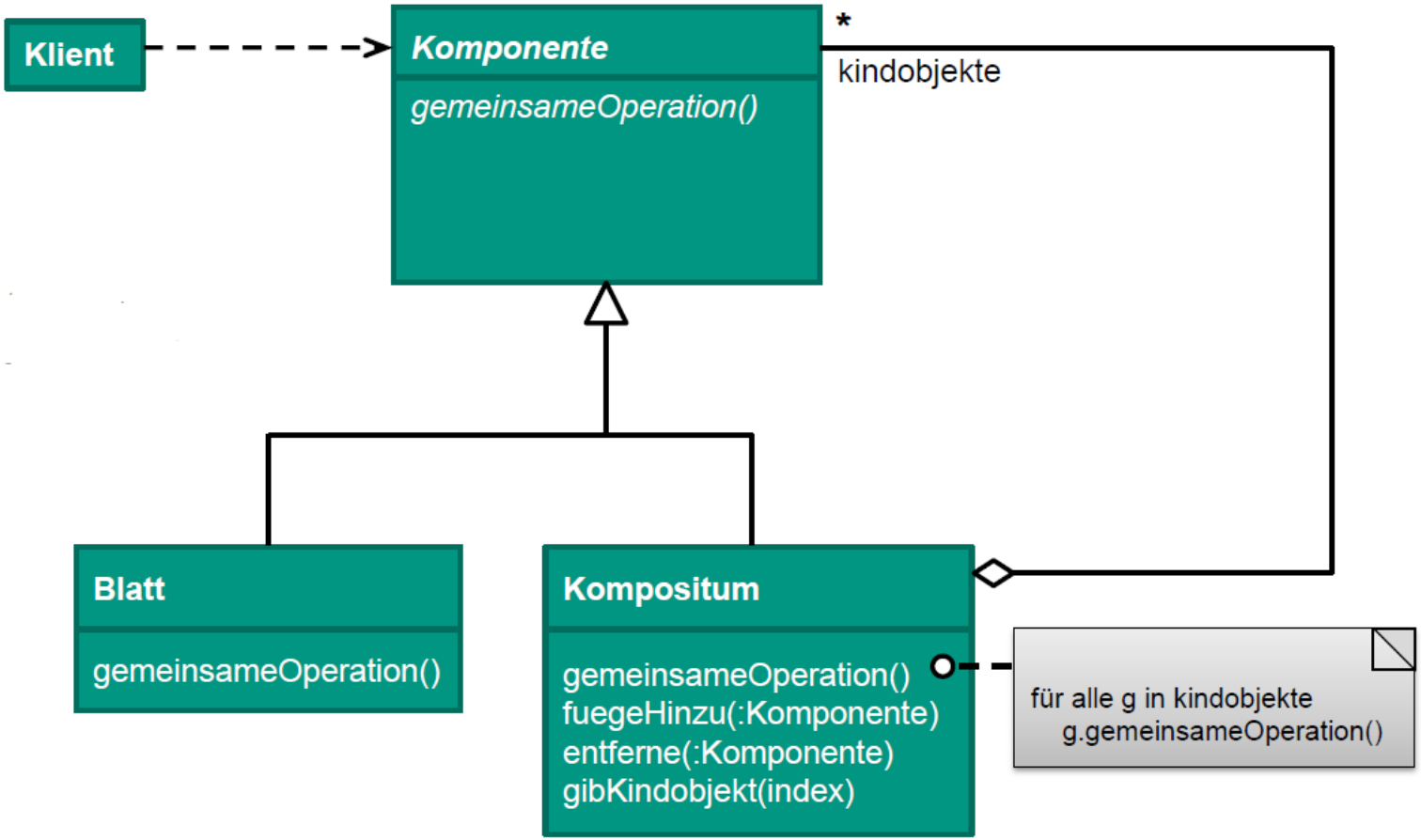
Wo Variation?

In den KonkretenDekorierern bzw. ihren Methoden. Hier: neueMethodeA(), neueMethodeB().

Wozu Instanzvariable?

Weiterleitung von Aufrufen der methodeEins() und methodeZwei() an die KonkreteKompenente.

Vorstellung Kompositum



MuLö Kompositum

Wo Gemeinsamkeiten?

gemeinsameOperation().

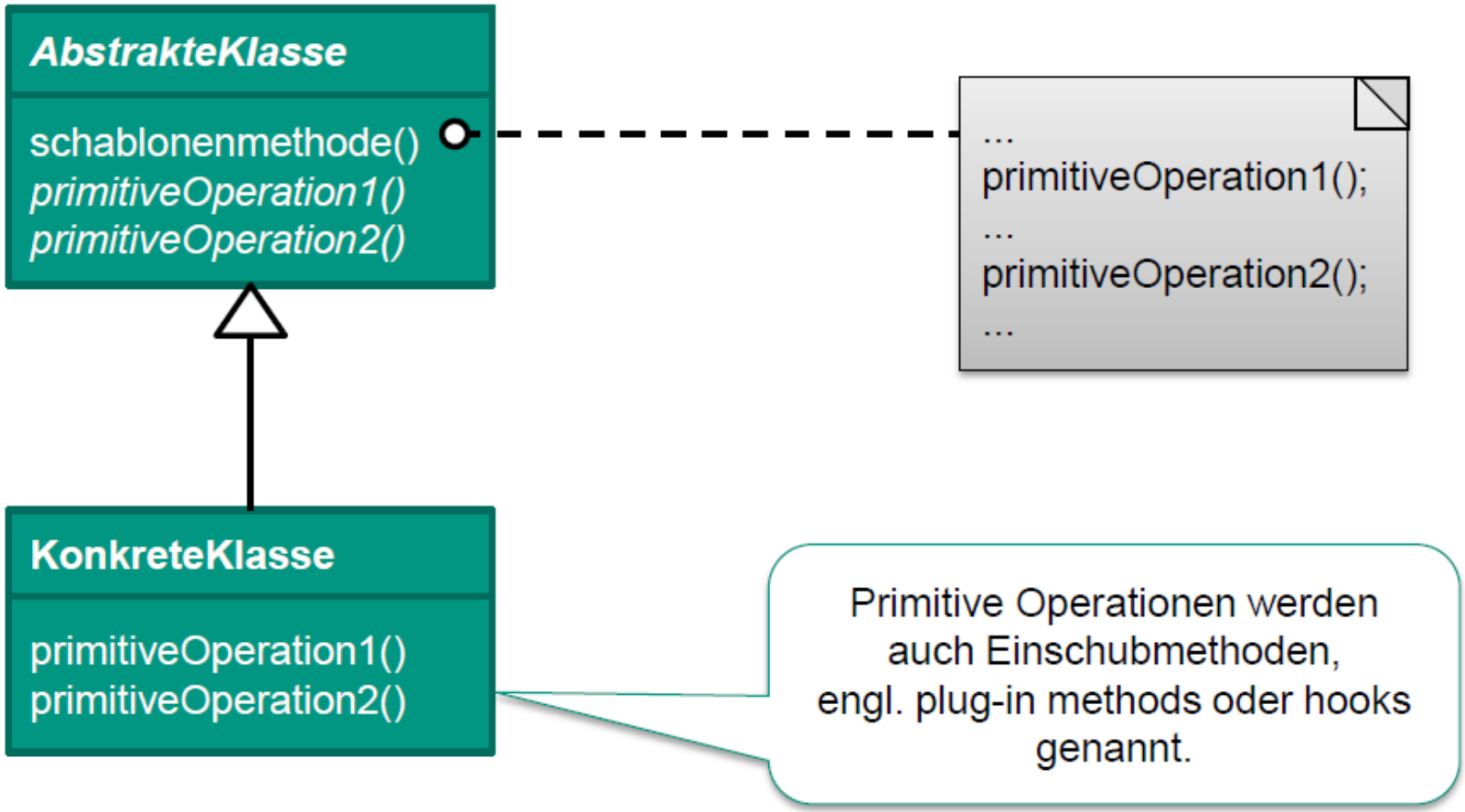
Wo Variation?

In Blatt/Kompositum-Klassen mit verschiedenen zusätzlichen Operationen.

Zusammengesetzt vs. nicht-zusammengesetzt

Kompositum = zusammengesetzt, Blatt = nicht-zusammengesetzt

Vorstellung Schablonenmethode



MuLö Schablonenmethode

Wo Gemeinsamkeiten?

Reihenfolge der Methodenaufrufe in der Schablonenmethode.

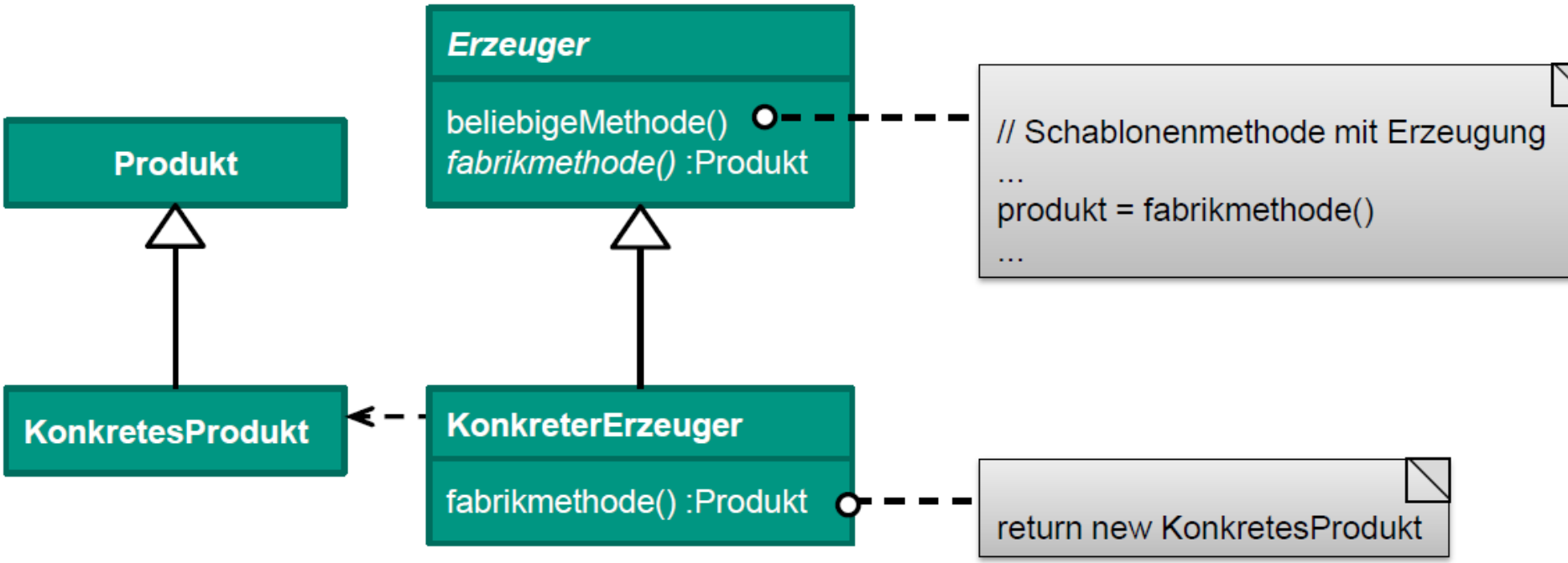
Wo Variation?

In den Einschubmethoden. (hier: primitiveOperation1() und primitiveOpoeration2())

Schablonenmethode vs. Einschubmethode

Einschubmethode ist eine der Methoden, die von der Schablonenmethode aufgerufen wird und deren Implementierung in den Unterklassen stattfindet.

Vorstellung Fabrikmethode



MuLö Fabrikmethode

Wo Gemeinsamkeiten?

Reihenfolge der Methodenaufrufe in der beliebigenMethode().

Wo Variation?

In der Fabrikmethode.

Klasse des Objekts, Oberklasse, Unterklasse

Klasse des Objekts = KonkretesProdukt, Oberklasse = Produkt, Unterklasse = KonkreterErzeuger

Unterschied zu Schablonenmethode?

Fabrikmethode benutzen, wenn ein Objekt erzeugt wird. Fabrikmethode ist Einschubmethode des Musters “Schablonenmethode“.

Wahr/falsch

Fabrikmethode ist eine Einschubmethode, keine Schablonenmethode.

Kategorien der Entwurfsmuster

- Entkopplungs-Muster fertig
- Varianten-Muster fertig
- **Zustandshandhabungs-Muster**
  - **Einzelstück**
  - (Fliegengewicht)
  - **Memento**
  - (Prototyp)
  - (Zustand)
- Steuerungs-Muster
- Bequemlichkeits-Muster

Zustandshandhabungs-Muster

Übergeordnetes Ziel

- den Zustand eines Objektes beschreiben (wer hätt's gedacht? :D)

Zustandshandhabungs-Muster

Übergeordnetes Ziel

- den Zustand eines Objektes beschreiben (wer hätt's gedacht? :D)
- aber unabhängig von dem Zweck des Objekts!

Einzelstück/Singleton

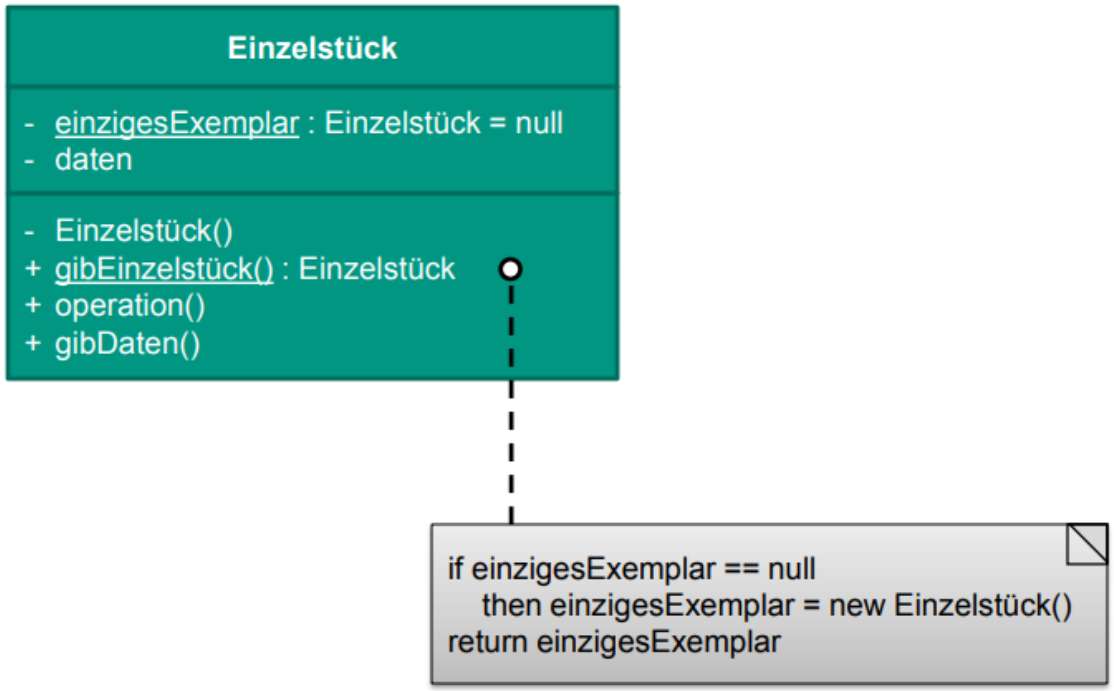
Problem

- von einer Klasse soll nur eine Instanz existieren
- Konstruktor könnte überall benutzt werden!

Einzelstück/Singleton

Problem

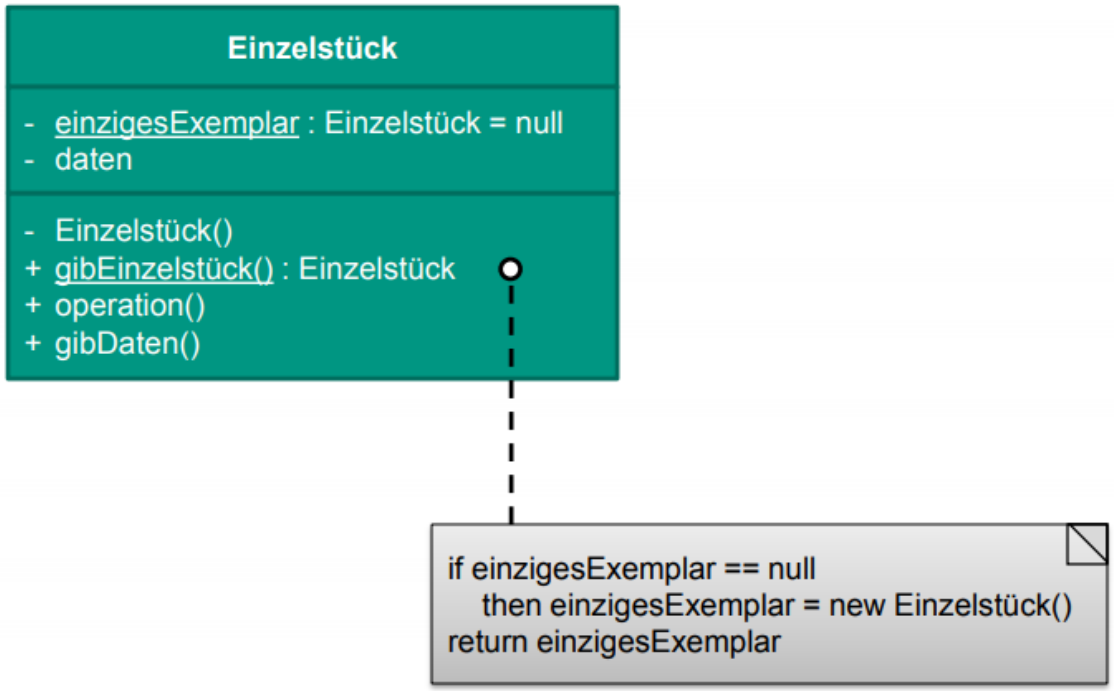
- von einer Klasse soll nur eine Instanz existieren
- Konstruktor könnte überall benutzt werden!



Einzelstück/Singleton

Problem

- von einer Klasse soll nur eine Instanz existieren
- Konstruktor könnte überall benutzt werden!

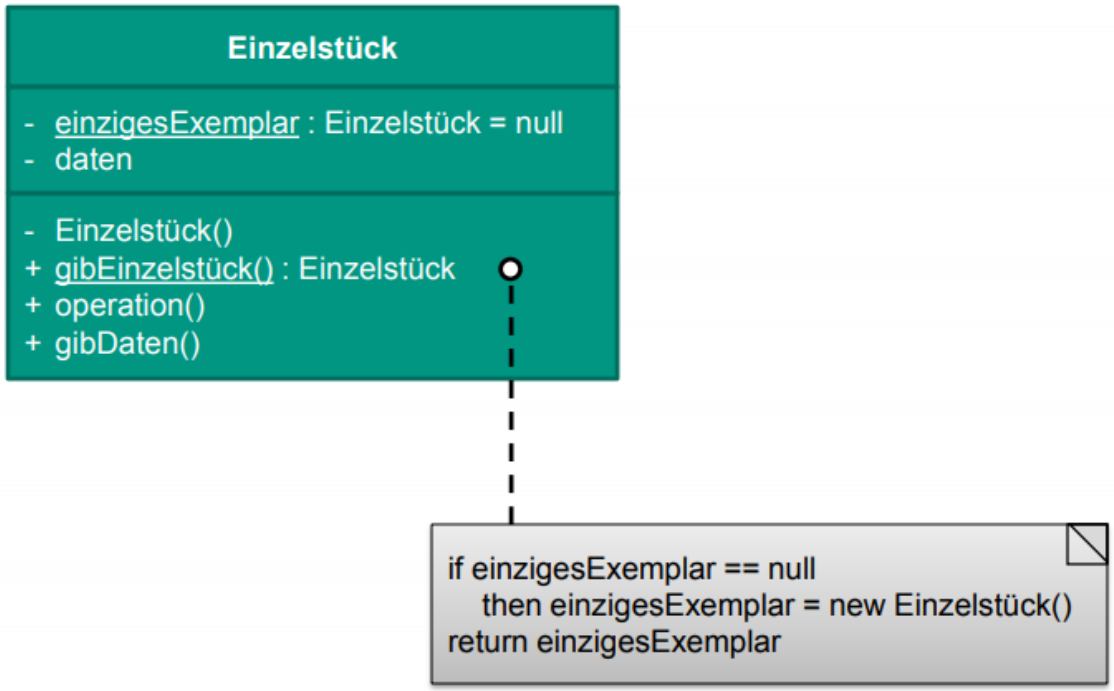


Aber warum nicht einfach statisch?

Einzelstück/Singleton

Problem

- von einer Klasse soll nur eine Instanz existieren
- Konstruktor könnte überall benutzt werden!



Aber warum nicht einfach statisch? Unterklassenbildung möglich!

Memento

Problem

- internen Zustand eines Objekts “externalisieren“, um z.B. Zurücksetzen möglich zu machen

Memento

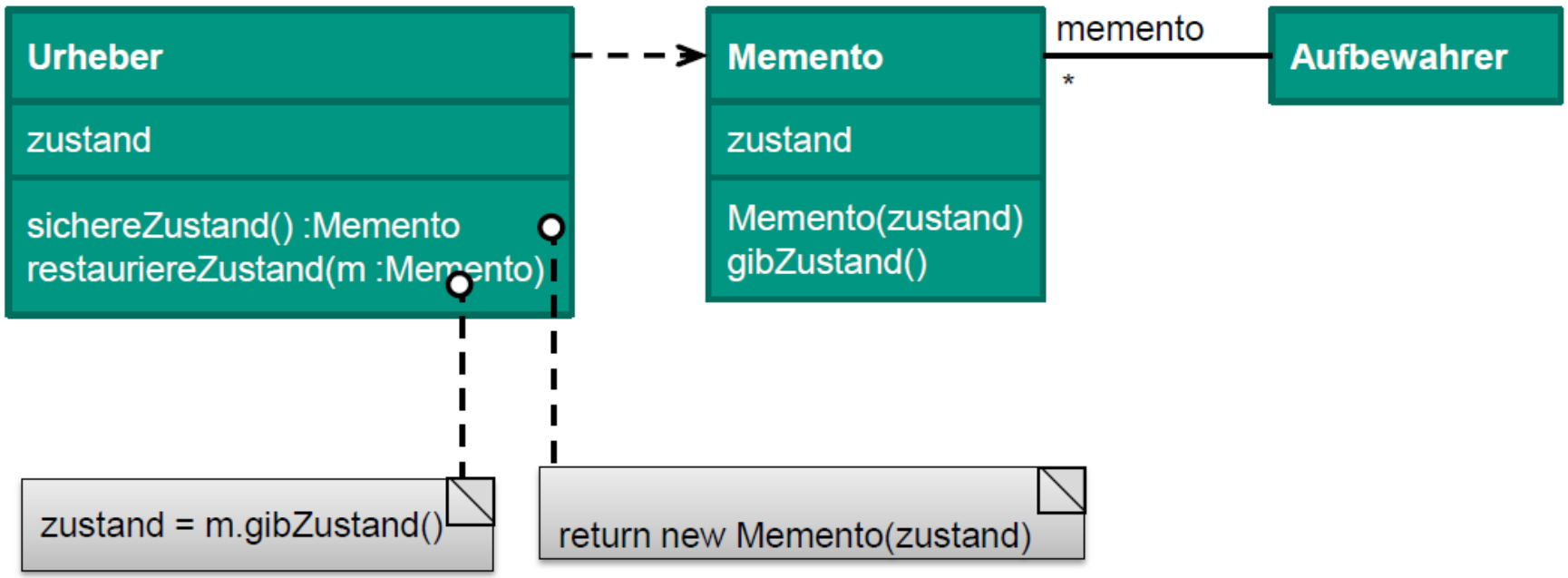
Problem

- internen Zustand eines Objekts “externalisieren“, um z.B. Zurücksetzen möglich zu machen
- ohne Kapselung zu verletzten!

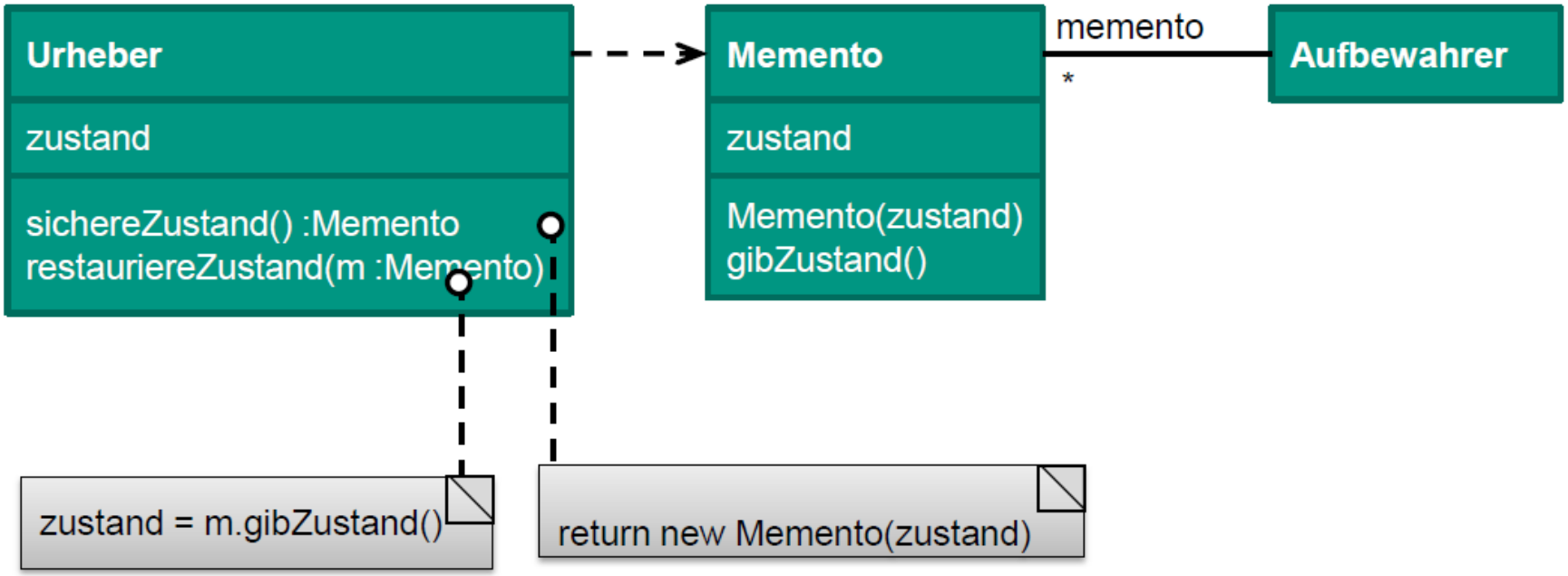
Memento

Problem

- internen Zustand eines Objekts “externalisieren“, um z.B. Zurücksetzen möglich zu machen
- ohne Kapselung zu verletzten!

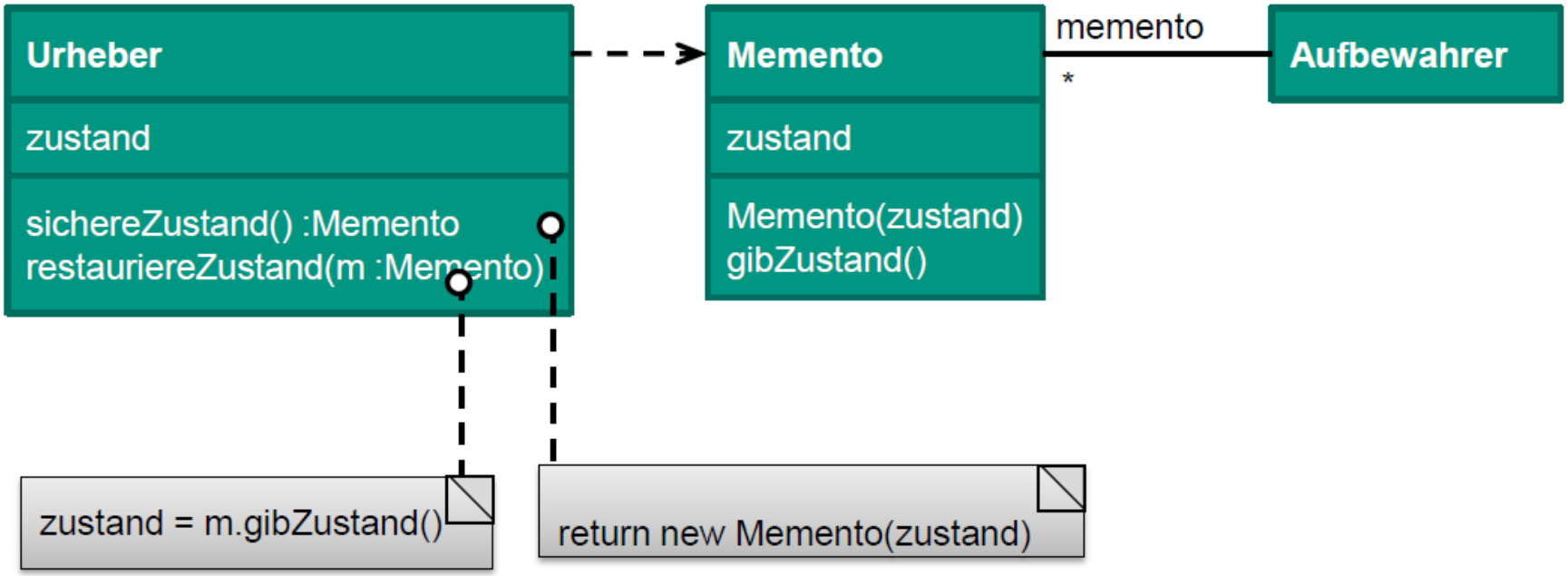


Memento



Problem gelöst?

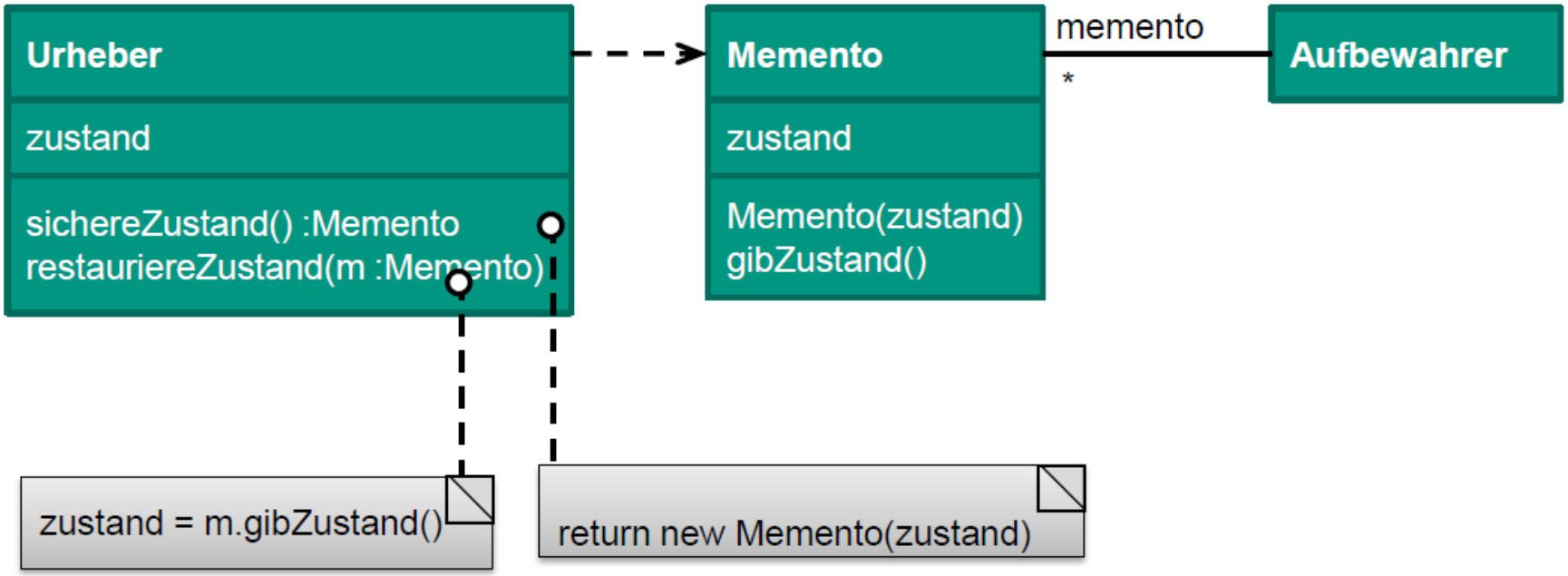
Memento



Problem gelöst?

- Ja

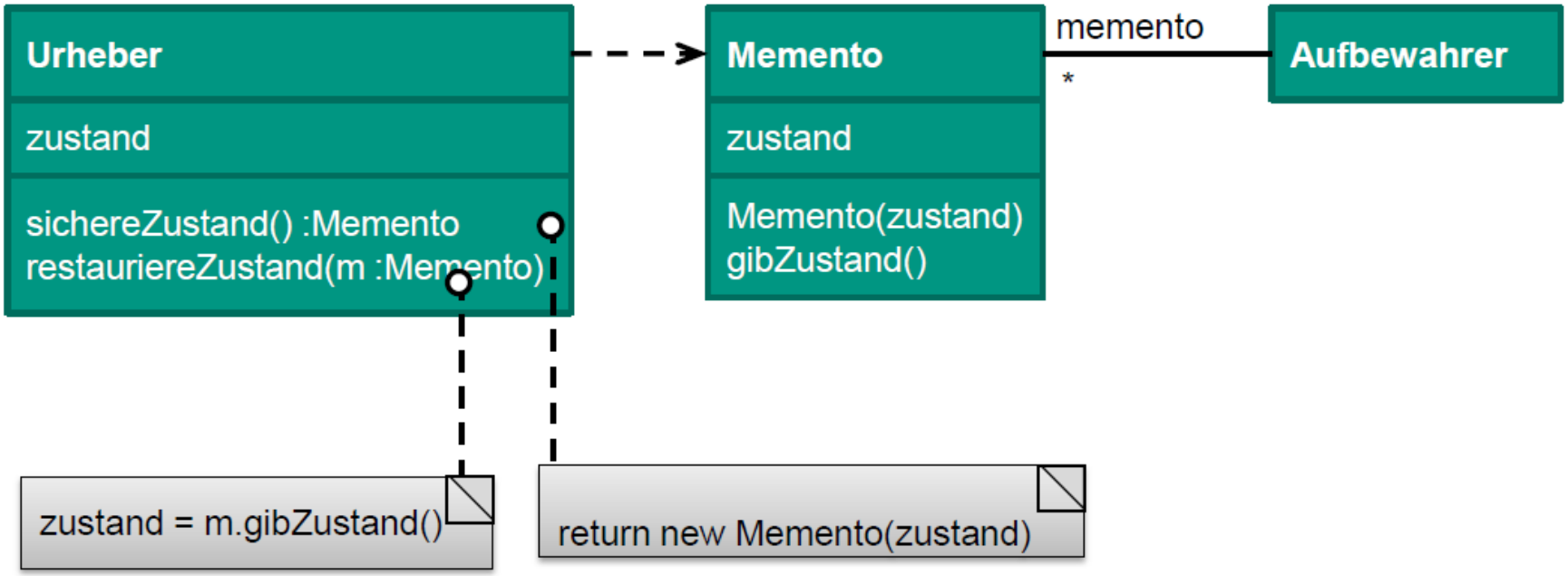
Memento



Problem gelöst?

- Ja
  - Zustand durch Memento externalisiert

Memento



Problem gelöst?

- Ja
  - Zustand durch Memento externalisiert
  - Kapselung nicht verletzt (Nutzer ruft nur sichereZustand() auf und kriegt neuen Memento)

Kategorien der Entwurfsmuster

- Entkopplungs-Muster fertig
- Varianten-Muster fertig
- Zustandshandhabungs-Muster fertig
- **Steuerungs-Muster**
  - **Befehl**
  - (master/worker)
- Bequemlichkeits-Muster

Steuerungs-Muster

Übergeordnetes Ziel

- steuern den Kontrollfluss

Steuerungs-Muster

Übergeordnetes Ziel

- steuern den Kontrollfluss
  $\Longrightarrow$ zur richtigen Zeit richtige Methoden aufrufen

Befehl

Problem

- Parametrisieren von Objekten mit einer auszuführenden Aktion

Befehl

Problem

- Parametrisieren von Objekten mit einer auszuführenden Aktion
- komplexe Operationen aus primitiven Operationen aufbauen

Befehl

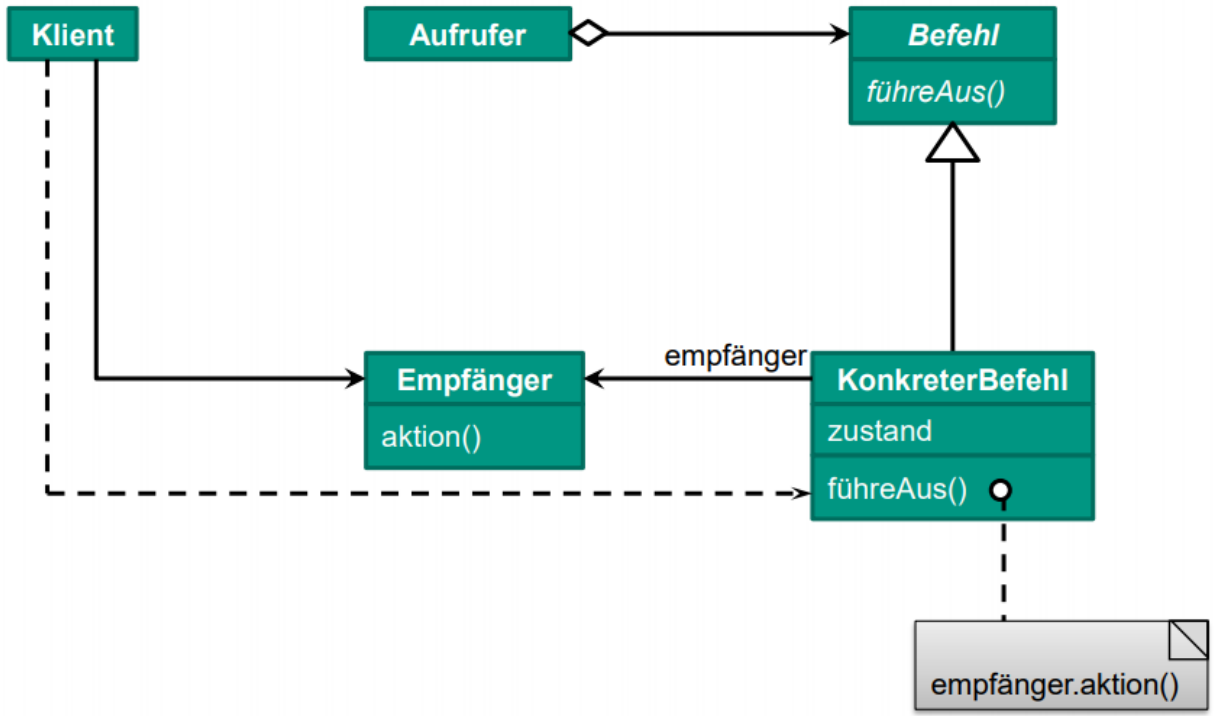
Problem

- Parametrisieren von Objekten mit einer auszuführenden Aktion
- komplexe Operationen aus primitiven Operationen aufbauen
  $\Longrightarrow$ Befehl nicht als Methode, sondern als Objekt modellieren

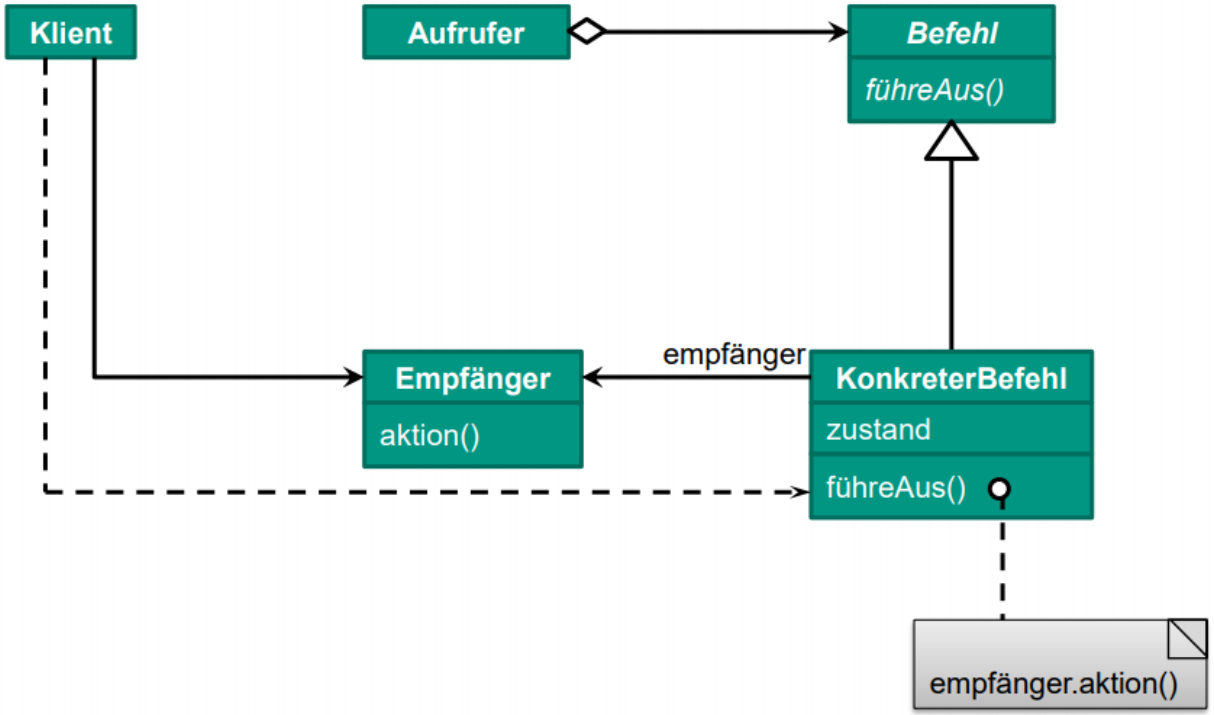
Befehl

Problem

- Parametrisieren von Objekten mit einer auszuführenden Aktion
- komplexe Operationen aus primitiven Operationen aufbauen
  $\Longrightarrow$ Befehl nicht als Methode, sondern als Objekt modellieren

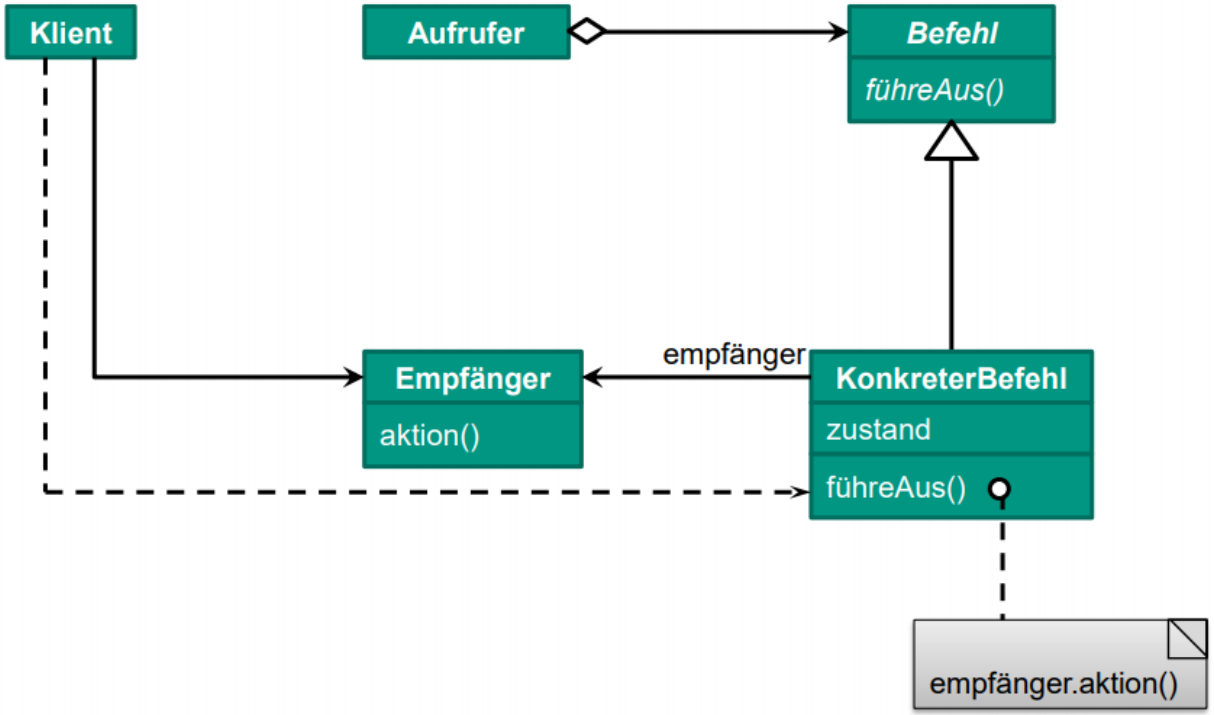


Befehl



Was haben wir erreicht?

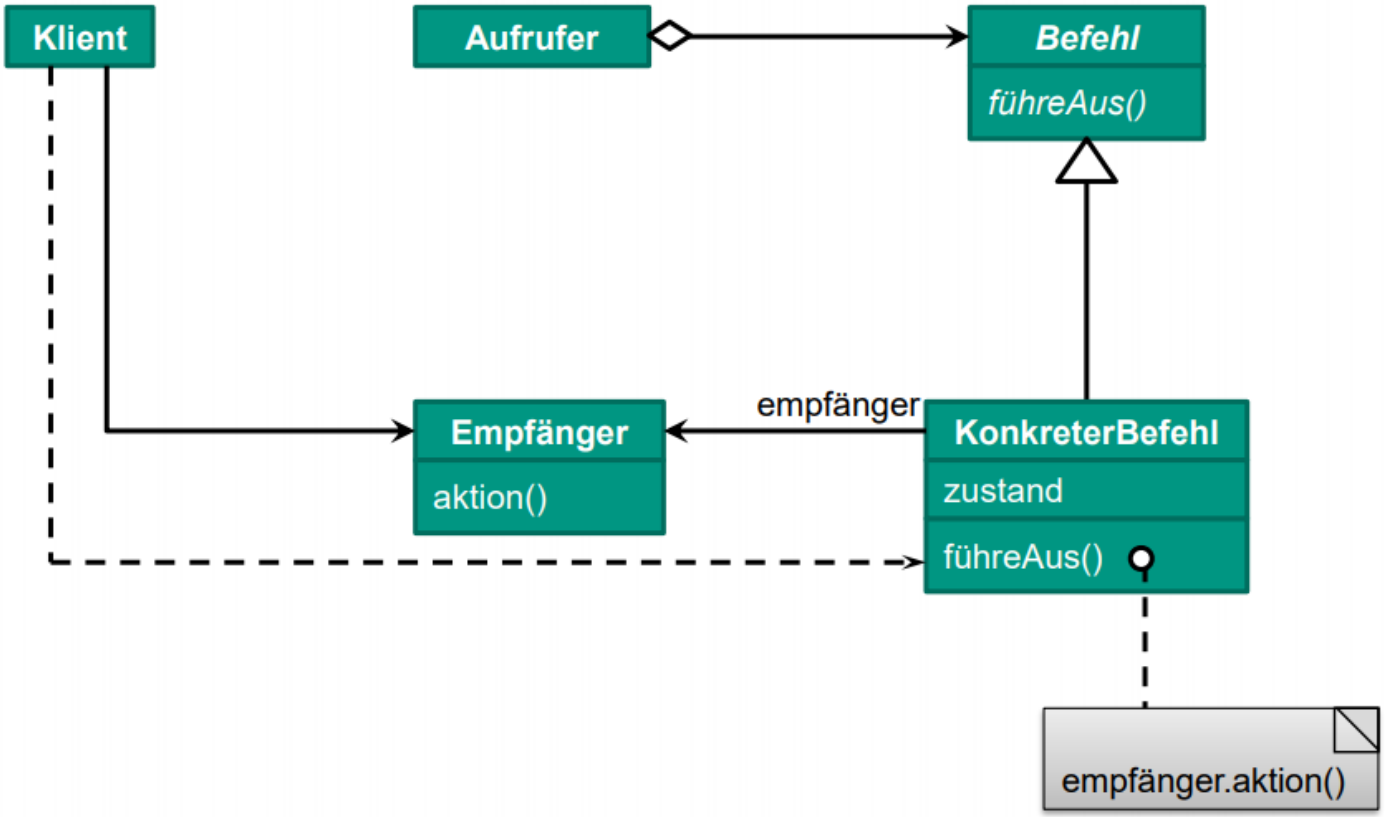
Befehl



Was haben wir erreicht?

- Austauschbarkeit: Befehle unabhängig vom Aufrufer, universell einsetzbar

Befehl



Beispiel!

Quiz (Ankreuzaufgaben aus Klausuren)

Wahr oder falsch?

- Bei dem Entwurfsmuster Befehl kennt der Empfänger den Befehl nicht, jedoch der Befehl den Empfänger.

Quiz (Ankreuzaufgaben aus Klausuren)

Wahr oder falsch?

- Bei dem Entwurfsmuster Befehl kennt der Empfänger den Befehl nicht, jedoch der Befehl den Empfänger. wahr

Quiz (Ankreuzaufgaben aus Klausuren)

Wahr oder falsch?

- Bei dem Entwurfsmuster Befehl kennt der Empfänger den Befehl nicht, jedoch der Befehl den Empfänger. wahr
- Ein Aufbewahrer im Entwurfsmuster Memento kann beliebig viele Mementos verwalten. Für die Restauration im Falle eines Reset ist er allerdings nicht verantwortlich.

Quiz (Ankreuzaufgaben aus Klausuren)

Wahr oder falsch?

- Bei dem Entwurfsmuster Befehl kennt der Empfänger den Befehl nicht, jedoch der Befehl den Empfänger. wahr
- Ein Aufbewahrer im Entwurfsmuster Memento kann beliebig viele Mementos verwalten. Für die Restauration im Falle eines Reset ist er allerdings nicht verantwortlich. wahr

Quiz (Ankreuzaufgaben aus Klausuren)

Wahr oder falsch?

- Bei dem Entwurfsmuster Befehl kennt der Empfänger den Befehl nicht, jedoch der Befehl den Empfänger. wahr
- Ein Aufbewahrer im Entwurfsmuster Memento kann beliebig viele Mementos verwalten. Für die Restauration im Falle eines Reset ist er allerdings nicht verantwortlich. wahr
- Die Fabrikmethode sorgt dafür, dass nur eine einzige Instanz einer Klasse fabriziert wird.

Quiz (Ankreuzaufgaben aus Klausuren)

Wahr oder falsch?

- Bei dem Entwurfsmuster Befehl kennt der Empfänger den Befehl nicht, jedoch der Befehl den Empfänger. wahr
- Ein Aufbewahrer im Entwurfsmuster Memento kann beliebig viele Mementos verwalten. Für die Restauration im Falle eines Reset ist er allerdings nicht verantwortlich. wahr
- Die Fabrikmethode sorgt dafür, dass nur eine einzige Instanz einer Klasse fabriziert wird. falsch

Quiz (Ankreuzaufgaben aus Klausuren)

Wahr oder falsch?

- Bei dem Entwurfsmuster Befehl kennt der Empfänger den Befehl nicht, jedoch der Befehl den Empfänger. wahr
- Ein Aufbewahrer im Entwurfsmuster Memento kann beliebig viele Mementos verwalten. Für die Restauration im Falle eines Reset ist er allerdings nicht verantwortlich. wahr
- Die Fabrikmethode sorgt dafür, dass nur eine einzige Instanz einer Klasse fabriziert wird. falsch
- Eine Schablonenmethode ist immer auch eine Fabrikmethode.

Quiz (Ankreuzaufgaben aus Klausuren)

Wahr oder falsch?

- Bei dem Entwurfsmuster Befehl kennt der Empfänger den Befehl nicht, jedoch der Befehl den Empfänger. wahr
- Ein Aufbewahrer im Entwurfsmuster Memento kann beliebig viele Mementos verwalten. Für die Restauration im Falle eines Reset ist er allerdings nicht verantwortlich. wahr
- Die Fabrikmethode sorgt dafür, dass nur eine einzige Instanz einer Klasse fabriziert wird. falsch
- Eine Schablonenmethode ist immer auch eine Fabrikmethode. falsch

Quiz (Ankreuzaufgaben aus Klausuren)

Wahr oder falsch?

- Bei dem Entwurfsmuster Befehl kennt der Empfänger den Befehl nicht, jedoch der Befehl den Empfänger. wahr
- Ein Aufbewahrer im Entwurfsmuster Memento kann beliebig viele Mementos verwalten. Für die Restauration im Falle eines Reset ist er allerdings nicht verantwortlich. wahr
- Die Fabrikmethode sorgt dafür, dass nur eine einzige Instanz einer Klasse fabriziert wird. falsch
- Eine Schablonenmethode ist immer auch eine Fabrikmethode. falsch
- Eine Komponente kann immer nur mit einem einzigen Dekorierer versehen werden.

Quiz (Ankreuzaufgaben aus Klausuren)

Wahr oder falsch?

- Bei dem Entwurfsmuster Befehl kennt der Empfänger den Befehl nicht, jedoch der Befehl den Empfänger. wahr
- Ein Aufbewahrer im Entwurfsmuster Memento kann beliebig viele Mementos verwalten. Für die Restauration im Falle eines Reset ist er allerdings nicht verantwortlich. wahr
- Die Fabrikmethode sorgt dafür, dass nur eine einzige Instanz einer Klasse fabriziert wird. falsch
- Eine Schablonenmethode ist immer auch eine Fabrikmethode. falsch
- Eine Komponente kann immer nur mit einem einzigen Dekorierer versehen werden. falsch

Für die Klausur

- Entwurfsmuster kommen sehr sehr sehr wahrscheinlich dran!

Für die Klausur

- Entwurfsmuster kommen sehr sehr sehr wahrscheinlich dran!
- Kategorien helfen beim Lernen

Für die Klausur

- Entwurfsmuster kommen sehr sehr sehr wahrscheinlich dran!
- Kategorien helfen beim Lernen
- jedes Entwurfsmuster erfüllt einen bestimmten Zweck
  $\Longrightarrow$ nicht nur die Klassen und Methoden auswendig lernen, sondern das Prinzip verstehen

Für die Klausur

- Entwurfsmuster kommen sehr sehr sehr wahrscheinlich dran!
- Kategorien helfen beim Lernen
- jedes Entwurfsmuster erfüllt einen bestimmten Zweck
  $\Longrightarrow$ nicht nur die Klassen und Methoden auswendig lernen, sondern das Prinzip verstehen
- bei Unklarheiten in Head First Design Patterns nachlesen ;)

Tipps - 5. Übungsblatt

Aufgabe 1: Shutterpile: Refaktorisierung + Entwurfsmuster anwenden

- Entwurfsmuster anschauen
- alte Tests verwenden + evtl. neue schreiben

Tipps - 5. Übungsblatt

Aufgabe 1: Shutterpile: Refaktorisierung + Entwurfsmuster anwenden

- Entwurfsmuster anschauen
- alte Tests verwenden + evtl. neue schreiben

Aufgabe 2: cmd-Programm für Pipeline

- wie Shutterpile-cmd, nur kommen nach Parameter „-p“ noch Werte
- `https://commons.apache.org/proper/commons-cli/usage.html`

Tipps - 5. Übungsblatt

Aufgabe 3: Wo sind Entwurfsmuster in Shutterpile?

- Maßstab ist Musterlösung
- nur finden reicht nicht, auch erklären wie und warum

Tipps - 5. Übungsblatt

Aufgabe 3: Wo sind Entwurfsmuster in Shutterpile?

- Maßstab ist Musterlösung
- nur finden reicht nicht, auch erklären wie und warum

Aufgabe 4: Entwurfsmuster in Java-API

- es handelt sich um „einfachere“ Muster

Tipps - 5. Übungsblatt

Aufgabe 3: Wo sind Entwurfsmuster in Shutterpile?

- Maßstab ist Musterlösung
- nur finden reicht nicht, auch erklären wie und warum

Aufgabe 4: Entwurfsmuster in Java-API

- es handelt sich um „einfachere“ Muster

Aufgabe 5: Entwurfsmuster - Kaffeemaschine

- ein Muster anwenden

Denkt dran!

Abgabe

- Deadline am 27.6. um 12:00
- Aufgabe 3-5 handschriftlich

Bis dann! (dann := 03.07.18)