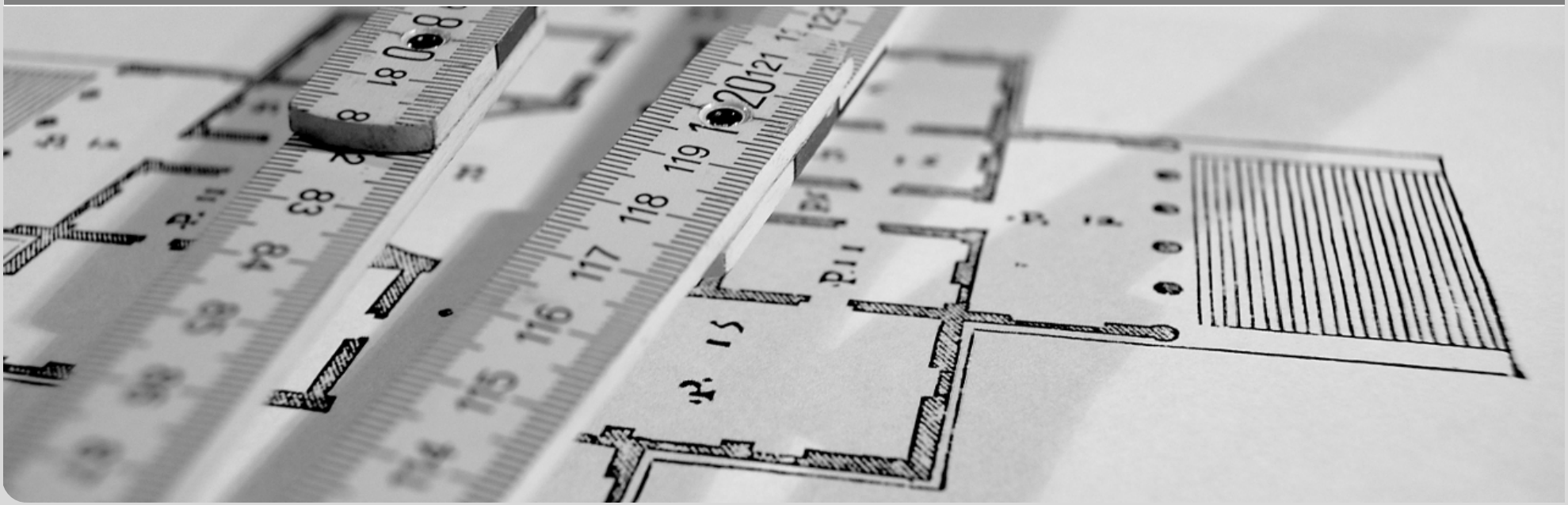
Karlsruher Institut für Technologie

Softwaretechnik 1 - 4. Tutorium

Tutorium 17

Felix Bachmann | 25.06.2019

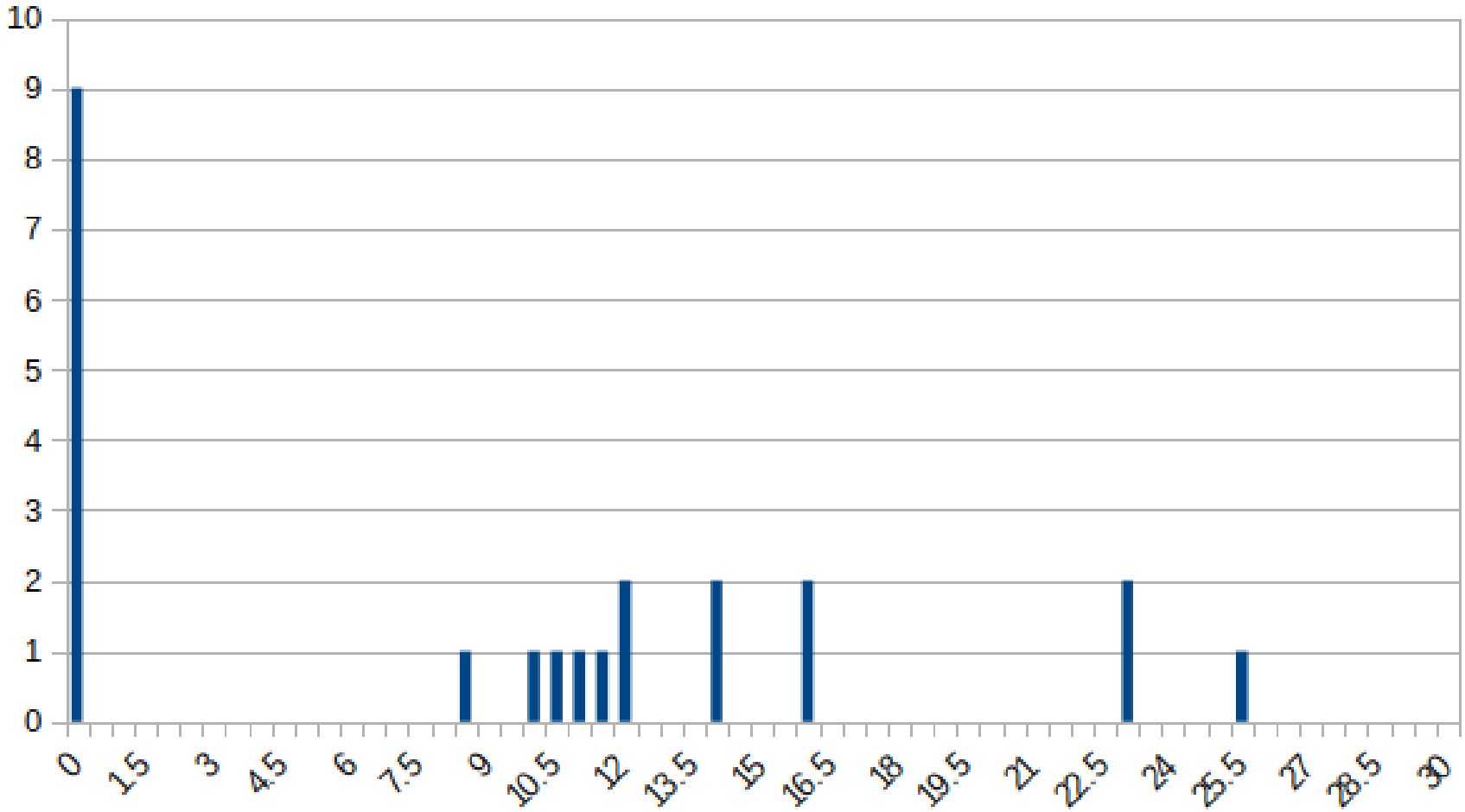
KIT - INSTITUT FÜR PROGRAMMSTRUKTUREN UND DATENORGANISATION (IPD)



Evaluation der Evaluation

Gut	Schlecht/Verbesserungswürdig

3. Übungsblatt Statistik



Häufige Fehler: Blatt 3

Programmieraufgaben generell

- wie letztes Mal gesagt

Häufige Fehler: Blatt 3

Programmieraufgaben generell

- wie letztes Mal gesagt
- und das Mal davor

Häufige Fehler: Blatt 3

Programmieraufgaben generell

- wie letztes Mal gesagt
- und das Mal davor
- und das Mal davor

Häufige Fehler: Blatt 3

Programmieraufgaben generell

- wie letztes Mal gesagt
- und das Mal davor
- und das Mal davor
- und das Mal davor

Häufige Fehler: Blatt 3

Programmieraufgaben generell

- wie letztes Mal gesagt
- und das Mal davor
- und das Mal davor
- und das Mal davor
- CheckStyle und Co.

Häufige Fehler: Blatt 3

Programmieraufgaben generell

- wie letztes Mal gesagt
- und das Mal davor
- und das Mal davor
- und das Mal davor
- CheckStyle und Co.
- nicht am JMJRST-Stil orientieren
  - das können wir besser :)

Aufgabe 1 (`PluginManagement` + `PluginForJMJRST`)

Häufige Fehler: Blatt 3

Programmieraufgaben generell

- wie letztes Mal gesagt
- und das Mal davor
- und das Mal davor
- und das Mal davor
- CheckStyle und Co.
- nicht am JMJRST-Stil orientieren
  - das können wir besser :)

Aufgabe 1 (`PluginManagement` + `PluginForJMJRST`)

- Plugins anhand **Klassen**namen vergleichen,
  - nicht `Object::getName()`
  - nicht `PluginForJMJRST::getName()`
  - z.B. `Object::getSimpleName()`

Häufige Fehler: Blatt 3

Aufgabe 2 (Instagrim Plug-In)

Häufige Fehler: Blatt 3

Aufgabe 2 (Instagrim Plug-In)

- Kommentarmenge komisch umgesetzt
  - am einfachsten: String-Array
  - dann aber keine magic numbers verwenden
    - z.B. beim Ziehen der zufälligen Kommentare

Häufige Fehler: Blatt 3

Aufgabe 2 (Instagrim Plug-In)

- Kommentarmenge komisch umgesetzt
  - am einfachsten: String-Array
  - dann aber keine magic numbers verwenden
    - z.B. beim Ziehen der zufälligen Kommentare
- Frame für `configure()`-Aufruf selbst gebaut
  - kein Abzug solange kein Quatsch passiert und sinnvoll programmiert
  - geht aber auch deutlich einfacher (siehe MuLö)

Häufige Fehler: Blatt 3

Aufgabe 2 (Instagrim Plug-In)

- Kommentarmenge komisch umgesetzt
  - am einfachsten: String-Array
  - dann aber keine magic numbers verwenden
    - z.B. beim Ziehen der zufälligen Kommentare
- Frame für `configure()`-Aufruf selbst gebaut
  - kein Abzug solange kein Quatsch passiert und sinnvoll programmiert
  - geht aber auch deutlich einfacher (siehe MuLö)

Aufgabe 3 (iMage-Bundle)

- keine :D

Häufige Fehler: Blatt 3

Aufgabe 4 (Aktivitätsdiagramm)

Häufige Fehler: Blatt 3

Aufgabe 4 (Aktivitätsdiagramm)

- wie schon bei Anwendungsfalldiagramm
  - Aktivitäten enthalten Verben
    - es geht darum wer etwas tut
    - „Startseite“ ist keine Aktivität
    - „Startseite anzeigen“ schon

Häufige Fehler: Blatt 3

Aufgabe 4 (Aktivitätsdiagramm)

- wie schon bei Anwendungsfalldiagramm
  - Aktivitäten enthalten Verben
    - es geht darum wer etwas tut
    - „Startseite“ ist keine Aktivität
    - „Startseite anzeigen“ schon
- keine Partition verwendet/ falsche Syntax

Häufige Fehler: Blatt 3

Aufgabe 4 (Aktivitätsdiagramm)

- wie schon bei Anwendungsfalldiagramm
  - Aktivitäten enthalten Verben
    - es geht darum wer etwas tut
    - „Startseite“ ist keine Aktivität
    - „Startseite anzeigen“ schon
- keine Partition verwendet/ falsche Syntax
- Aktivitiäten = runde Ecken, Objekte = spitze Ecken

Häufige Fehler: Blatt 3

Aufgabe 4 (Aktivitätsdiagramm)

- wie schon bei Anwendungsfalldiagramm
  - Aktivitäten enthalten Verben
    - es geht darum wer etwas tut
    - „Startseite“ ist keine Aktivität
    - „Startseite anzeigen“ schon
- keine Partition verwendet/ falsche Syntax
- Aktivitiäten = runde Ecken, Objekte = spitze Ecken
- [`Bedingung`]

Häufige Fehler: Blatt 3

Aufgabe 4 (Aktivitätsdiagramm)

- wie schon bei Anwendungsfalldiagramm
  - Aktivitäten enthalten Verben
    - es geht darum wer etwas tut
    - „Startseite“ ist keine Aktivität
    - „Startseite anzeigen“ schon
- keine Partition verwendet/ falsche Syntax
- Aktivitiäten = runde Ecken, Objekte = spitze Ecken
- `[Bedingung]`
- Modellierung der Nutzerinteraktion
  - z.B. „Klick“ nachdem „Startseite anzeigen“ vorbei? nicht sinnvoll

Häufige Fehler: Blatt 3

Aufgabe 5 (Zustandsdiagramm)

Häufige Fehler: Blatt 3

Aufgabe 5 (Zustandsdiagramm)

- Übergänge von innerem parallelen Zustand `VxW` nach außen (`A`, `I`)
  - `VxW` ist auch nur ein Zustand
  - auch wenn „innen drin“ auch noch was passiert
  - war in MuLö leider auch erst falsch
    - deswegen bei einigen Korrekturen evtl. etwas Geschmiere an der Stelle, sorry!

Häufige Fehler: Blatt 3

Aufgabe 6 (Sequenzdiagramm)

- allerlei Syntax-Kram
  - Lebenslinien, Kästen, Doppelpunkte, Rückgabe-Notation
  - asynchron vs. synchron (Pfeilspitzen wichtig)
  - kein Doppelpunkt bei „statischen Kästen“
  - `create` zeigt auf Kasten, nicht Lebenslinie/Steuerungsfokus

Häufige Fehler: Blatt 3

Aufgabe 6 (Sequenzdiagramm)

- allerlei Syntax-Kram
  - Lebenslinien, Kästen, Doppelpunkte, Rückgabe-Notation
  - asynchron vs. synchron (Pfeilspitzen wichtig)
  - kein Doppelpunkt bei „statischen Kästen“
  - `create` zeigt auf Kasten, nicht Lebenslinie/Steuerungsfokus
- Achtung: Steuerungsfokus ist zwar optional, erhöht aber stark die Lesbarkeit. In Klausur bitte verwenden. Bei Selbstaufrufen sonst problematisch, da Rückgabe ebenfalls optional.

Häufige Fehler: Blatt 3

Aufgabe 6 (Sequenzdiagramm)

- allerlei Syntax-Kram
  - Lebenslinien, Kästen, Doppelpunkte, Rückgabe-Notation
  - asynchron vs. synchron (Pfeilspitzen wichtig)
  - kein Doppelpunkt bei „statischen Kästen“
  - `create` zeigt auf Kasten, nicht Lebenslinie/Steuerungsfokus
- Achtung: Steuerungsfokus ist zwar optional, erhöht aber stark die Lesbarkeit. In Klausur bitte verwenden. Bei Selbstaufrufen sonst problematisch, da Rückgabe ebenfalls optional.
- Vorsicht bei Objekt-Zerstörung
  - `CameraCurve` vor Verwendeung zerstört

Häufige Fehler: Blatt 3

Aufgabe 7 (Substitutionsprinzip: Logger)

Häufige Fehler: Blatt 3

Aufgabe 7 (Substitutionsprinzip: Logger)

- Varianz war kein Problem
  - getter: Kovarianz bei Rückgabetyp passt
  - setter: Invarianter Parametertyp passt

Häufige Fehler: Blatt 3

Aufgabe 7 (Substitutionsprinzip: Logger)

- Varianz war kein Problem
  - getter: Kovarianz bei Rückgabetyp passt
  - setter: Invarianter Parametertyp passt
- Problem war Verhalten, schwächere Nachbedingung

Häufige Fehler: Blatt 3

Aufgabe 7 (Substitutionsprinzip: Logger)

- Varianz war kein Problem
  - getter: Kovarianz bei Rückgabetyp passt
  - setter: Invarianter Parametertyp passt
- Problem war Verhalten, schwächere Nachbedingung
- wurde oft zu schwammig formuliert
  - oft sowas wie „Unterklasse tut nicht mehr dasselbe wie Oberklasse“
  - aber das soll sie ja auch nicht, dann bräuchten wir kein Überschreiben von Methoden :)
- bei solchen Aufgaben immer aus Klient-Sicht anschauen

Häufige Fehler: Blatt 3

```
O o = new O();
o.m(x);

O o = new U();
o.m(x);
```

- zweiter Aufruf „muss sich so verhalten wie“ erster Aufruf
- Vorbedingung schwächer/gleich
- Nachbedingung stärker/gleich

Häufige Fehler: Blatt 3

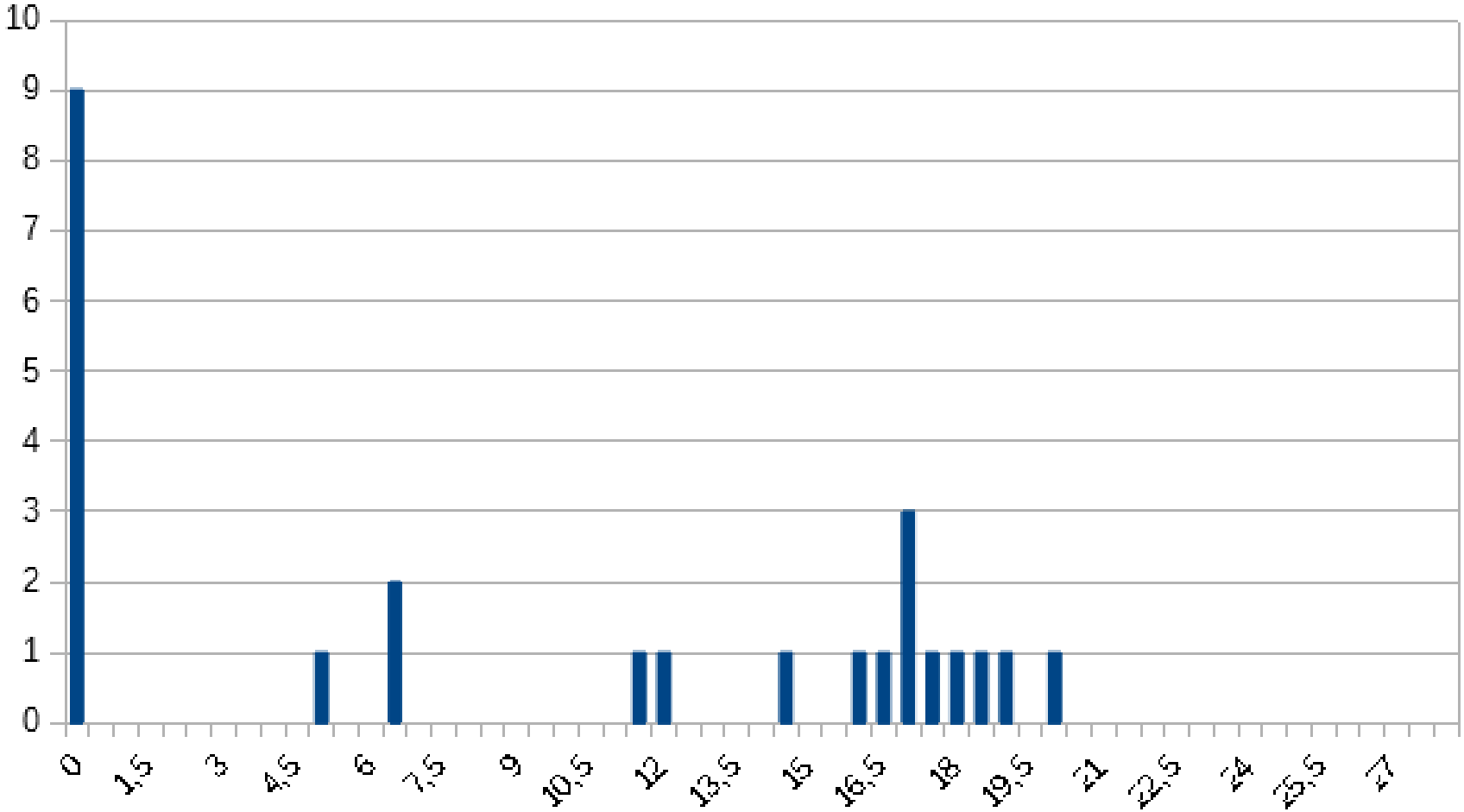
```
O o = new O();
o.m(x);

O o = new U();
o.m(x);
```

- zweiter Aufruf „muss sich so verhalten wie“ erster Aufruf
- Vorbedingung schwächer/gleich
- Nachbedingung stärker/gleich

- „Unterklasse darf weniger verlangen, muss aber mehr leisten“
- (oder gleich viel)
- Hintergrund: Klient muss Unterklasse so benutzen können wie Oberklasse. Würde die Unterklasse bspw. mehr verlangen als Oberklasse müsste Klient wissen, dass er Unterklasse benutzt, um sie korrekt aufrufen zu können.

4. Übungsblatt Statistik



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

```
// 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

```
// 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

Was bisher geschah..

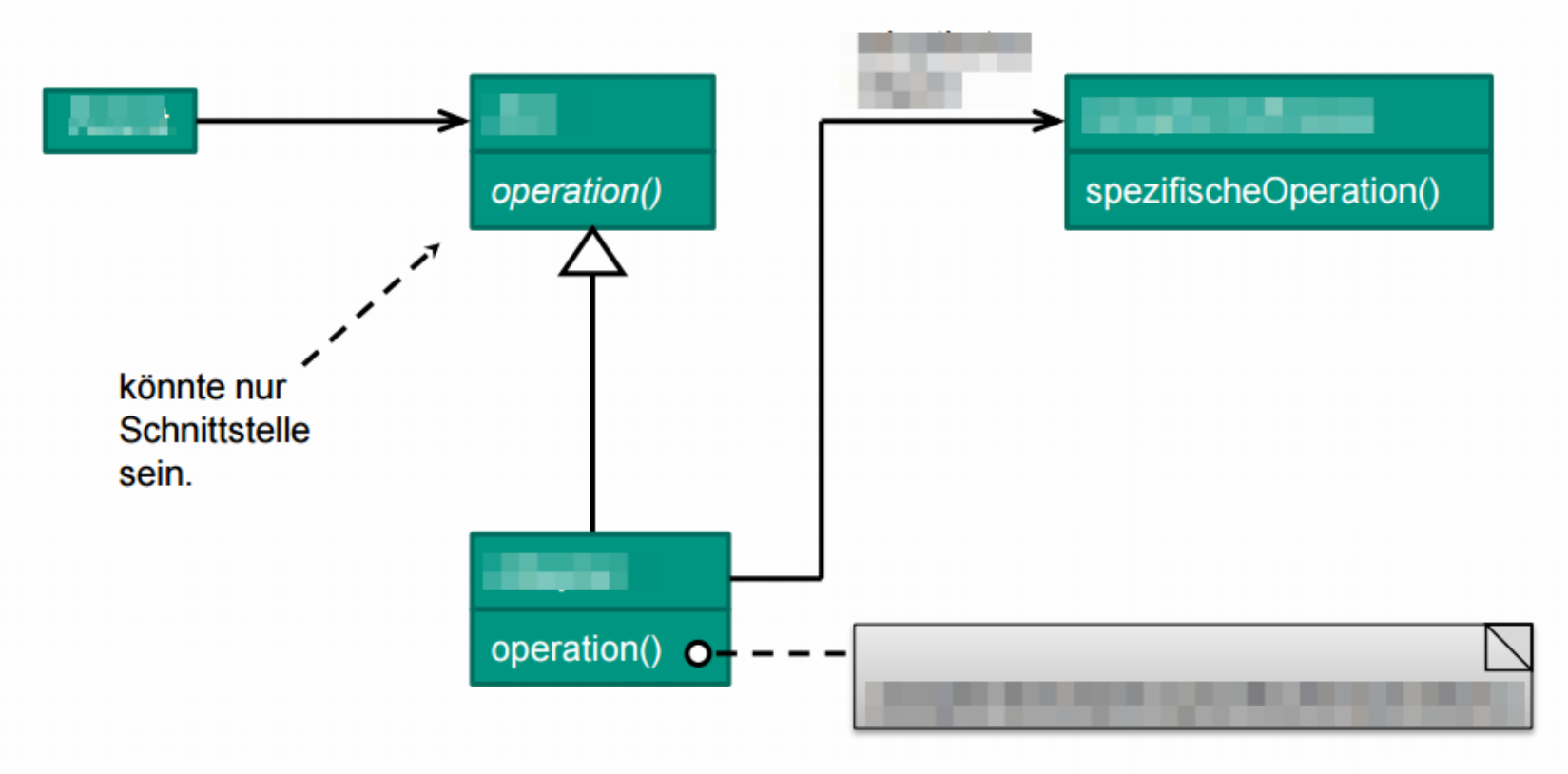
- haben uns Entkopplungmuster angeschaut

Was bisher geschah..

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

Was bisher geschah..

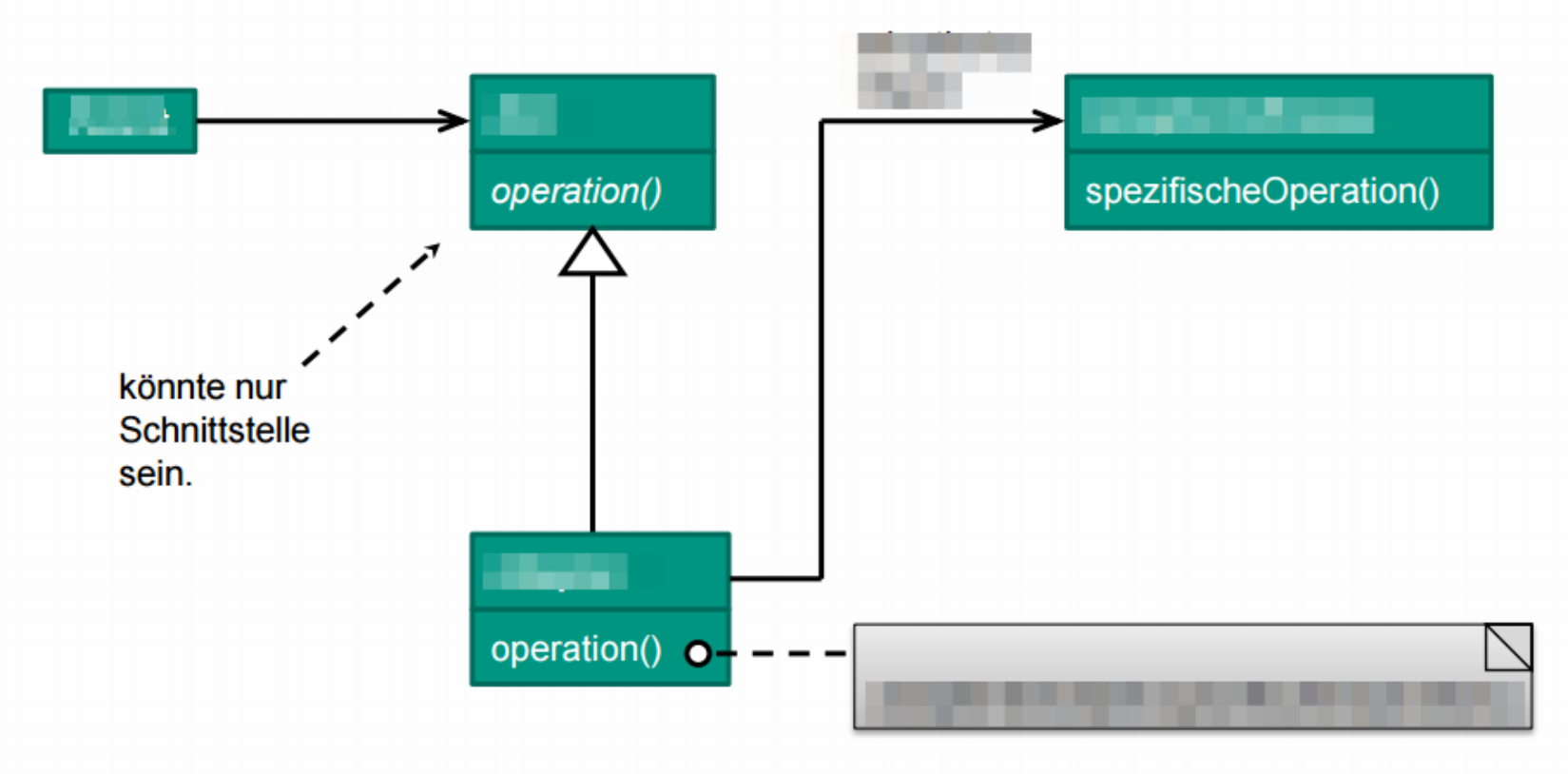
- haben uns Entkopplungmuster angeschaut
  $\Longrightarrow$ Beobachter, Iterator, Adapter, Stellvertreter, Vermittler



Welches Entwurfsmuster?

Was bisher geschah..

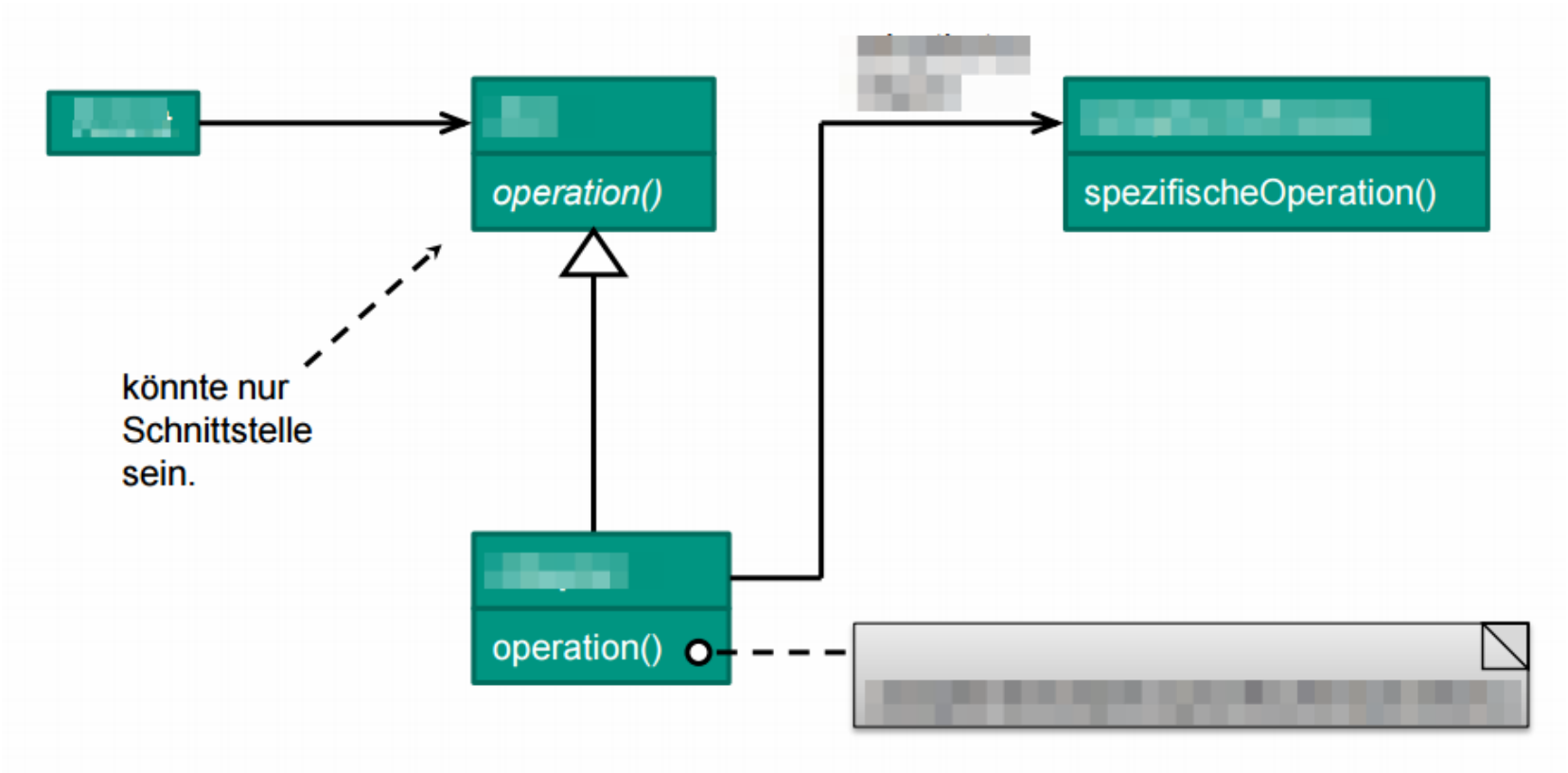
- haben uns Entkopplungmuster angeschaut
  $\Longrightarrow$ Beobachter, Iterator, Adapter, Stellvertreter, Vermittler



Welches Entwurfsmuster? (Objekt-)Adapter

Was bisher geschah..

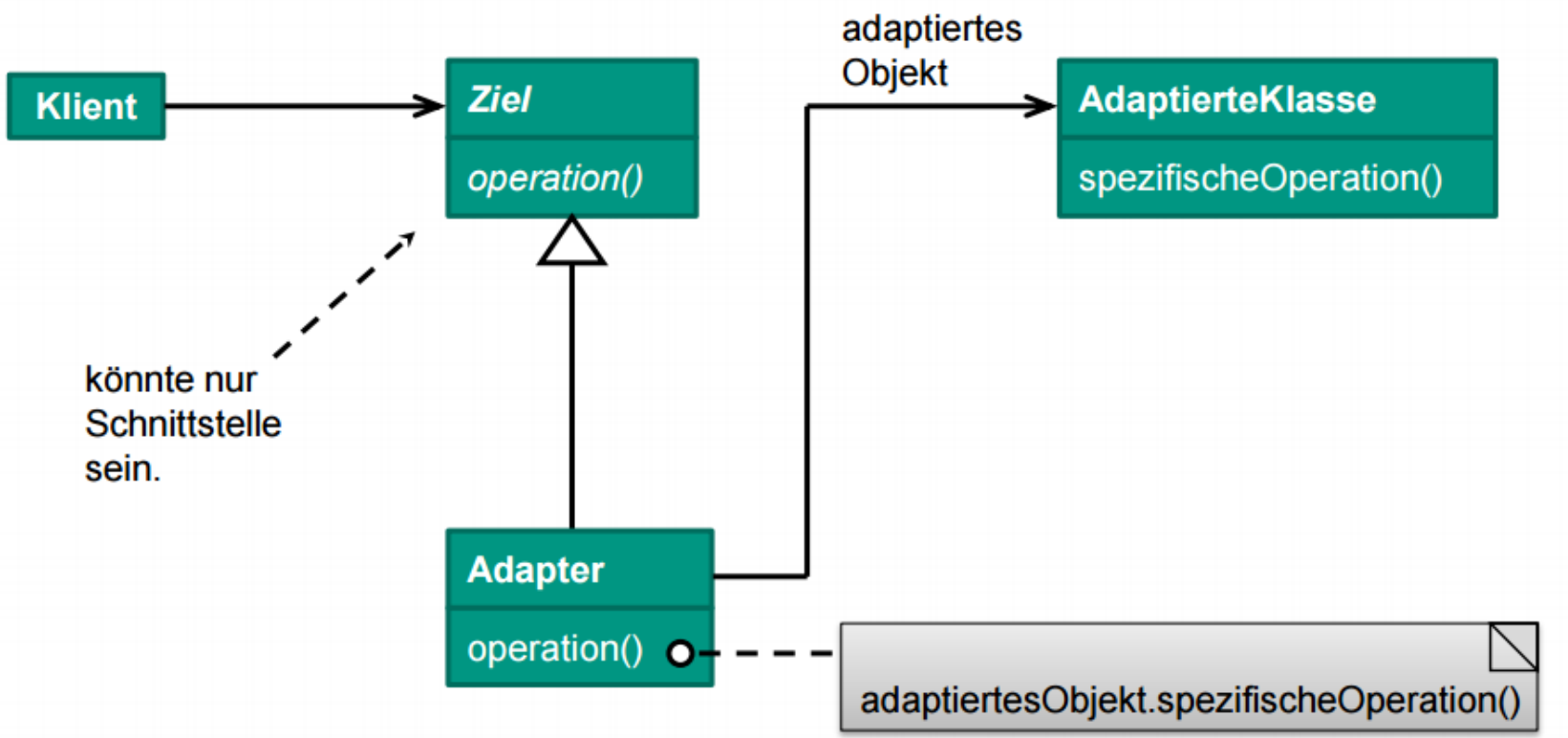
- haben uns Entkopplungmuster angeschaut
  $\Longrightarrow$ Beobachter, Iterator, Adapter, Stellvertreter, Vermittler



Welche Klassen?

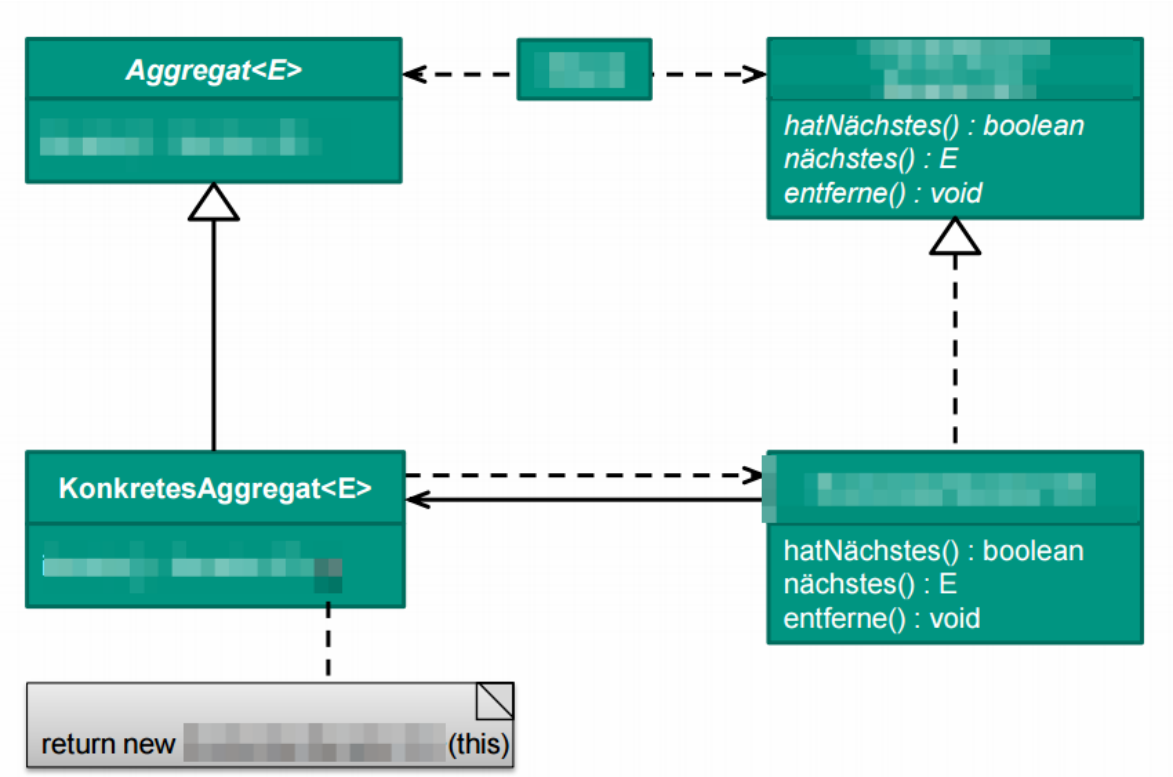
Was bisher geschah..

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



Was bisher geschah..

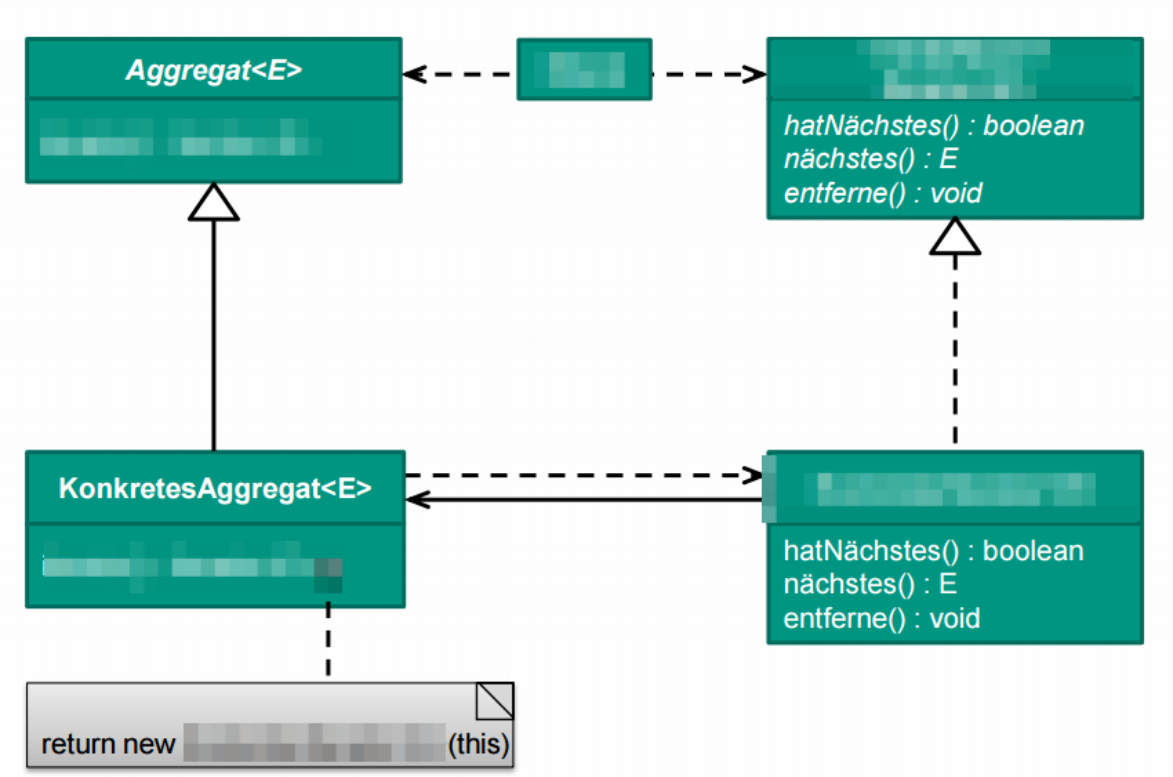
- haben uns Entkopplungmuster angeschaut
  $\Longrightarrow$ Beobachter, Iterator, Adapter, Stellvertreter, Vermittler



Welches Entwurfsmuster?

Was bisher geschah..

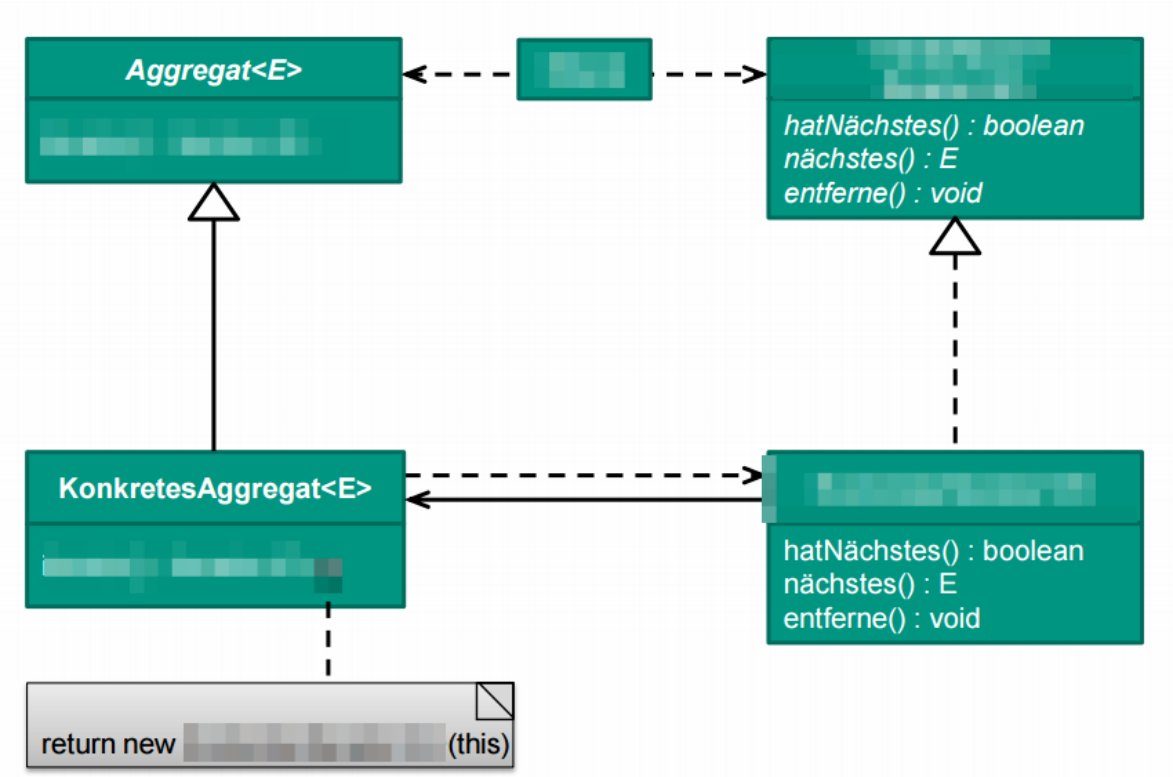
- haben uns Entkopplungmuster angeschaut
  $\Longrightarrow$ Beobachter, Iterator, Adapter, Stellvertreter, Vermittler



Welches Entwurfsmuster? Iterator

Was bisher geschah..

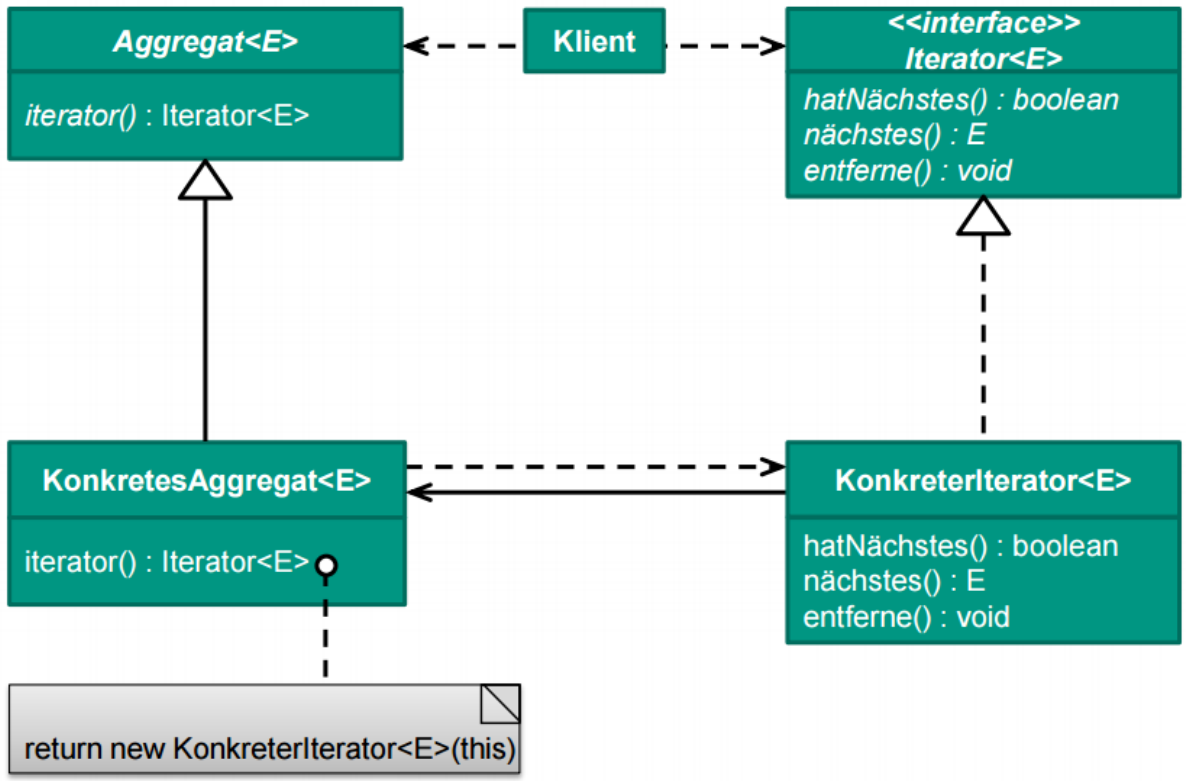
- haben uns Entkopplungmuster angeschaut
  $\Longrightarrow$ Beobachter, Iterator, Adapter, Stellvertreter, Vermittler



Welche Klassen und Methoden?

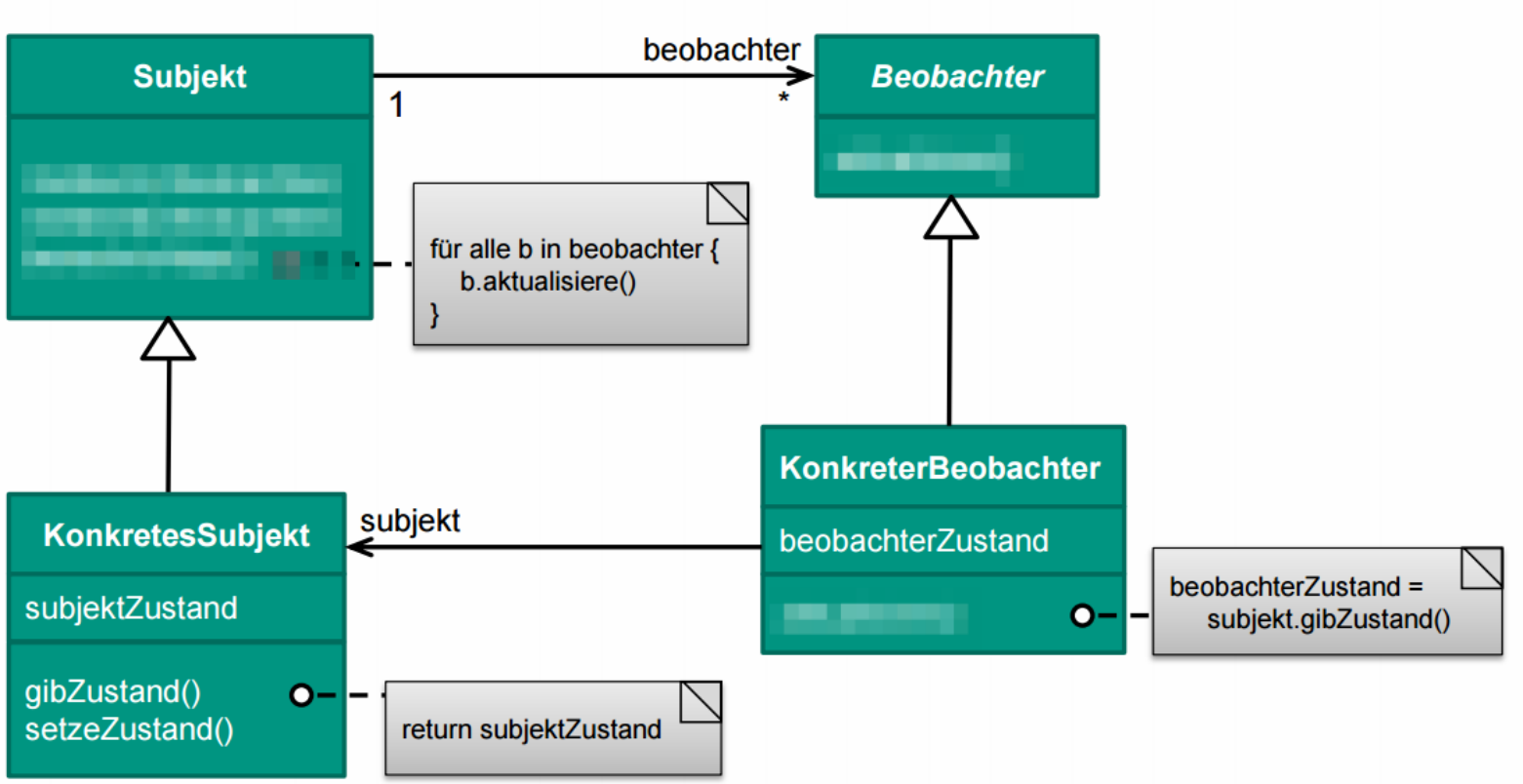
Was bisher geschah..

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



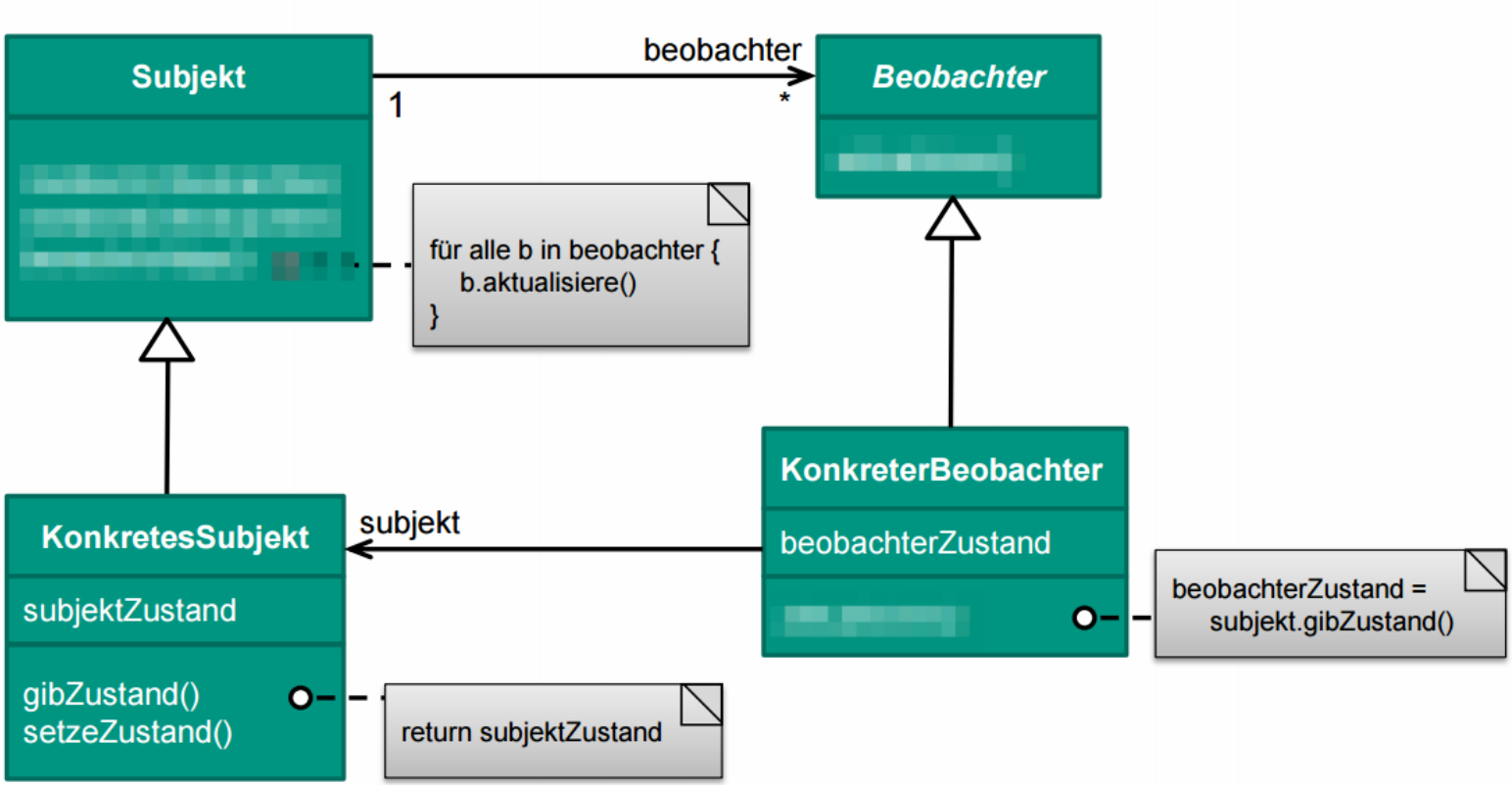
Was bisher geschah..

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



Was bisher geschah..

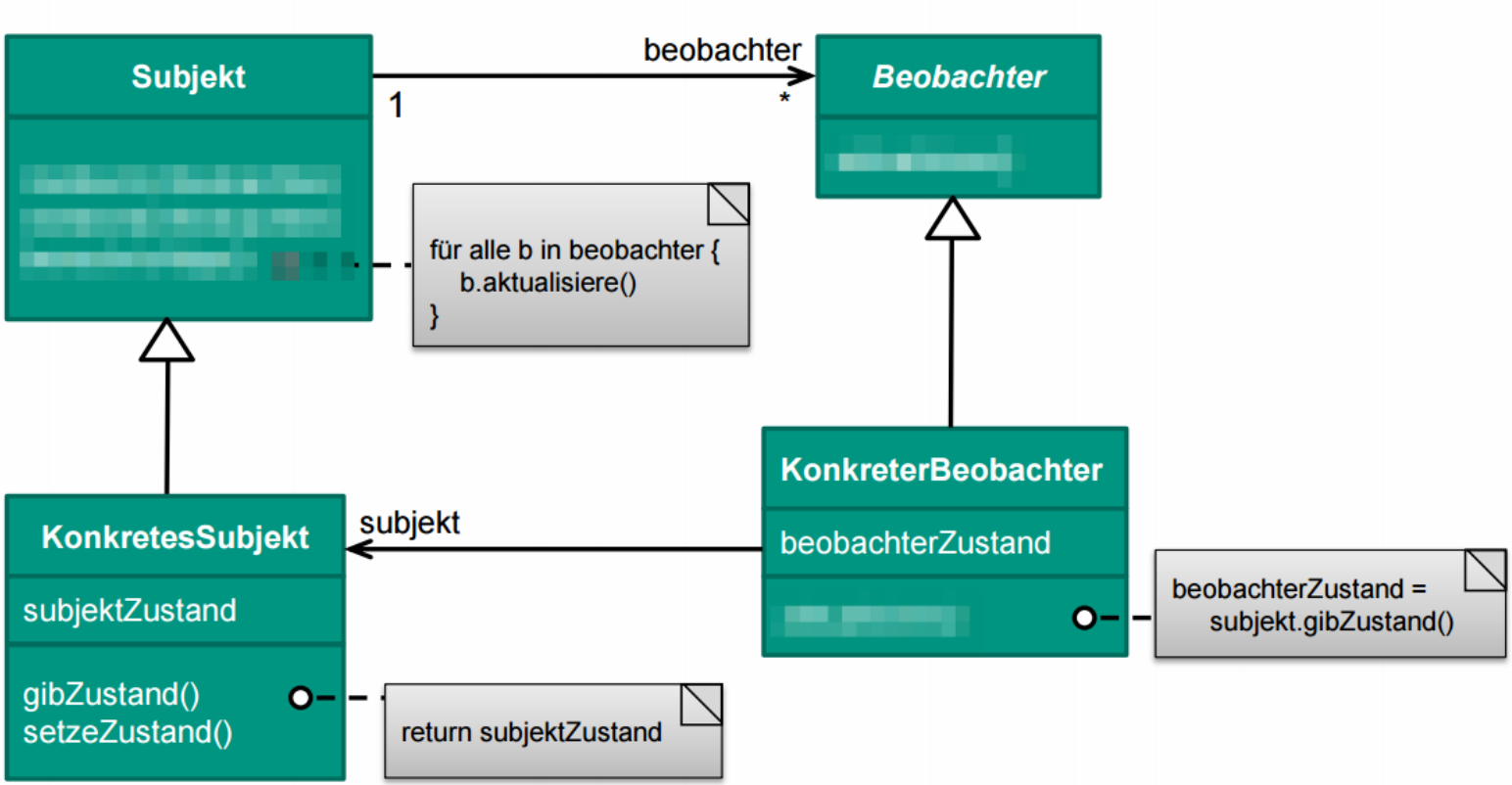
- haben uns Entkopplungmuster angeschaut
  $\Longrightarrow$ Beobachter, Iterator, Adapter, Stellvertreter, Vermittler



Ist wohl ein Beobachter :)

Was bisher geschah..

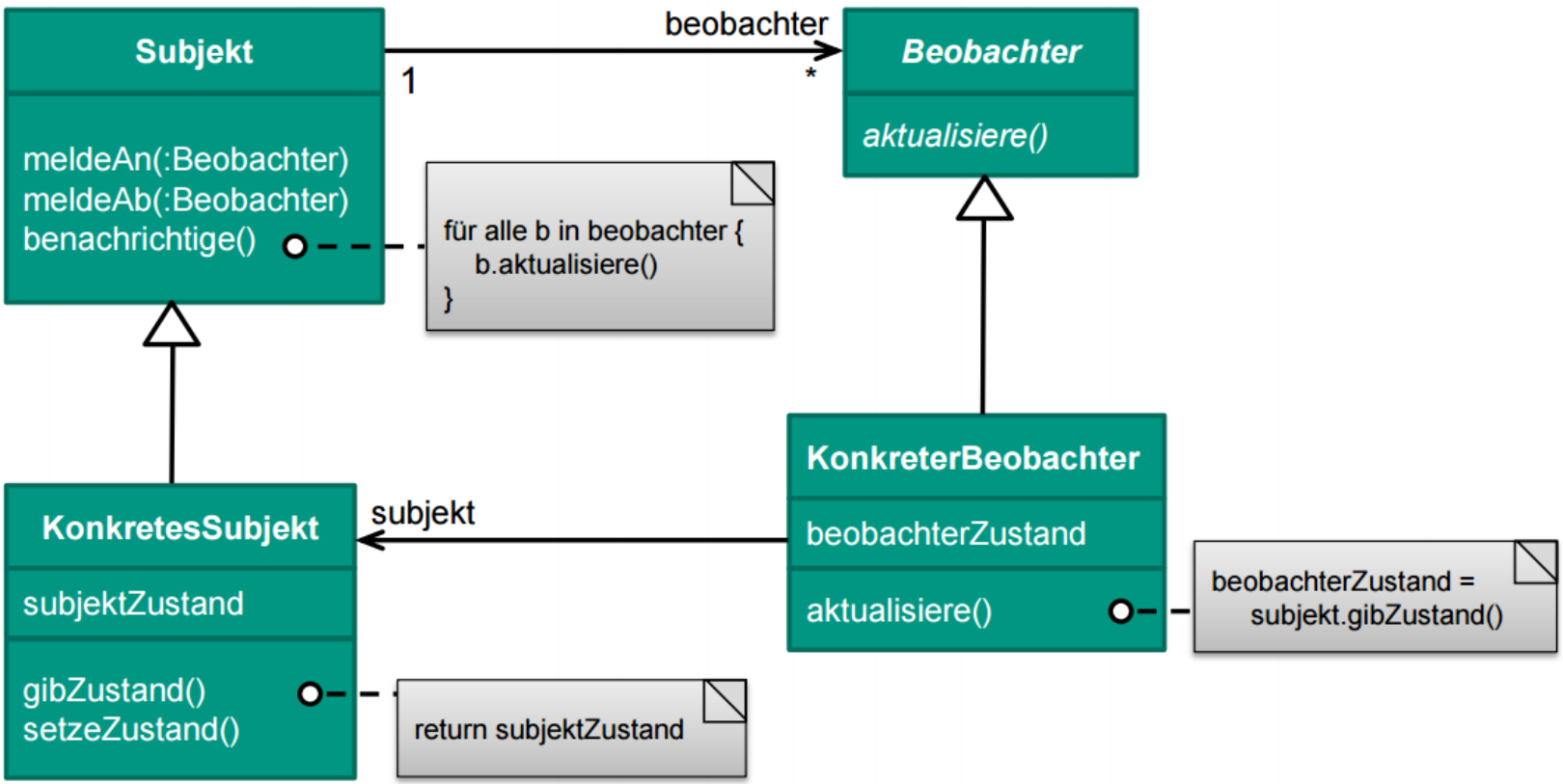
- haben uns Entkopplungmuster angeschaut
  $\Longrightarrow$ Beobachter, Iterator, Adapter, Stellvertreter, Vermittler



Ist wohl ein Beobachter :) Methoden?

Was bisher geschah..

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



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

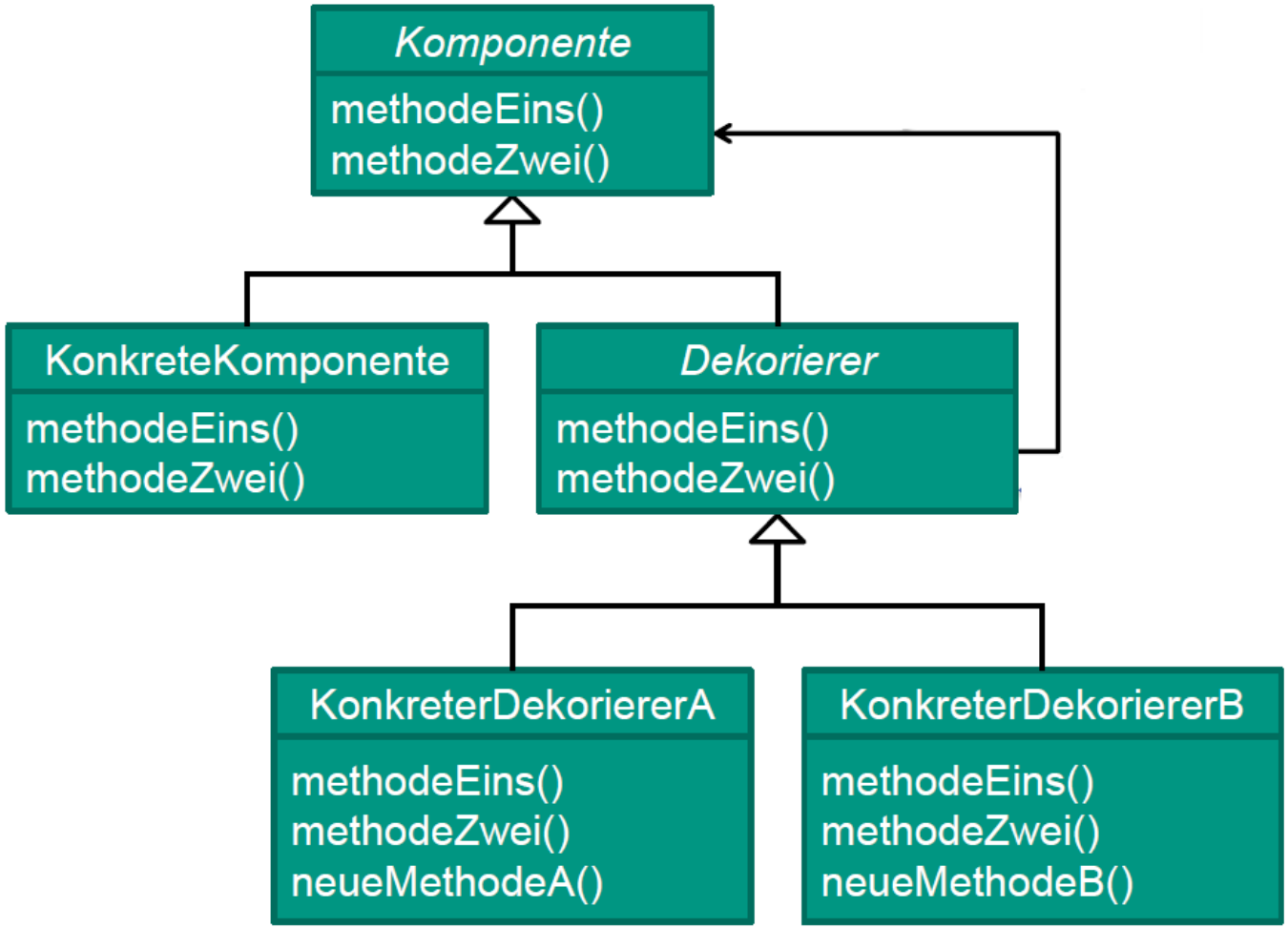
Übergeordnetes Ziel

- 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().

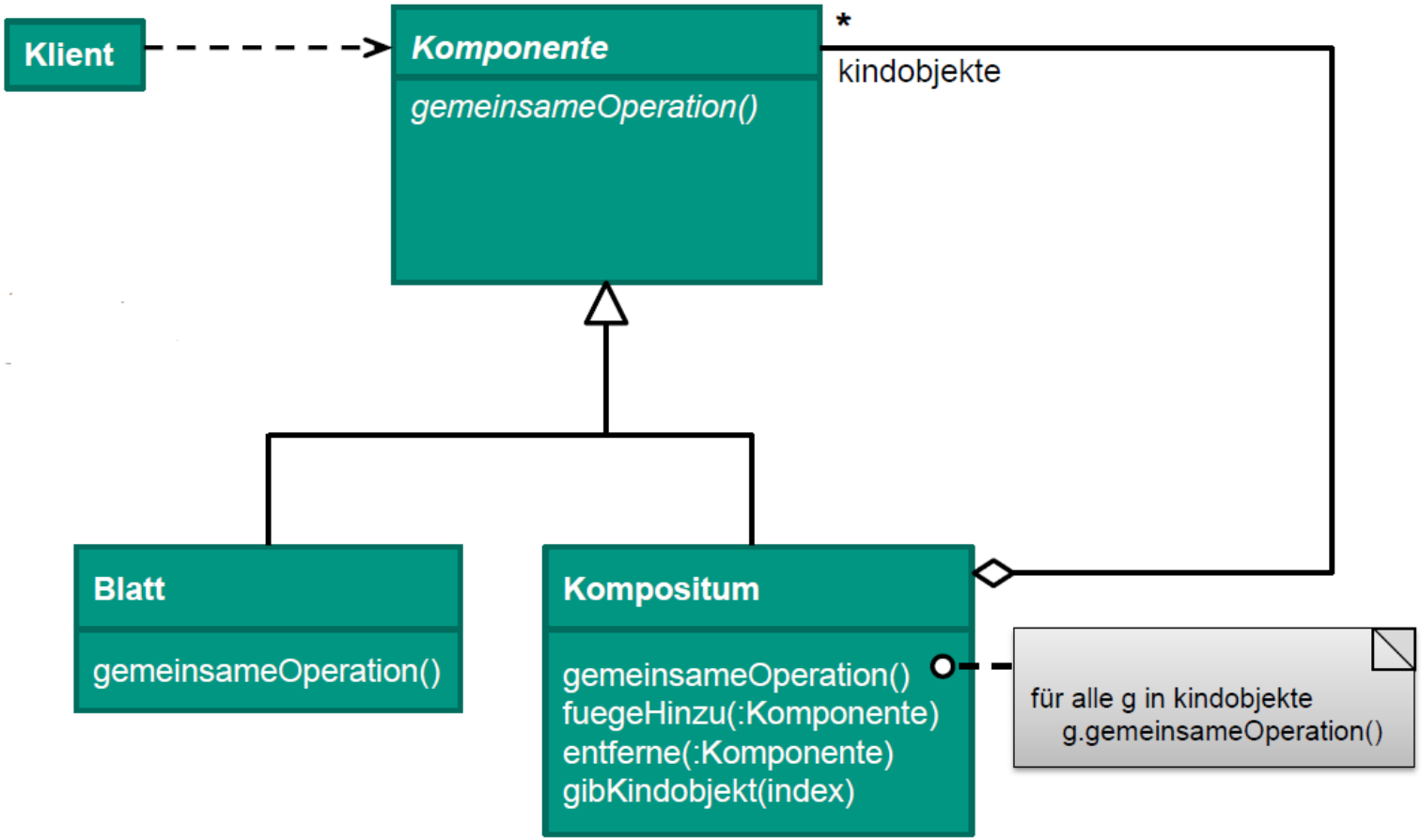
Wo Variation?

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().

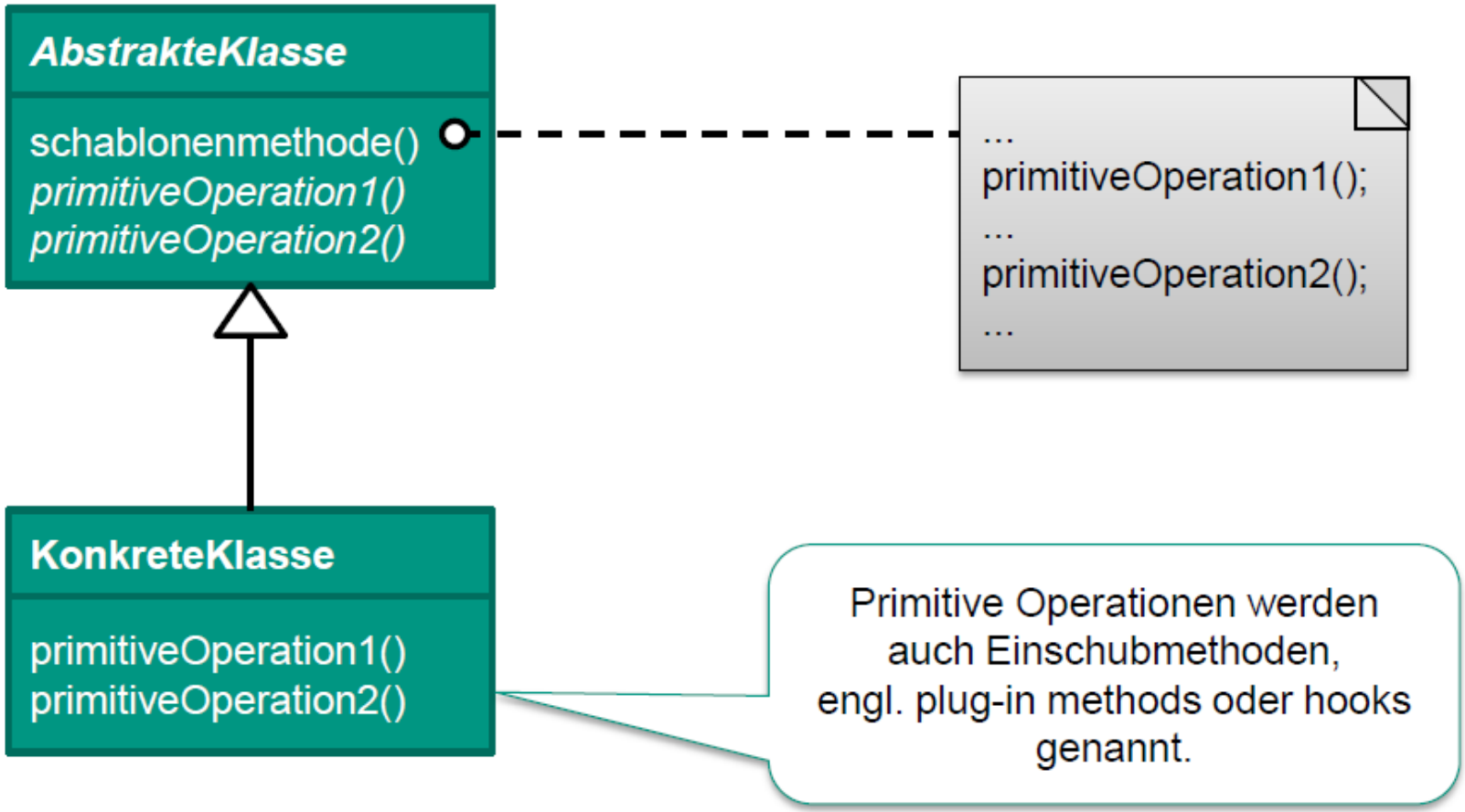
Wo Variation?

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.

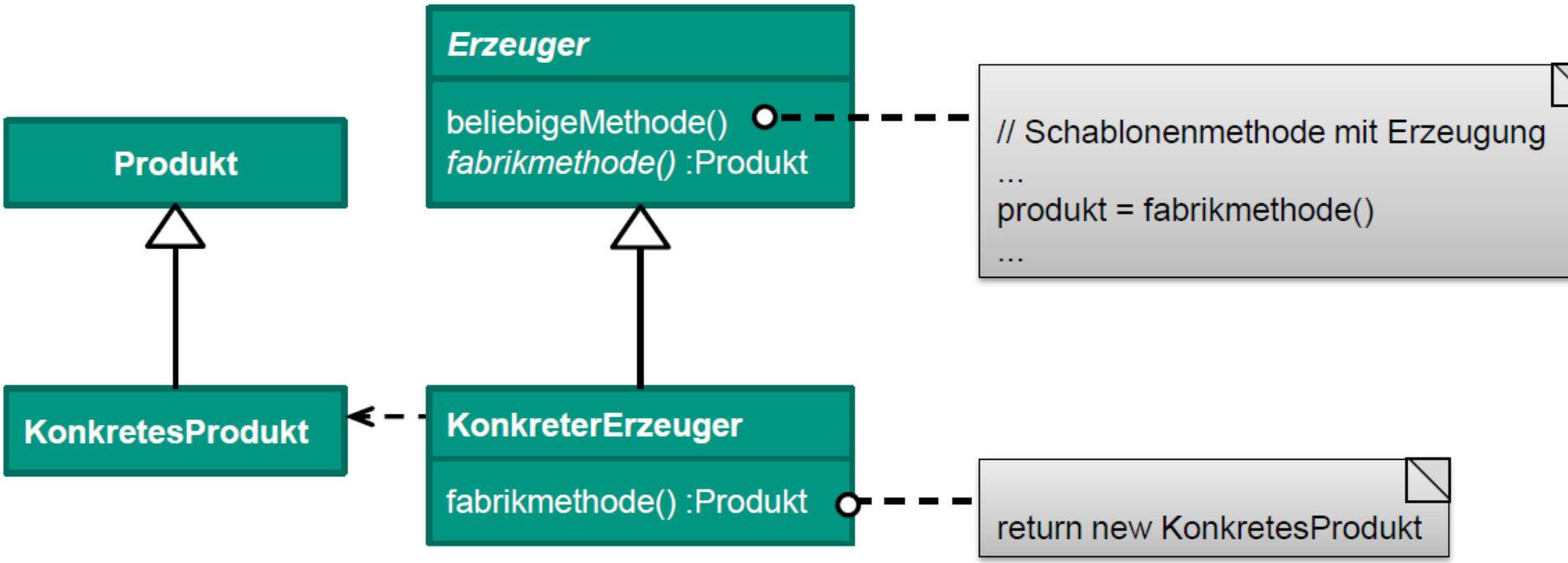
Wo Variation?

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

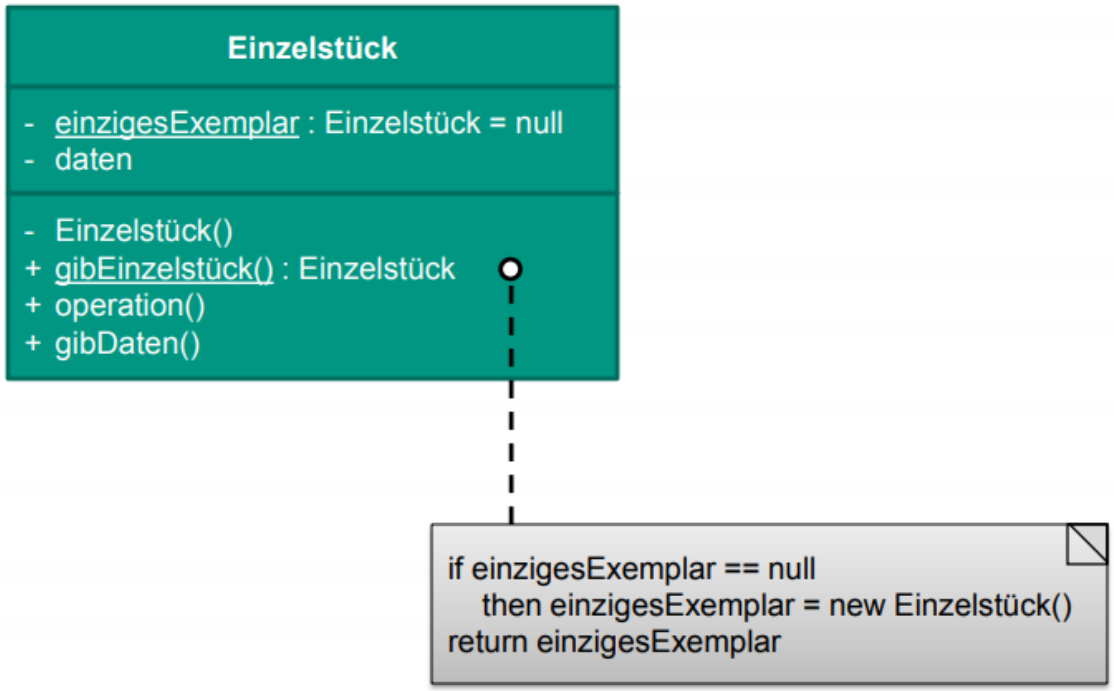
Problem

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

Einzelstück/Singleton

Problem

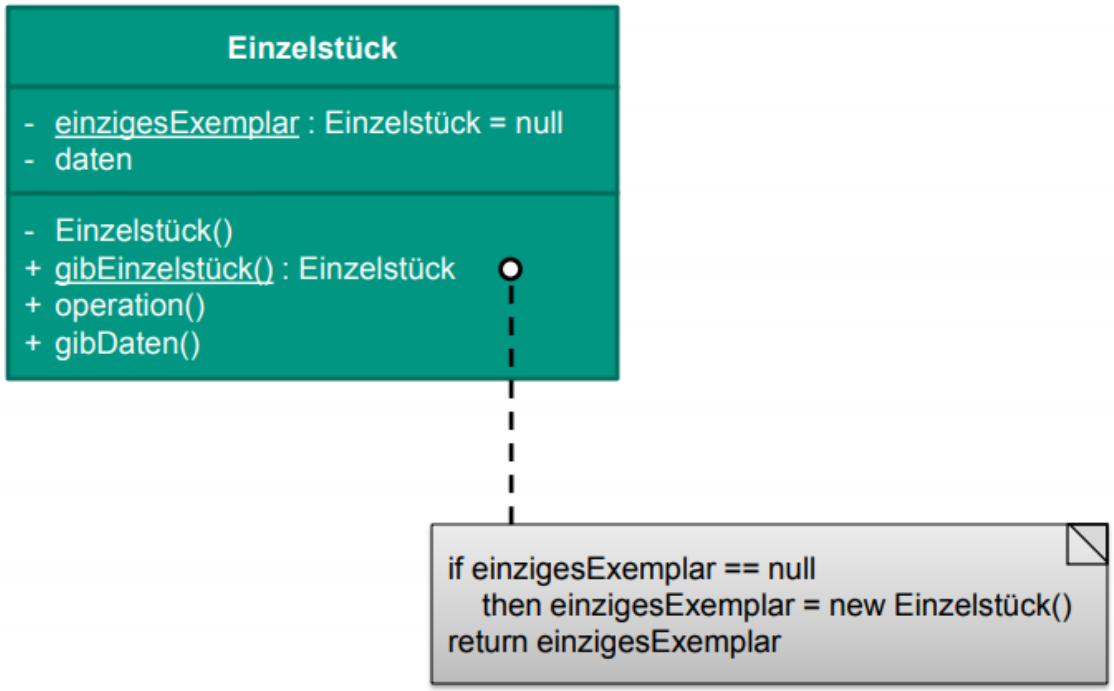
- von einer Klasse soll nur eine Instanz existieren
- Konstruktor könnte überall benutzt werden!



Einzelstück/Singleton

Problem

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

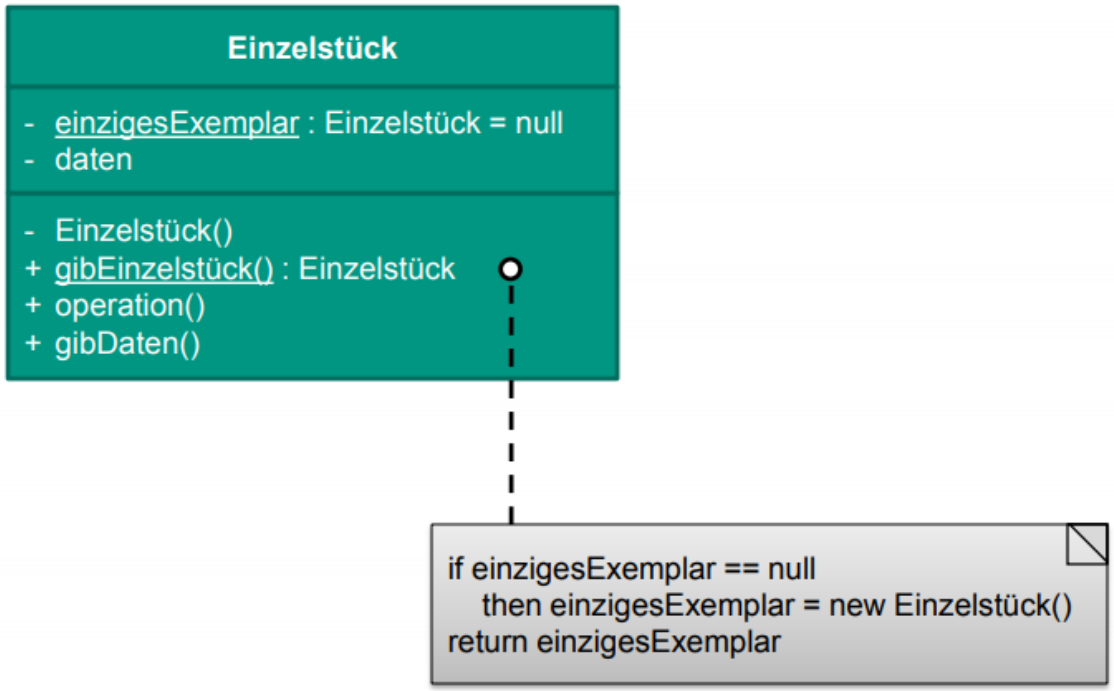


Aber warum nicht einfach statisch?

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

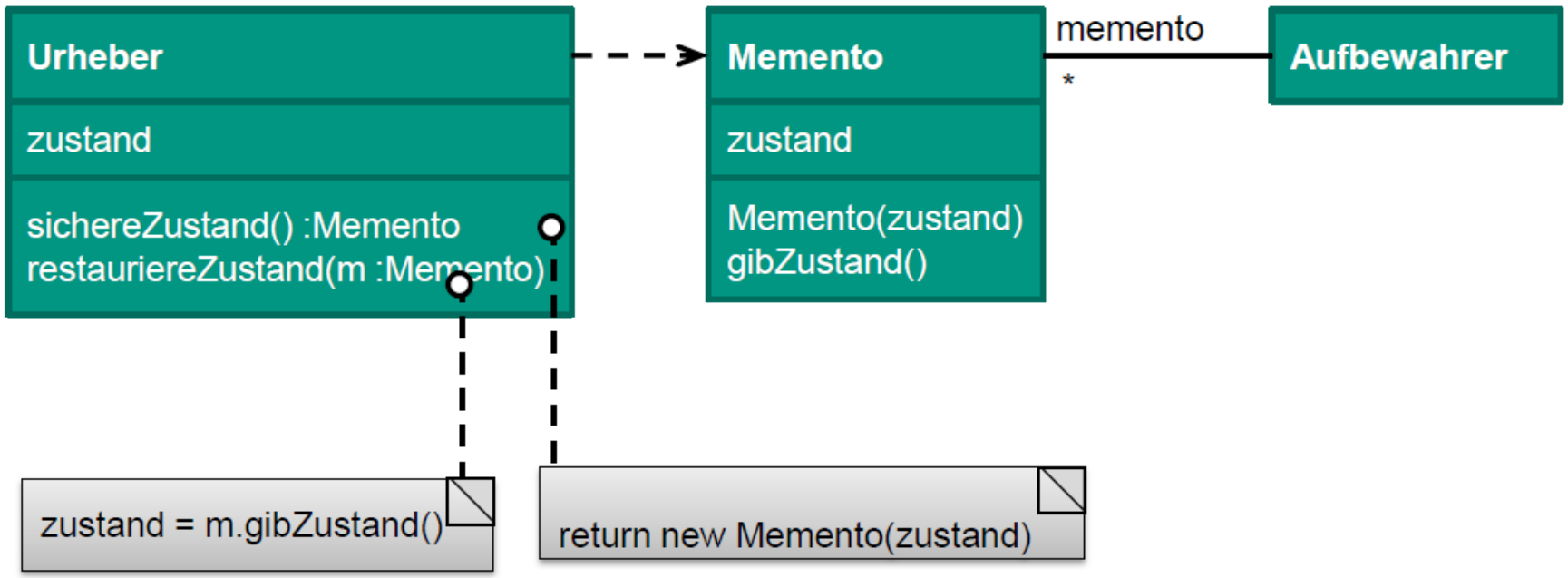
Problem

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

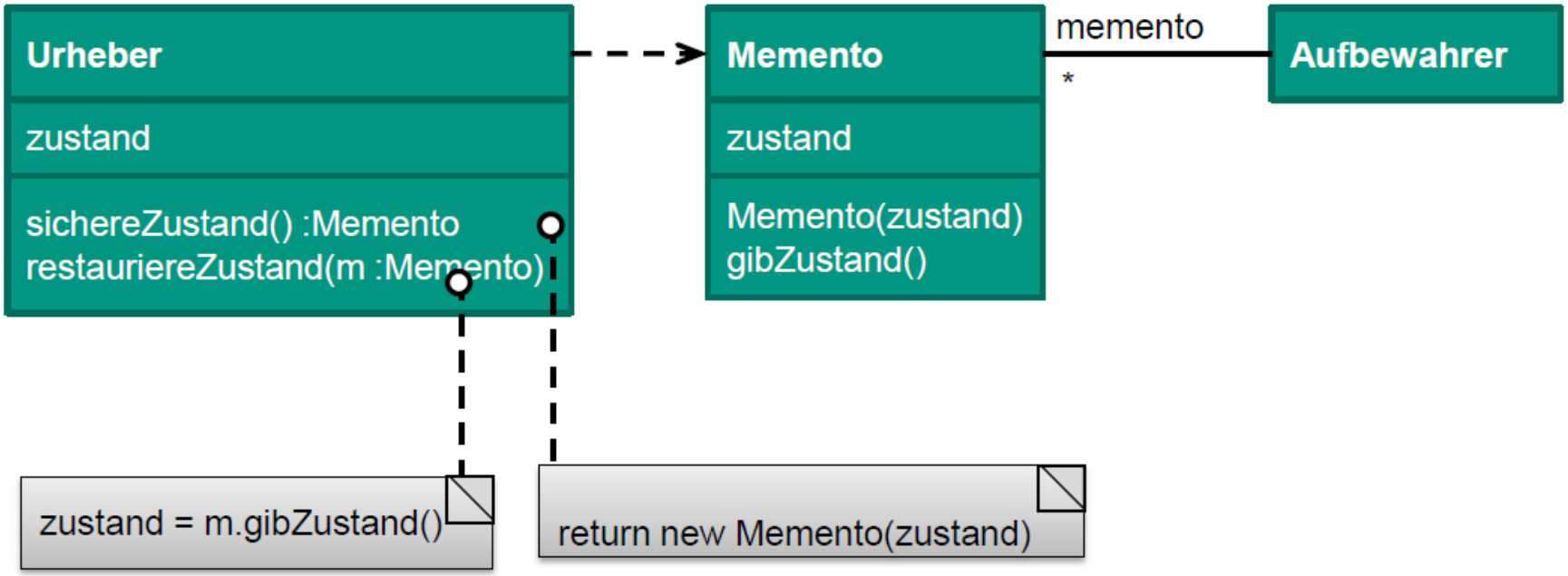
Memento

Problem

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

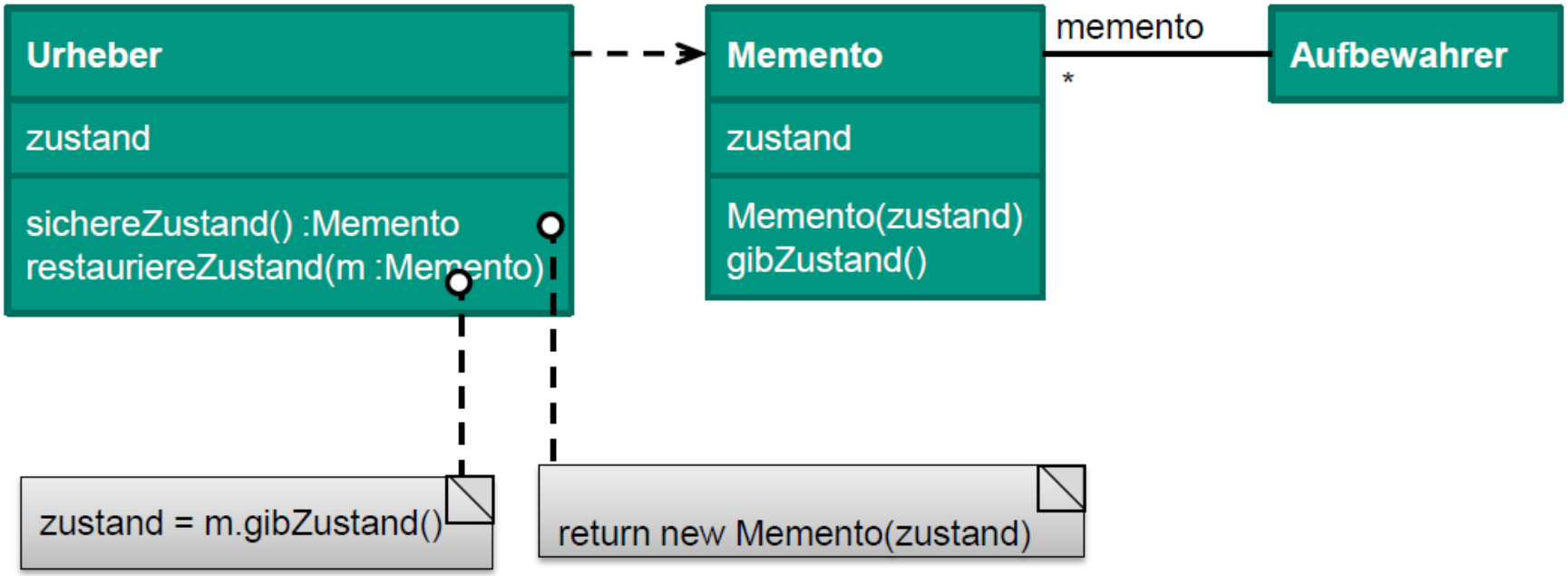


Memento



Problem gelöst?

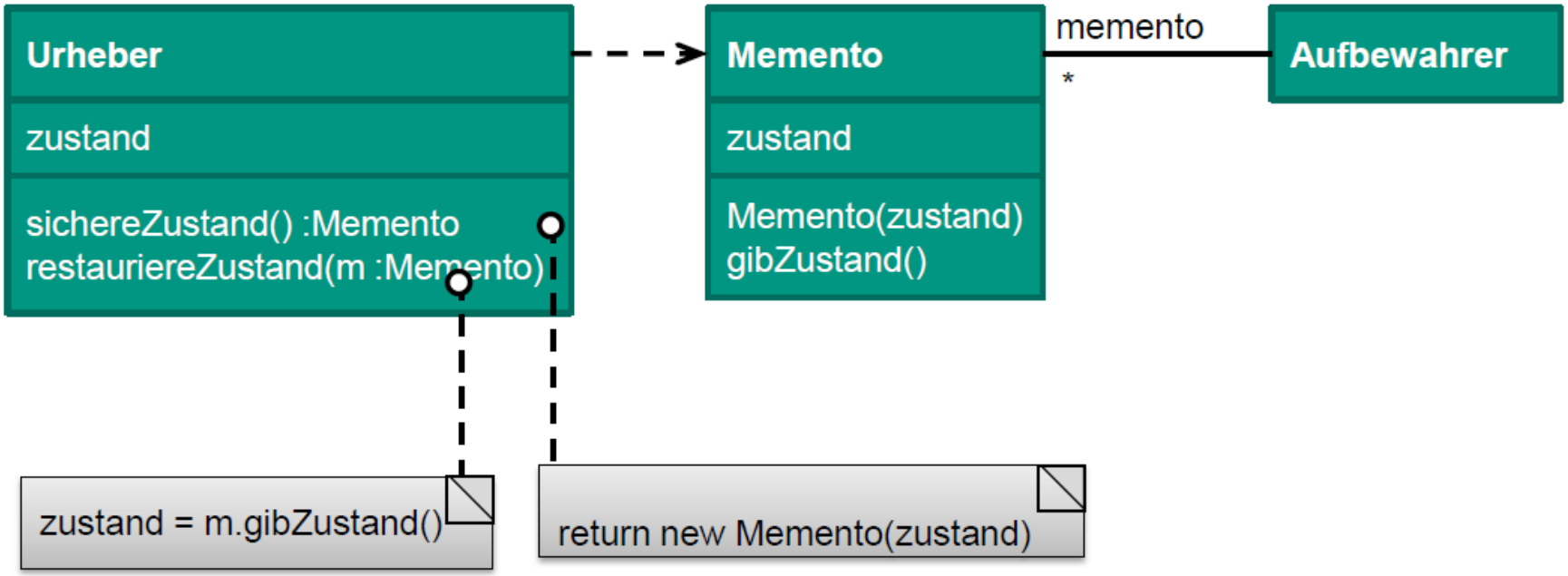
Memento



Problem gelöst?

- Ja

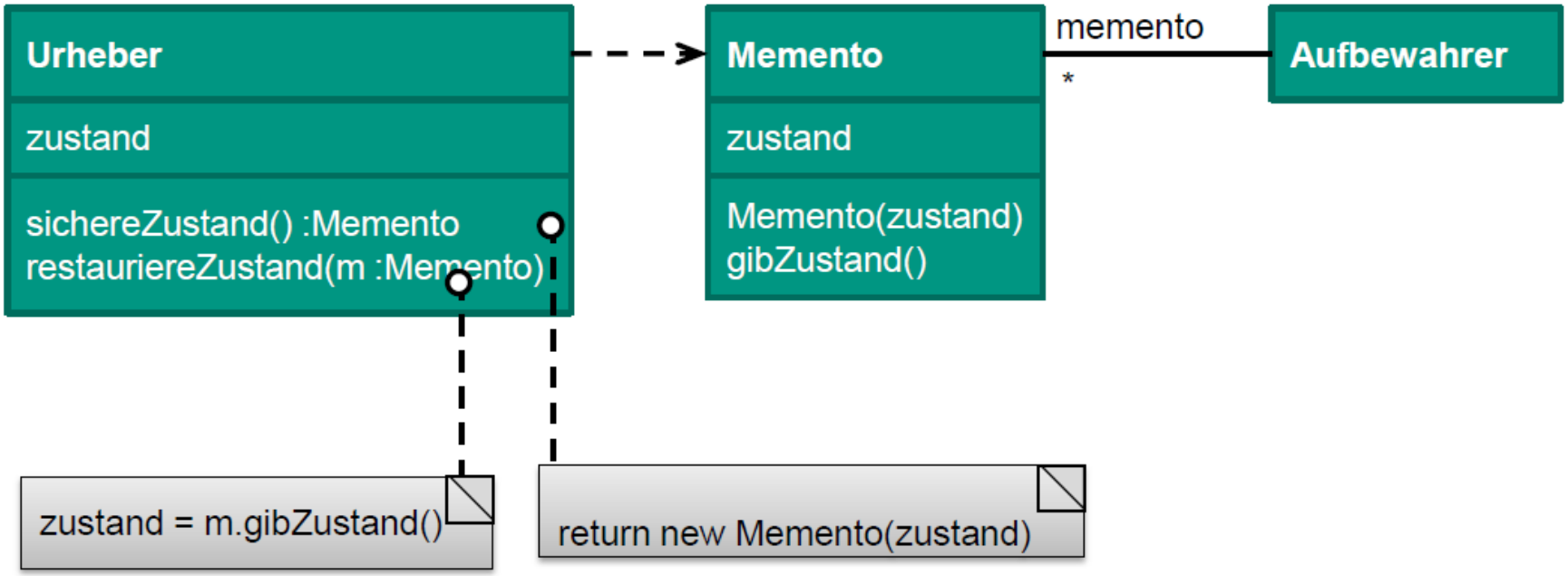
Memento



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

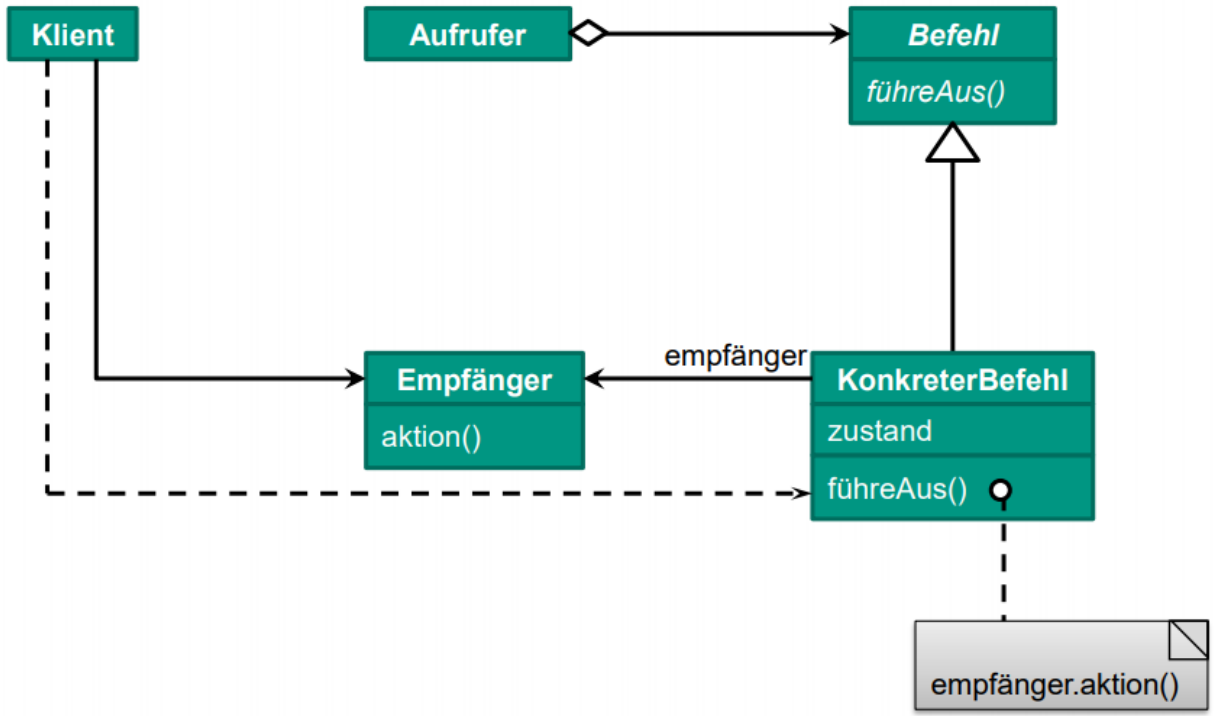
Problem

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

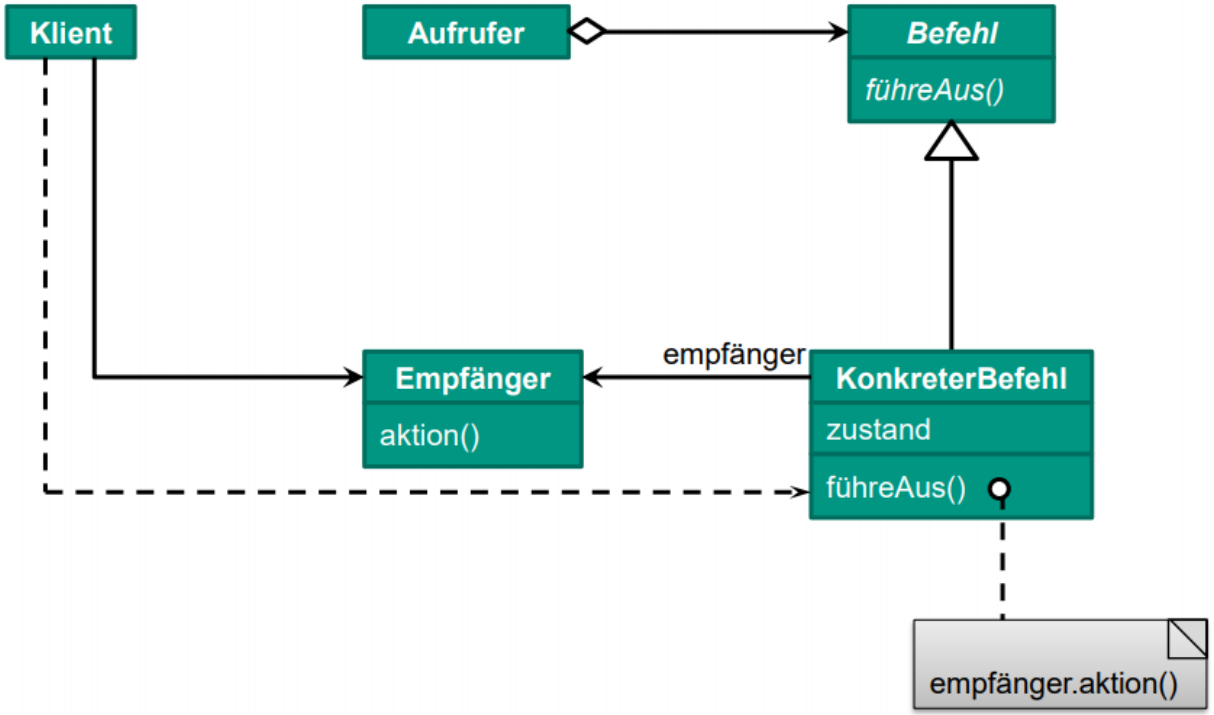
Befehl

Problem

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

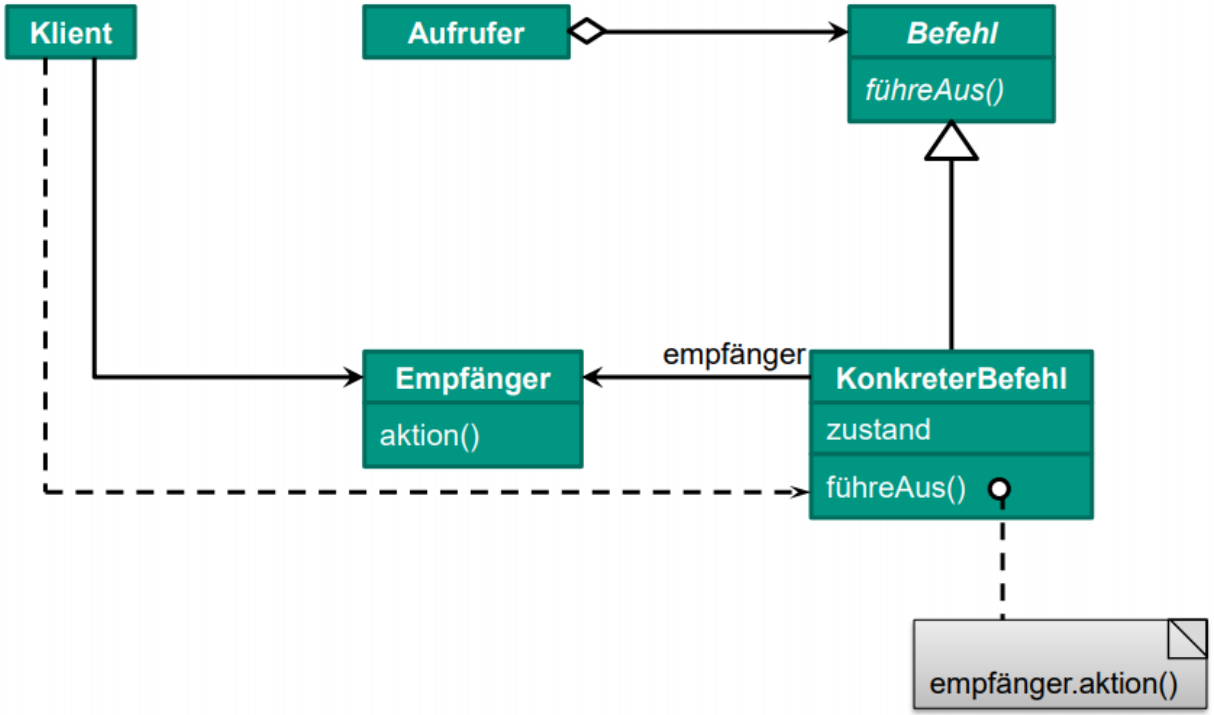


Befehl



Was haben wir erreicht?

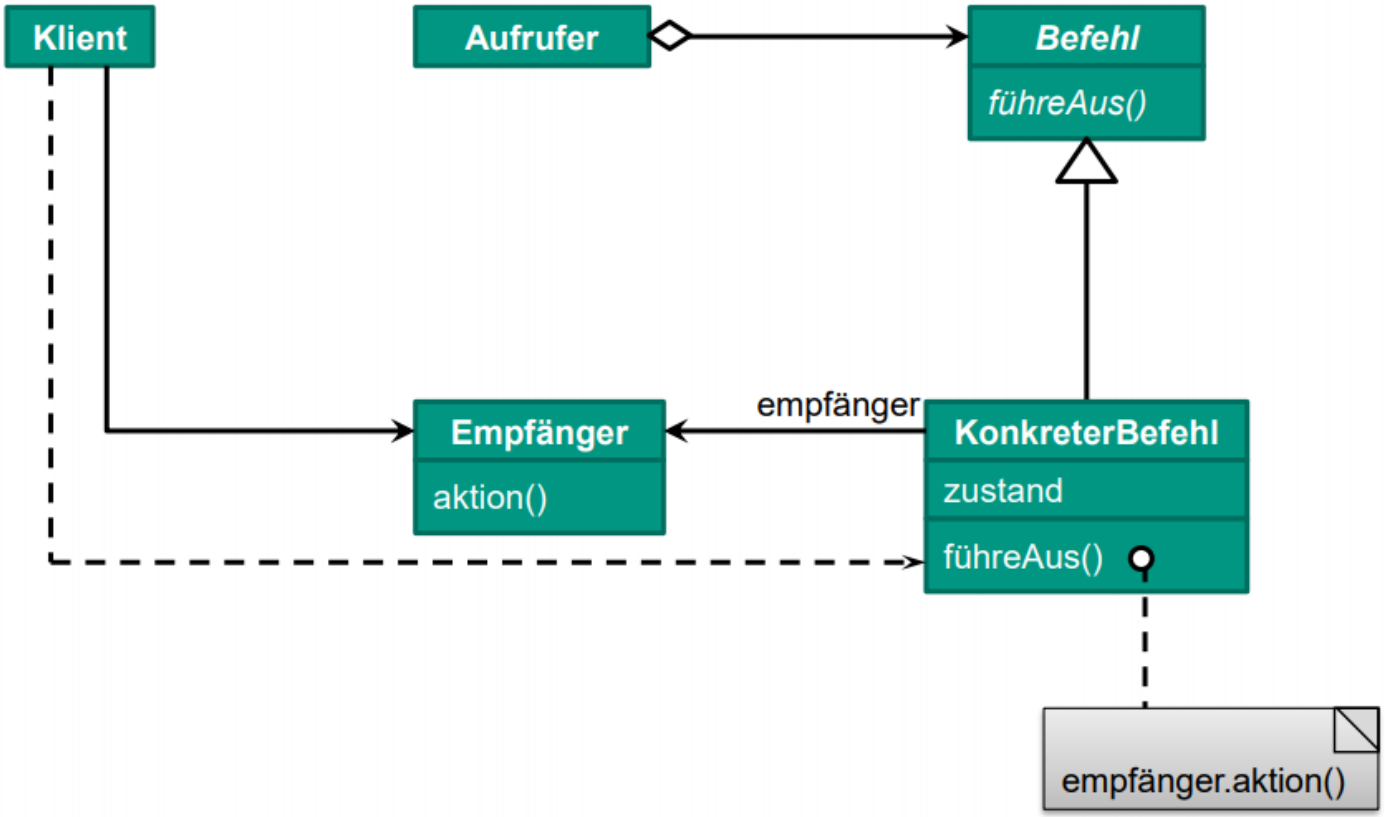
Befehl



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)