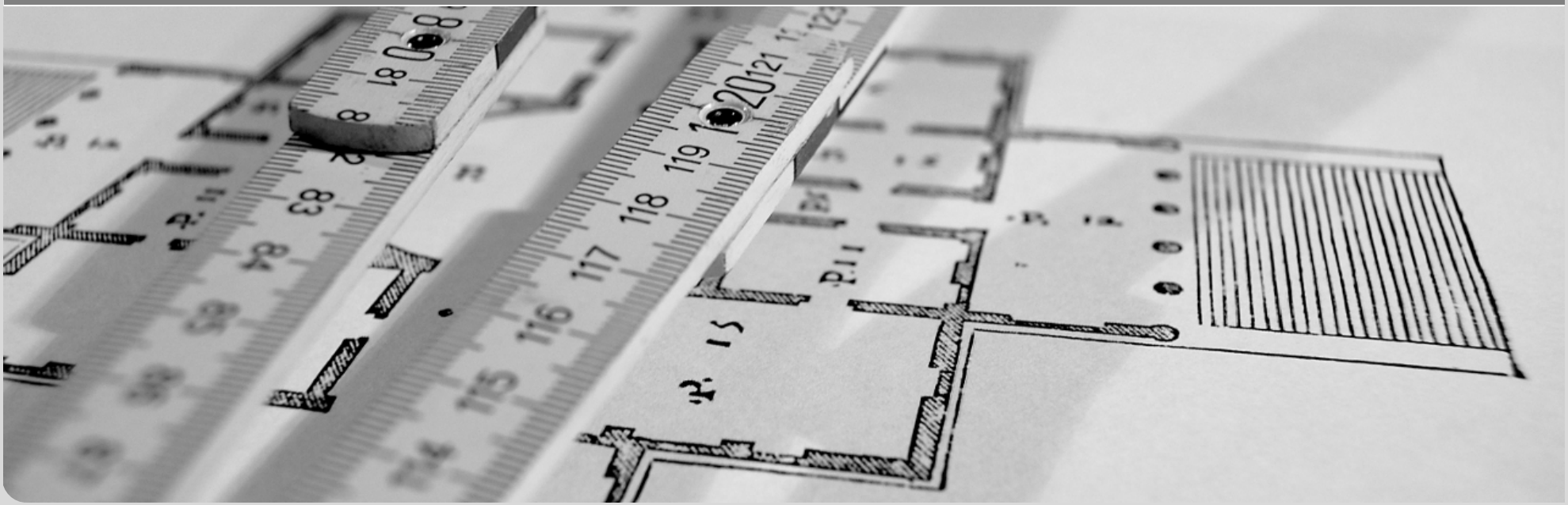
Karlsruher Institut für Technologie

Softwaretechnik 1 - 3. Tutorium

Tutorium 17

Felix Bachmann | 04.06.2019

KIT - INSTITUT FÜR PROGRAMMSTRUKTUREN UND DATENORGANISATION (IPD)



Themen

Wo sind wir?

- die ersten 2 Phasen des Wasserfallmodells sind geschafft

Wo sind wir?

- die ersten 2 Phasen des Wasserfallmodells sind geschafft
  $\Longrightarrow$ Welche waren das nochmal?

Wo sind wir?

- die ersten 2 Phasen des Wasserfallmodells sind geschafft
  $\Longrightarrow$ Welche waren das nochmal? Planung, Definition!

Wo sind wir?

- die ersten 2 Phasen des Wasserfallmodells sind geschafft
  $\Longrightarrow$ Welche waren das nochmal? Planung, Definition!
  $\Longrightarrow$ Dokumente?

Wo sind wir?

- die ersten 2 Phasen des Wasserfallmodells sind geschafft
  $\Longrightarrow$ Welche waren das nochmal? Planung, Definition!
  $\Longrightarrow$ Dokumente? Lastenheft, Pflichtenheft (+ andere. . .)

Wo sind wir?

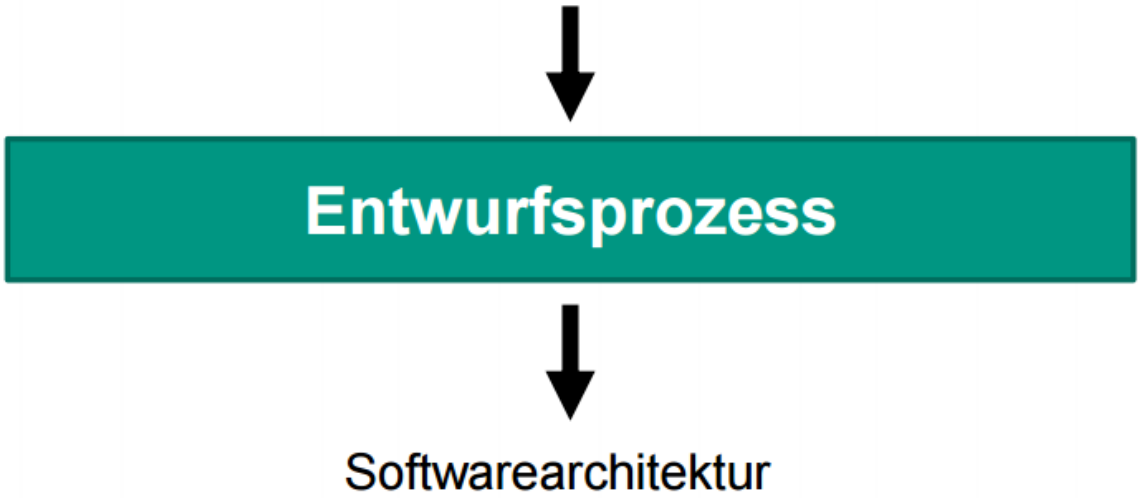
- die ersten 2 Phasen des Wasserfallmodells sind geschafft
  $\Longrightarrow$ Welche waren das nochmal? Planung, Definition!
  $\Longrightarrow$ Dokumente? Lastenheft, Pflichtenheft (+ andere...)
- und welche Phasen gibt es sonst noch?

Wo sind wir?

- die ersten 2 Phasen des Wasserfallmodells sind geschafft
  $\Longrightarrow$ Welche waren das nochmal? Planung, Definition!
  $\Longrightarrow$ Dokumente? Lastenheft, Pflichtenheft (+ andere...)
- und welche Phasen gibt es sonst noch?
- jetzt: Entwurf!

Wozu Entwurf?

- Pflichtenheft (einschl. Modelle)
- Konzept Benutzungsoberfläche
- Benutzerhandbuch + Hilfekonzept



Softwarearchitektur ist Grundlage für Implementierung!

Entwurf vs Planung/Definition

- Planung, Definition: **Was** ist zu implementieren?
  - müssen wir mit dem Kunden besprechen

Entwurf vs Planung/Definition

- Planung, Definition: **Was** ist zu implementieren?
  - müssen wir mit dem Kunden besprechen
- Entwurf: **Wie** ist das System zu implementieren?
  - können wir uns selbst überlegen

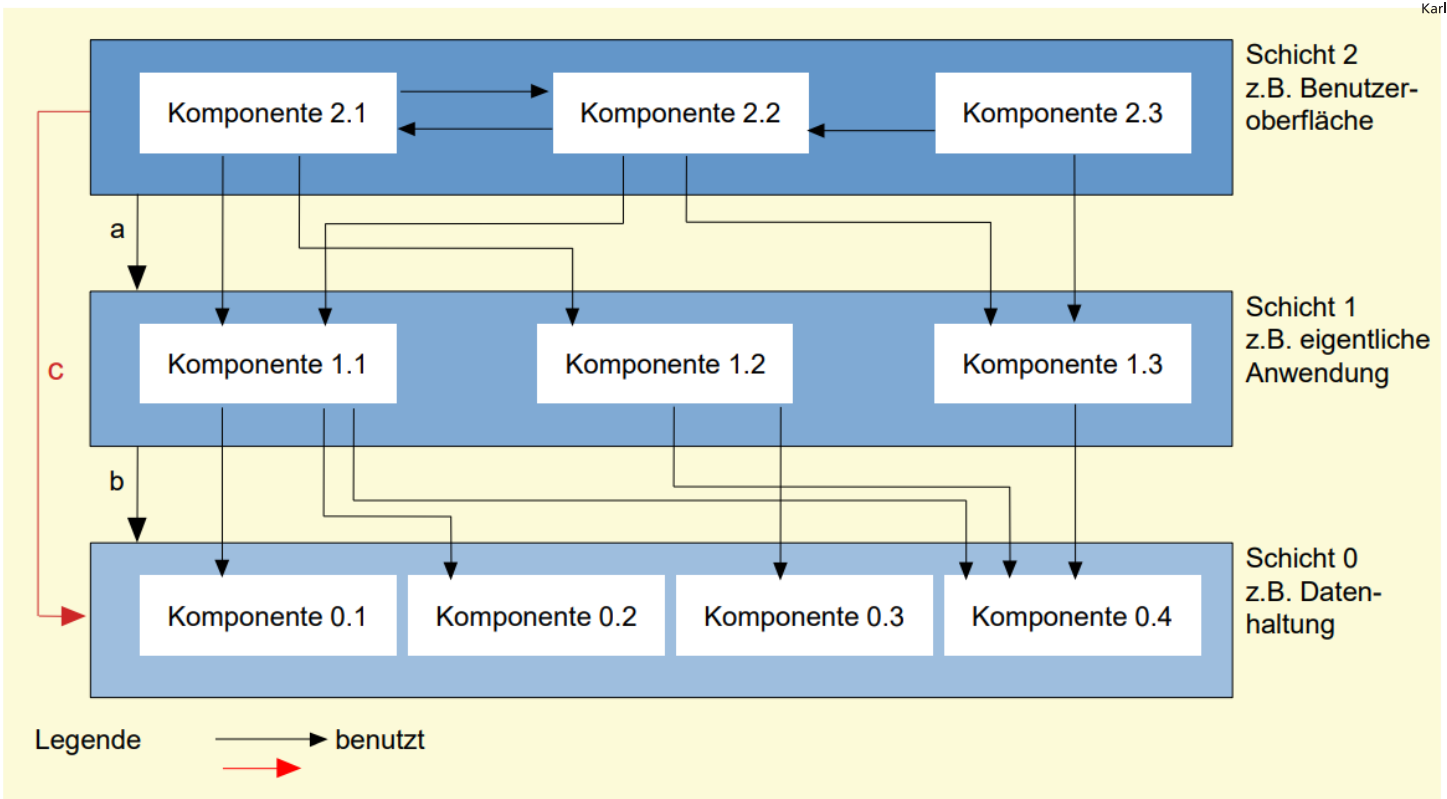
Architekturstile

- legen den Grobaufbau der Software fest
  - heißt: wir sind noch nicht (unbedingt) auf Klassen-Ebene
- in der Vorlesung besprochen
  - **Schichtenarchitektur**
  - **Klient/Dienstgeber**
  - **Partnernetze**
  - **Modell-Präsentation-Steuerung**
  - **Fließband**
  - **Rahmenarchitektur**
  - Datenablage
  - Dienstorientierte Architektur

Schichtenarchitektur

- Funktionalität (Pakete, Klassen, . . .) in Schichten getrennt
- Schichten „aufeinander gestapelt“
- jede Schicht
  - bietet seine Funktionalität den höheren Schichten an
  - nutzt Funktionalität der tieferen Schichten
- Arten von Schichtenarchitekturen
  - intransparent: jede Schicht kann nur auf direkt darunterliegende Schicht zugreifen
  - transparent: jede Schicht kann auch weiter unten liegende Schichten benutzen

Schichtenarchitektur

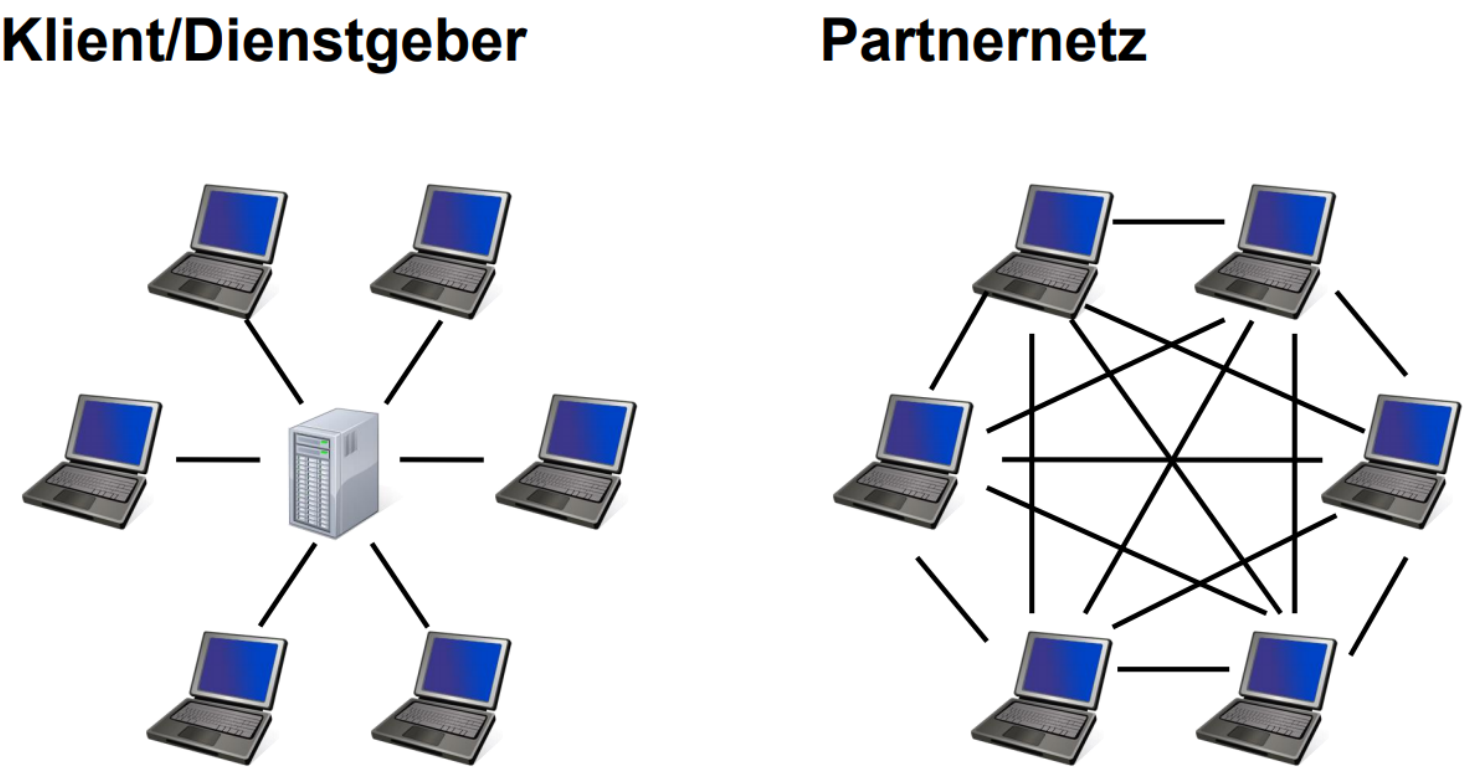


- wichtige Beispiele
  - Betriebssysteme
    - siehe Vorlesung „Betriebssysteme“ (3. Semester)
  - Protokolltürme (aka. protocol stacks)
    - siehe Vorlesung „Einführung in Rechnernetze“ (4. Semester)

Klient/Dienstgeber und Partnernetze

- eher bekannt als *Client/Server* und *Peer-to-Peer*
- Client/Server
  - Server bietet Dienst an
  - Client nutzt diesen Dienst
- Peer-to-Peer
  - jeder Teilnehmer kann Dienst anbieten und nutzen

Klient/Dienstgeber und Partnernetze



- Vor- und Nachteile jeweils?

Modell-Präsentation-Steuerung

- eher bekannt als *model-view-controller*, *MVC*
- Software in drei große Bereiche (oft Pakete) unterteilt
  - Model
    - Datenspeicherung, Anwendungslogik
  - View
    - Darstellung der Daten
    - Schaltflächen für Benutzerinteraktion
  - Controller
    - verwaltet Model und View
    - zuständig für Interaktion mit Nutzer
    - Benutzerinteraktion mit View soll Model-Daten verändern
      → aktualisiere Model

Modell-Präsentation-Steuerung

- eher bekannt als *model-view-controller*, *MVC*
- Software in drei große Bereiche (oft Pakete) unterteilt
  - Model
    - Datenspeicherung, Anwendungslogik
  - View
    - Darstellung der Daten
    - Schaltflächen für Benutzerinteraktion
  - Controller
    - verwaltet Model und View
    - zuständig für Interaktion mit Nutzer
    - Benutzerinteraktion mit View soll Model-Daten verändern
      → aktualisiere Model
- Was haben wir gewonnen?
  - Erweiterbarkeit: können z.B. selbes Modell verwenden für Linux, Windows, Internet
  - Austauschbarkeit/Wartbarkeit: Model unabhängig von View

Fließband

- eher bekannt als *pipeline*
- Idee: Datenverarbeitung in mehreren Stufen
  - jede Stufe verarbeitet Eingabedaten zu Ausgabedaten
  - Stufe 1 hat die Rohdaten als Eingabe
  - jede andere Stufe hat die Ausgabe der vorherigen Stufe als Eingabe

Fließband

- eher bekannt als *pipeline*
- Idee: Datenverarbeitung in mehreren Stufen
  - jede Stufe verarbeitet Eingabedaten zu Ausgabedaten
  - Stufe 1 hat die Rohdaten als Eingabe
  - jede andere Stufe hat die Ausgabe der vorherigen Stufe als Eingabe

- gut anwendbar bei Verarbeitung von Datenströmen
- praktisch bei Parallelität: Stufen können unabhängig voneinander ausgeführt werden

Fließband bei Parallelität

- angenommen wir haben 3-stufige Pipeline, dessen Stufen auf drei unterschiedlichen Prozessoren gleichzeitig ausgeführt werden

	CPU1 (Stufe 1)	CPU2 (Stufe 2)	CPU3 (Stufe 3)
t=0	x	x	x
t=1	Daten 1	x	x
t=2	Daten 2	Daten 1	x
t=3	Daten 3	Daten 2	Daten 1
t=4	Daten 4	Daten 3	Daten 2
t=5	Daten 5	Daten 4	Daten 3
...	...	...	...

Rahmenarchitektur

- eher bekannt als *framework*
- Idee
  - entwickle funktionsfähiges Programm
  - biete an bestimmten Stellen Möglichkeiten zur Erweiterung
    - *plug-ins*
- oft in open source Projekten verwendet
  - Eclipse, Atom, VSCode, . . .

Ankreuzaufgaben aus Altklausuren

- Bei einer intransparenten Schichtenarchitektur kann eine innere Schicht nur auf direkt benachbarte Schichten zugreifen.

Ankreuzaufgaben aus Altklausuren

- Bei einer intransparenten Schichtenarchitektur kann eine innere Schicht nur auf direkt benachbarte Schichten zugreifen. ✗
- Schichtenarchitekturen unterstützen das Testen von Programmen.

Ankreuzaufgaben aus Altklausuren

- Bei einer intransparenten Schichtenarchitektur kann eine innere Schicht nur auf direkt benachbarte Schichten zugreifen. ✗
- Schichtenarchitekturen unterstützen das Testen von Programmen. ✓
- Ist die Benutztrelation zwischen Modulen zyklenfrei, sind Teillieferungen bei Entwicklungsverzögerungen einzelner Module möglich.

Ankreuzaufgaben aus Altklausuren

- Bei einer intransparenten Schichtenarchitektur kann eine innere Schicht nur auf direkt benachbarte Schichten zugreifen. ✗
- Schichtenarchitekturen unterstützen das Testen von Programmen. ✓
- Ist die Benutztrelation zwischen Modulen zyklenfrei, sind Teillieferungen bei Entwicklungsverzögerungen einzelner Module möglich. ✓
- Eine Fließband-Architektur in Software kann nur auf Parallelrechnern ausgeführt werden.

Ankreuzaufgaben aus Altklausuren

- Bei einer intransparenten Schichtenarchitektur kann eine innere Schicht nur auf direkt benachbarte Schichten zugreifen. ✗
- Schichtenarchitekturen unterstützen das Testen von Programmen. ✓
- Ist die Benutztrelation zwischen Modulen zyklenfrei, sind Teillieferungen bei Entwicklungsverzögerungen einzelner Module möglich. ✓
- Eine Fließband-Architektur in Software kann nur auf Parallelrechnern ausgeführt werden. ✗
- Bei einer Schichtenarchitektur mit drei übereinander liegenden Schichten kann die mittlere Schicht auf die Dienste der oberen und der unteren Schicht zugreifen.

Ankreuzaufgaben aus Altklausuren

- Bei einer intransparenten Schichtenarchitektur kann eine innere Schicht nur auf direkt benachbarte Schichten zugreifen. ✗
- Schichtenarchitekturen unterstützen das Testen von Programmen. ✓
- Ist die Benutztrelation zwischen Modulen zyklenfrei, sind Teillieferungen bei Entwicklungsverzögerungen einzelner Module möglich. ✓
- Eine Fließband-Architektur in Software kann nur auf Parallelrechnern ausgeführt werden. ✗
- Bei einer Schichtenarchitektur mit drei übereinander liegenden Schichten kann die mittlere Schicht auf die Dienste der oberen und der unteren Schicht zugreifen. ✗
- Wenn die Benutztrelation zyklenfrei ist, heißt sie Benutzthierarchie.

Ankreuzaufgaben aus Altklausuren

- Bei einer intransparenten Schichtenarchitektur kann eine innere Schicht nur auf direkt benachbarte Schichten zugreifen. ✗
- Schichtenarchitekturen unterstützen das Testen von Programmen. ✓
- Ist die Benutztrelation zwischen Modulen zyklenfrei, sind Teillieferungen bei Entwicklungsverzögerungen einzelner Module möglich. ✓
- Eine Fließband-Architektur in Software kann nur auf Parallelrechnern ausgeführt werden. ✗
- Bei einer Schichtenarchitektur mit drei übereinander liegenden Schichten kann die mittlere Schicht auf die Dienste der oberen und der unteren Schicht zugreifen. ✗
- Wenn die Benutztrelation zyklenfrei ist, heißt sie Benutzthierarchie. ✓

Welcher Architekturstil ist gemeint?

„Wir wollen unseren customern solutions anbieten, die möglichst extensible sind. Falls die Software successfull ist, können wir später neuen stuff implementen, ohne den alten Code zu ändern. Über einen marketplace könnten wir dann auch noch Geld dafür getten. Sheesh.“

Welcher Architekturstil ist gemeint?

„Wir wollen unseren customern solutions anbieten, die möglichst extensible sind. Falls die Software successfull ist, können wir später neuen stuff implementen, ohne den alten Code zu ändern. Über einen marketplace könnten wir dann auch noch Geld dafür getten. Sheesh.“

Framework

Welcher Architekturstil ist gemeint?

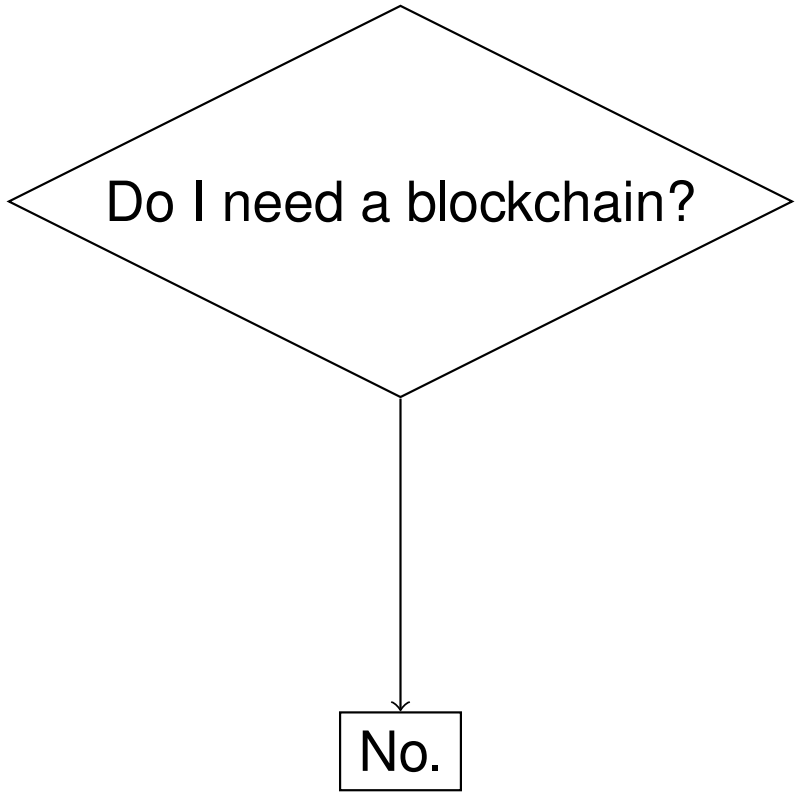
„Auf der letzten Conference waren viele Speaker, die etwas von einer ‚Blockkette‘ erzählt haben. Da wir Amazon und Co. nicht trusten, könnten wir sowas usen um unseren Service decentralized anzubieten. Besides können wir damit money einsparen.“

Welcher Architekturstil ist gemeint?

„Auf der letzten Conference waren viele Speaker, die etwas von einer ‚Blockkette‘ erzählt haben. Da wir Amazon und Co. nicht trusten, könnten wir sowas usen um unseren Service decentralized anzubieten. Besides können wir damit money einsparen.“

Peer-to-Peer

Eine Entwurfsentscheidung :)



Welcher Architekturstil ist gemeint?

„Ganz wichtig ist mir ja ein Separation-of-Concerns: Die Design-People sollen einen user-zentrierten Look-and-Feel entwerfen. In the meantime sollen sich die devs um neue functionality kümmern. Beide sollen independently arbeiten.“

Welcher Architekturstil ist gemeint?

„Ganz wichtig ist mir ja ein Separation-of-Concerns: Die Design-People sollen einen user-zentrierten Look-and-Feel entwerfen. In the meantime sollen sich die devs um neue functionality kümmern. Beide sollen independently arbeiten.“

MVC

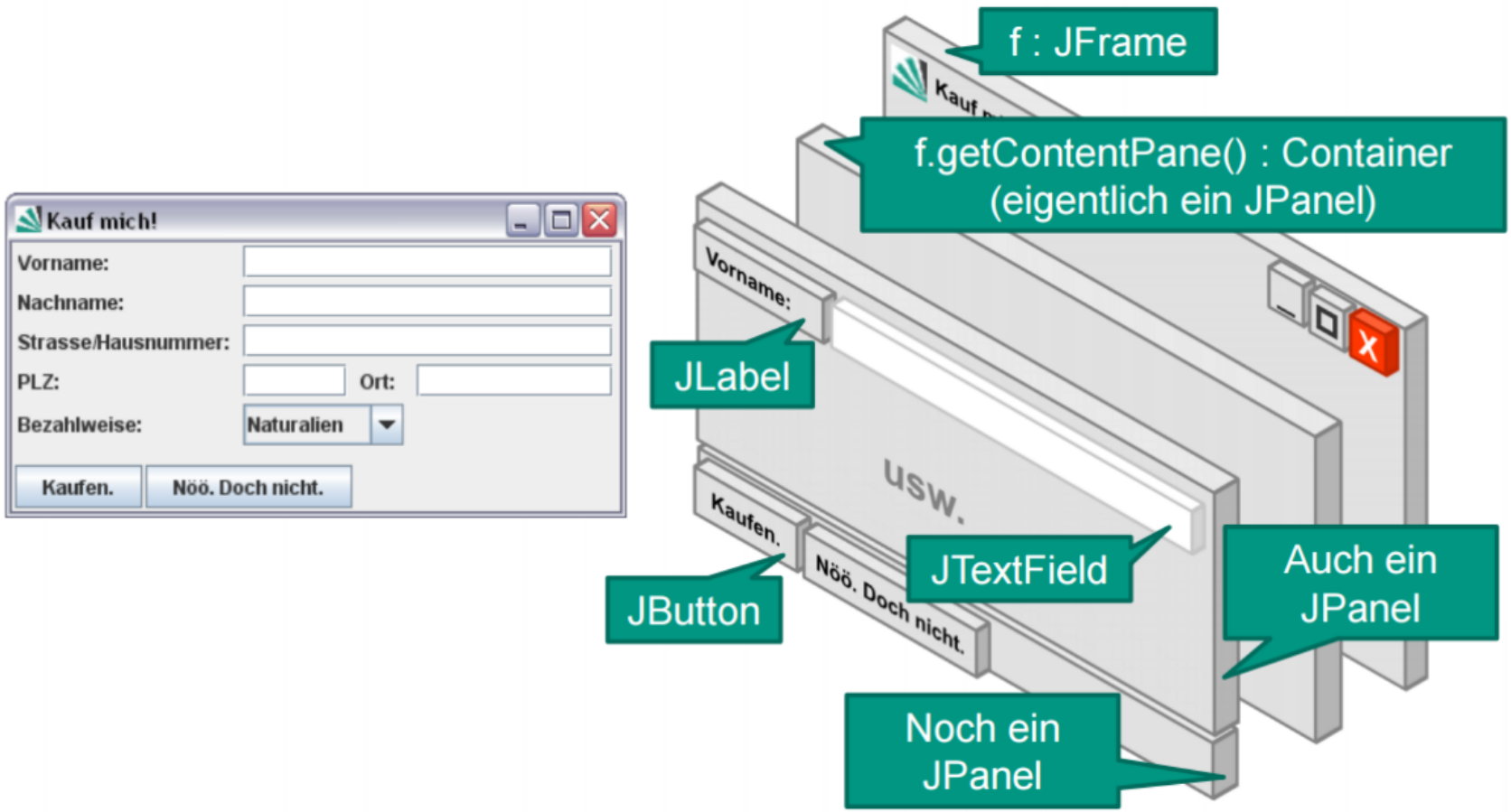
Java Swing

- mit Swing kann man GUIs bauen, ohne was anderes als Java zu benutzen
- graphische Elemente sind als Klassen implementiert

wichtige Klassen und Interfaces

`JFrame`	Fenster (nicht-blockierend)
`JDialog`	Dialogfenster (blockierend)
`JPanel`	Container, hier kommen die Bedienelemente rein
`JFileChooser`	Dateiauswahlfenster
`JButton`	Button
`JTextField`	Eingabefeld
`JMenuBar`	Menüleiste
`JLabel`	Text/Bild anzeigen
`LayoutManager`	um Layout in `JPanel` zu setzen

Übersicht


Kauf mich!
Vorname:
Nachname:
Strasse/Hausnummer:
PLZ:
Ort:
Bezahlweise:
Naturalien
Kaufen.
Nöö. Doch nicht.
f : JFrame
f.getContentPane() : Container
(eigentlich ein JPanel)
Vorname:
JLabel
usw.
Kaufen.
Nöö. Doch nicht.
JButton
JTextField
Auch ein
JPanel
Noch ein
JPanel

Wie erstelle ich meine eigene GUI?

grundsätzlich 2 Möglichkeiten

`JFrame` erstellen und anpassen

```
JFrame f = new JFrame("Titel");
f.add(new JButton("Knopf-Text"));
f.setVisible(true);
```

Wie erstelle ich meine eigene GUI?

grundsätzlich 2 Möglichkeiten

`JFrame` erstellen und anpassen

```
JFrame f = new JFrame("Titel");
f.add(new JButton("Knopf-Text"));
f.setVisible(true);
```

von `JFrame` erben, Anpassung z.B. in Konstruktor

```
public class MyFrame extends JFrame {
    MyFrame() { add(new JButton("Knopf-Text")); // ... }
}
// ...
JFrame f = new MyFrame();
```

Wie erstelle ich eigene Elemente?

ebenfalls 2 Möglichkeiten (`JFrame` ist auch nur ein Element)

Instanzen erstellen und anpassen

```
JButton b = new JButton();
b.setText("Click me!");
b.setToolTipText("Just do it, bro!");
```

Wie erstelle ich eigene Elemente?

ebenfalls 2 Möglichkeiten (`JFrame` ist auch nur ein Element)

Instanzen erstellen und anpassen

```
JButton b = new JButton();
b.setText("Click me!");
b.setToolTipText("Just do it, bro!");
```

Klassen überschreiben, Anpassung z.B. in Konstruktor

```
public class MyButton extends JButton {
    MyButton() { setText("Click me!"); // ... }
}
// ...
JButton b = new MyButton();
```

Link-Sammlung

- allgemeine Anlaufstelle
  - `https://docs.oracle.com/javase/tutorial/uiswing/`
- Layout Manager erklärt
  - `https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html`
- jedes Element erklärt
  - `https://docs.oracle.com/javase/tutorial/uiswing/components/componentlist.html`

Karlsruher Institut für Technologie

kleines Beispiel

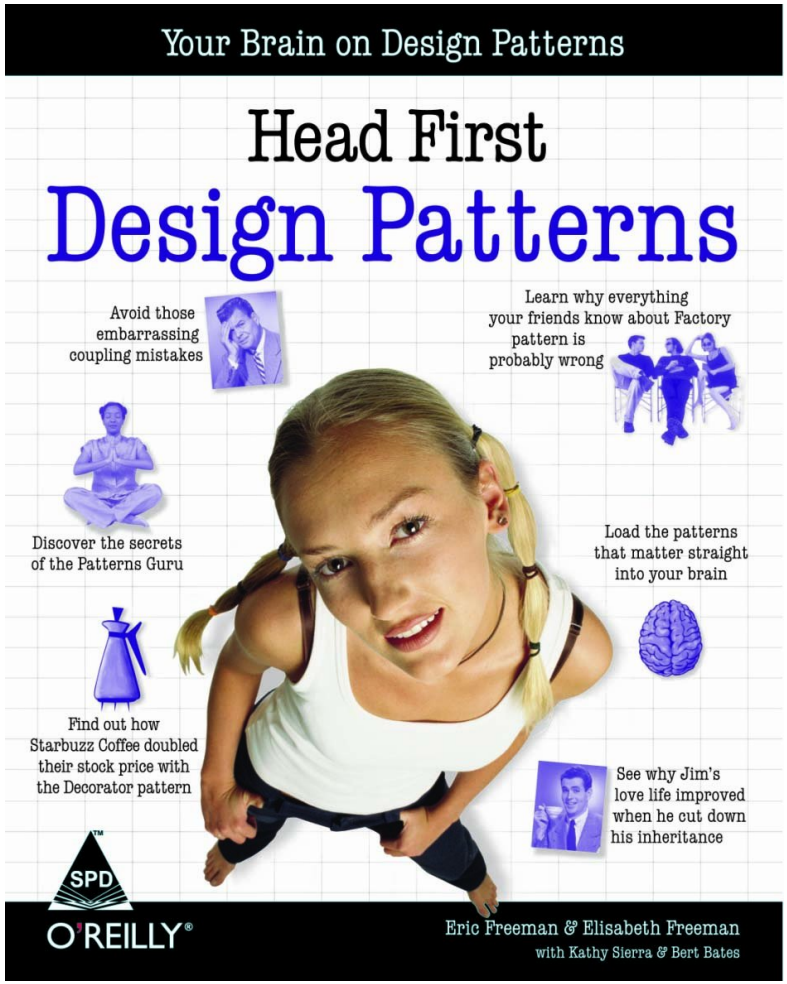
Evaluation

Tutorien-Halbzeit erreicht: Her mit der Kritik!

`https://forms.gle/N7rjXDUu8BR6gx2KA`

Empfehlenswerte Literatur (wirklich!)

- knapp 700 Seiten
  - als interaktives Nachschlagewerk



Was sind Entwurfsmuster?

Entwurfsmuster

Ein Software-Entwurfsmuster beschreibt eine Familie von Lösungen für ein Software-Entwurfsproblem.

Was sind Entwurfsmuster?

Entwurfsmuster

Ein Software-Entwurfsmuster beschreibt eine Familie von Lösungen für ein Software-Entwurfsproblem.

- schematische Klassendiagramme zur Lösung von häufig auftretenden Problemen

Was sind Entwurfsmuster?

Entwurfsmuster

Ein Software-Entwurfsmuster beschreibt eine Familie von Lösungen für ein Software-Entwurfsproblem.

- schematische Klassendiagramme zur Lösung von häufig auftretenden Problemen
- Wiederverwendung von Entwurfswissen $\Longrightarrow$ Rad nicht neu erfinden!

Was sind Entwurfsmuster?

Entwurfsmuster

Ein Software-Entwurfsmuster beschreibt eine Familie von Lösungen für ein Software-Entwurfsproblem.

- schematische Klassendiagramme zur Lösung von häufig auftretenden Problemen
- Wiederverwendung von Entwurfswissen $\Longrightarrow$ Rad nicht neu erfinden!

Wozu Entwurfsmuster?

- erleichtern Kommunikation

Wozu Entwurfsmuster?

- erleichtern Kommunikation
- erleichtern “gute“ Entwürfe
  - damit auch das Schreiben von wartbarem/erweiterbarem Code

Geheimnisprinzip

Geheimnis- / Kapselungsprinzip

Jedes Modul verbirgt eine wichtige Entwurfsentscheidung hinter einer wohldefinierten Schnittstelle, die sich bei einer Änderung der Entscheidung nicht mit ändert.

Geheimnisprinzip

Geheimnis- / Kapselungsprinzip

Jedes Modul verbirgt eine wichtige Entwurfsentscheidung hinter einer wohldefinierten Schnittstelle, die sich bei einer Änderung der Entscheidung nicht mit ändert.

Warum eigentlich? Lokalitätsprinzip!

lokale Änderungen sollen sich nicht auf andere Teile auswirken

$\Longrightarrow$ weniger Fehler und Arbeit

Beispiel?

Geheimnisprinzip

Geheimnis- / Kapselungsprinzip

Jedes Modul verbirgt eine wichtige Entwurfsentscheidung hinter einer wohldefinierten Schnittstelle, die sich bei einer Änderung der Entscheidung nicht mit ändert.

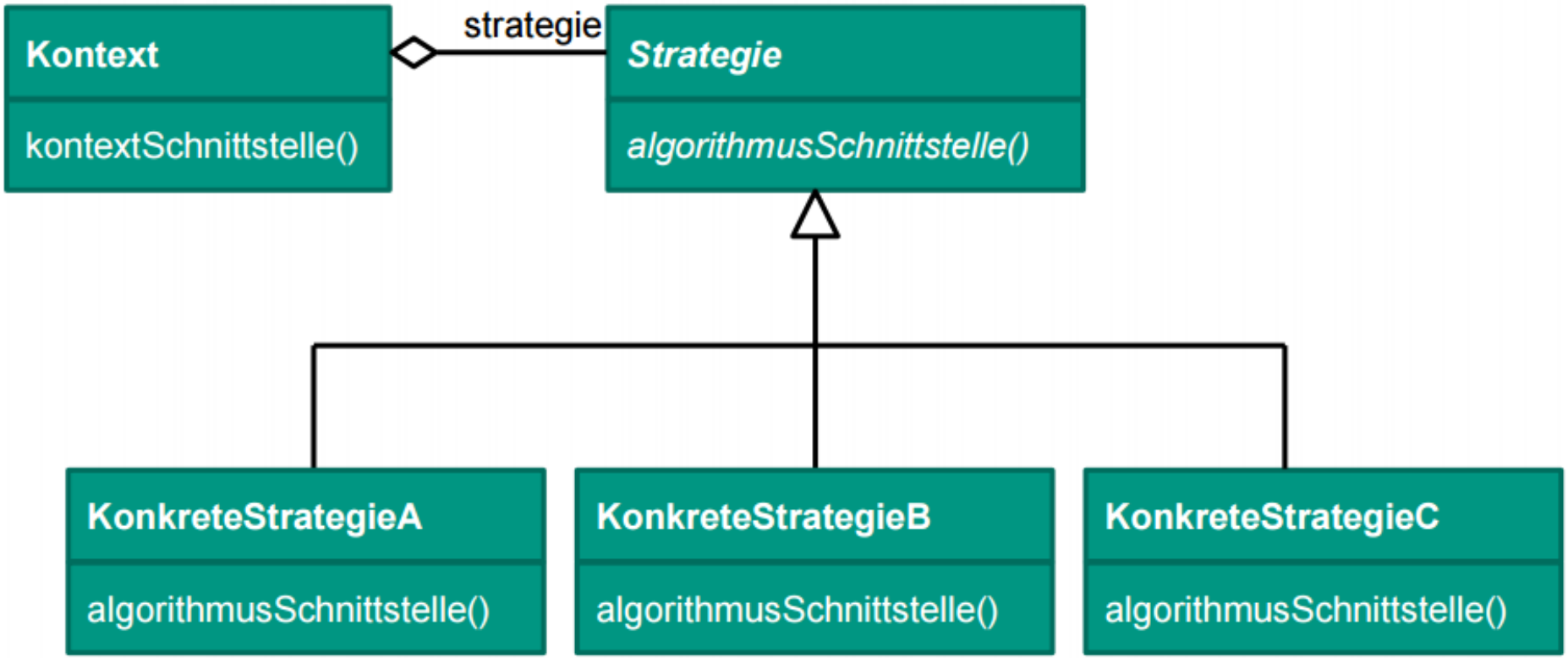
Warum eigentlich? Lokalitätsprinzip!

lokale Änderungen sollen sich nicht auf andere Teile auswirken
$\Longrightarrow$ weniger Fehler und Arbeit

Beispiel? $\Longrightarrow$ private Attribute mit get()- und set()-Methoden

Vorgriff: Entwurfsmuster Strategie

- Ziel: Algorithmen kapseln, austauschbar machen
- wird in vielen Entwurfsmustern verwendet



Quiz (Ankreuzaufgaben aus Klausuren)

Wahr oder falsch?

- Das Entwurfsmuster Strategie bietet die Möglichkeit, eine Klasse mit einer von mehreren möglichen Verhaltensweisen zu konfigurieren.

Quiz (Ankreuzaufgaben aus Klausuren)

Wahr oder falsch?

- Das Entwurfsmuster Strategie bietet die Möglichkeit, eine Klasse mit einer von mehreren möglichen Verhaltensweisen zu konfigurieren. ✓

Quiz (Ankreuzaufgaben aus Klausuren)

Wahr oder falsch?

- Das Entwurfsmuster Strategie bietet die Möglichkeit, eine Klasse mit einer von mehreren möglichen Verhaltensweisen zu konfigurieren. ✓
- Das Muster Strategie kapselt austauschbares Verhalten und verwendet Delegierung, um zu entscheiden, welches Verhalten verwendet wird.

Quiz (Ankreuzaufgaben aus Klausuren)

Wahr oder falsch?

- Das Entwurfsmuster Strategie bietet die Möglichkeit, eine Klasse mit einer von mehreren möglichen Verhaltensweisen zu konfigurieren. ✓
- Das Muster Strategie kapselt austauschbares Verhalten und verwendet Delegierung, um zu entscheiden, welches Verhalten verwendet wird. ✓

Quiz (Ankreuzaufgaben aus Klausuren)

Wahr oder falsch?

- Das Entwurfsmuster Strategie bietet die Möglichkeit, eine Klasse mit einer von mehreren möglichen Verhaltensweisen zu konfigurieren. ✓
- Das Muster Strategie kapselt austauschbares Verhalten und verwendet Delegierung, um zu entscheiden, welches Verhalten verwendet wird. ✓
- Das Hinzufügen einer neuen konkreten Strategie erfordert keine Änderung existierender konkreter Strategien.

Quiz (Ankreuzaufgaben aus Klausuren)

Wahr oder falsch?

- Das Entwurfsmuster Strategie bietet die Möglichkeit, eine Klasse mit einer von mehreren möglichen Verhaltensweisen zu konfigurieren. ✓
- Das Muster Strategie kapselt austauschbares Verhalten und verwendet Delegierung, um zu entscheiden, welches Verhalten verwendet wird. ✓
- Das Hinzufügen einer neuen konkreten Strategie erfordert keine Änderung existierender konkreter Strategien. ✓

Kategorien der Entwurfsmuster

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

Entkopplungs-Muster

- übergeordnetes Ziel: System in Teile aufspalten, die unabhängig voneinander sind
  $\Longrightarrow$ Teile austauschbar bzw. veränderbar

Adapter

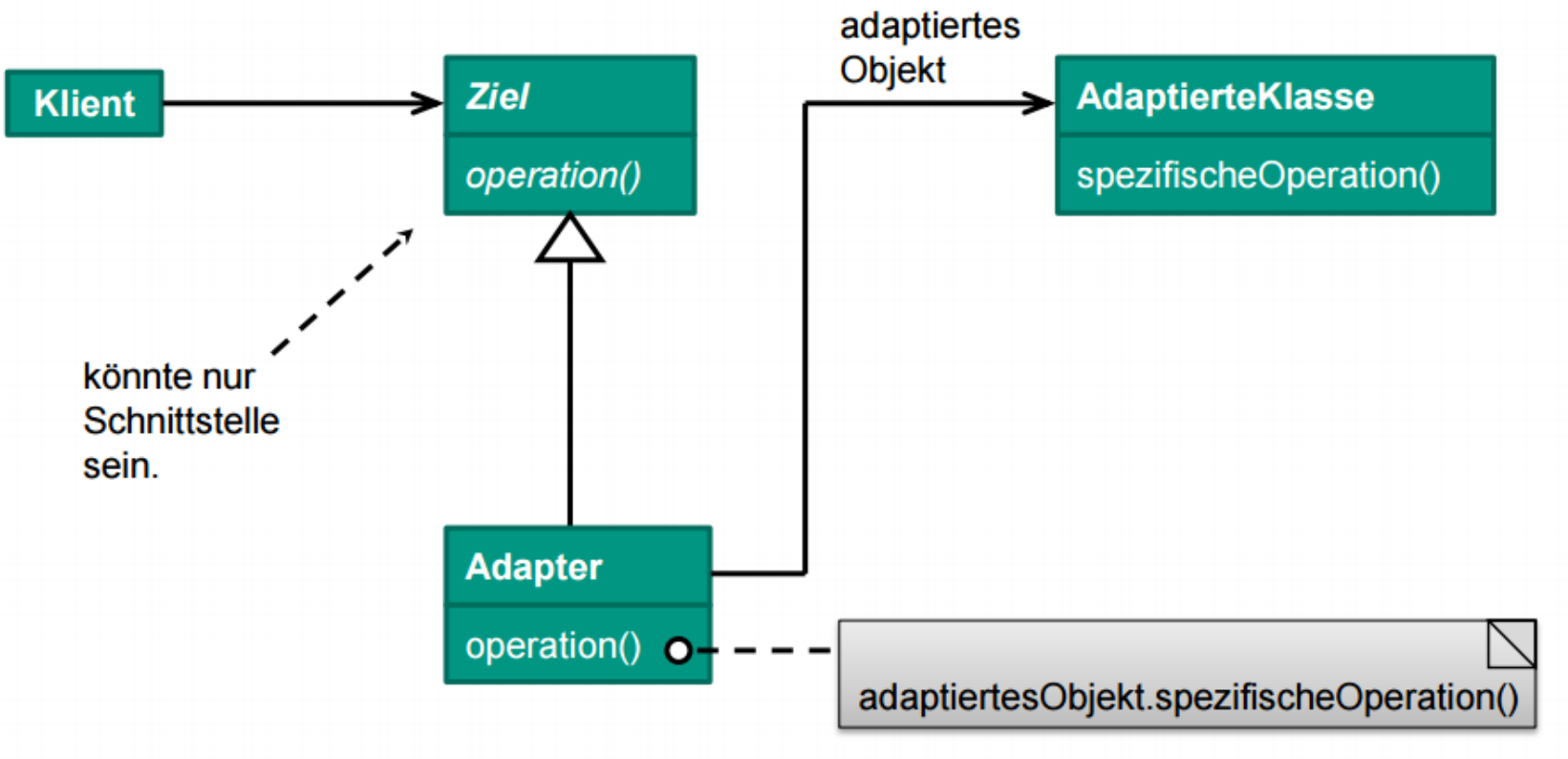
Problem

- Klassen mit inkompatiblen Schnittstellen, die wir aber zusammen benutzen wollen
- Schnittstellen nicht änderbar (z.B. externe Bibliotheken)

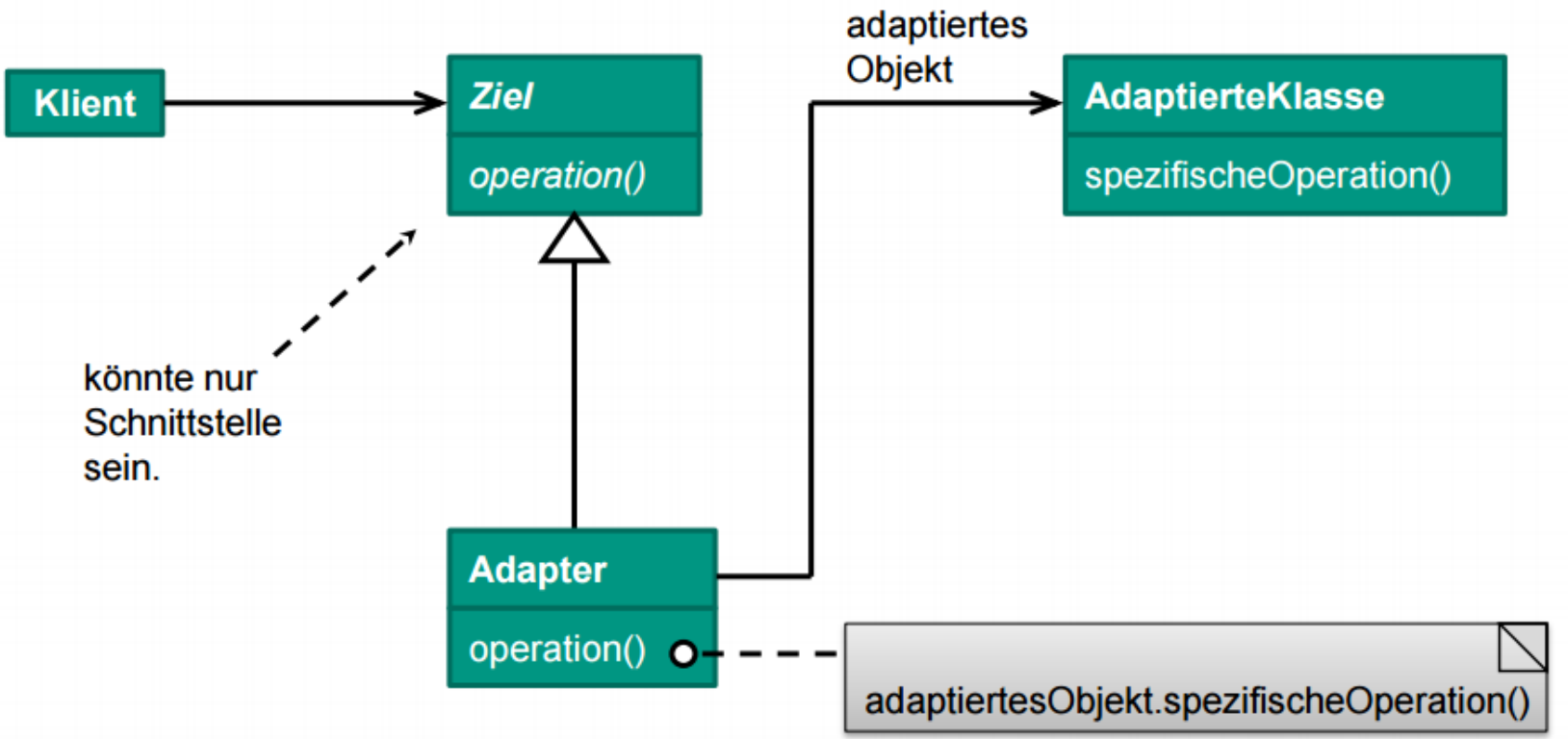
Adapter

Problem

- Klassen mit inkompatiblen Schnittstellen, die wir aber zusammen benutzen wollen
- Schnittstellen nicht änderbar (z.B. externe Bibliotheken)

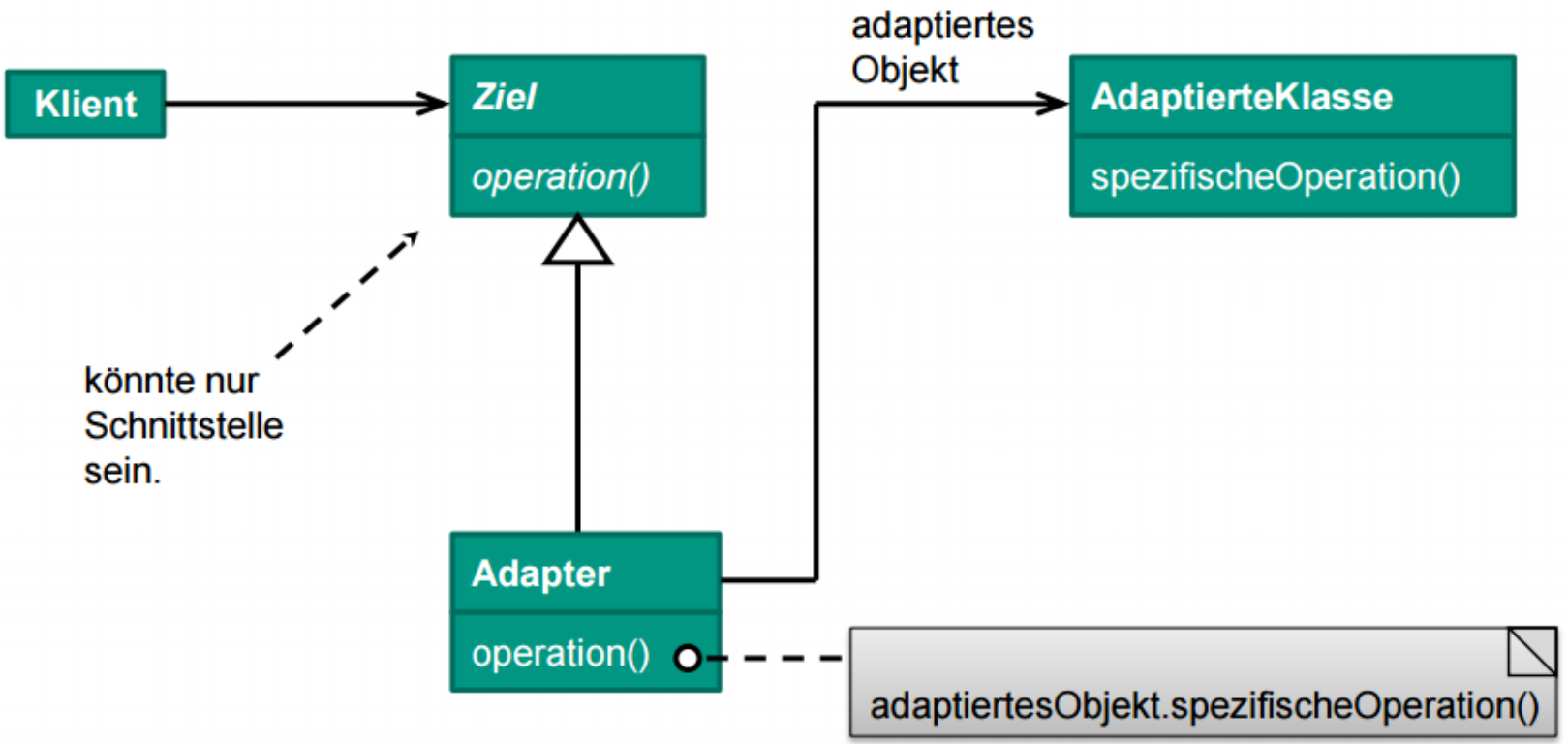


Adapter (Objektadapter)



Preisfrage: Wo ist die Entkopplung?

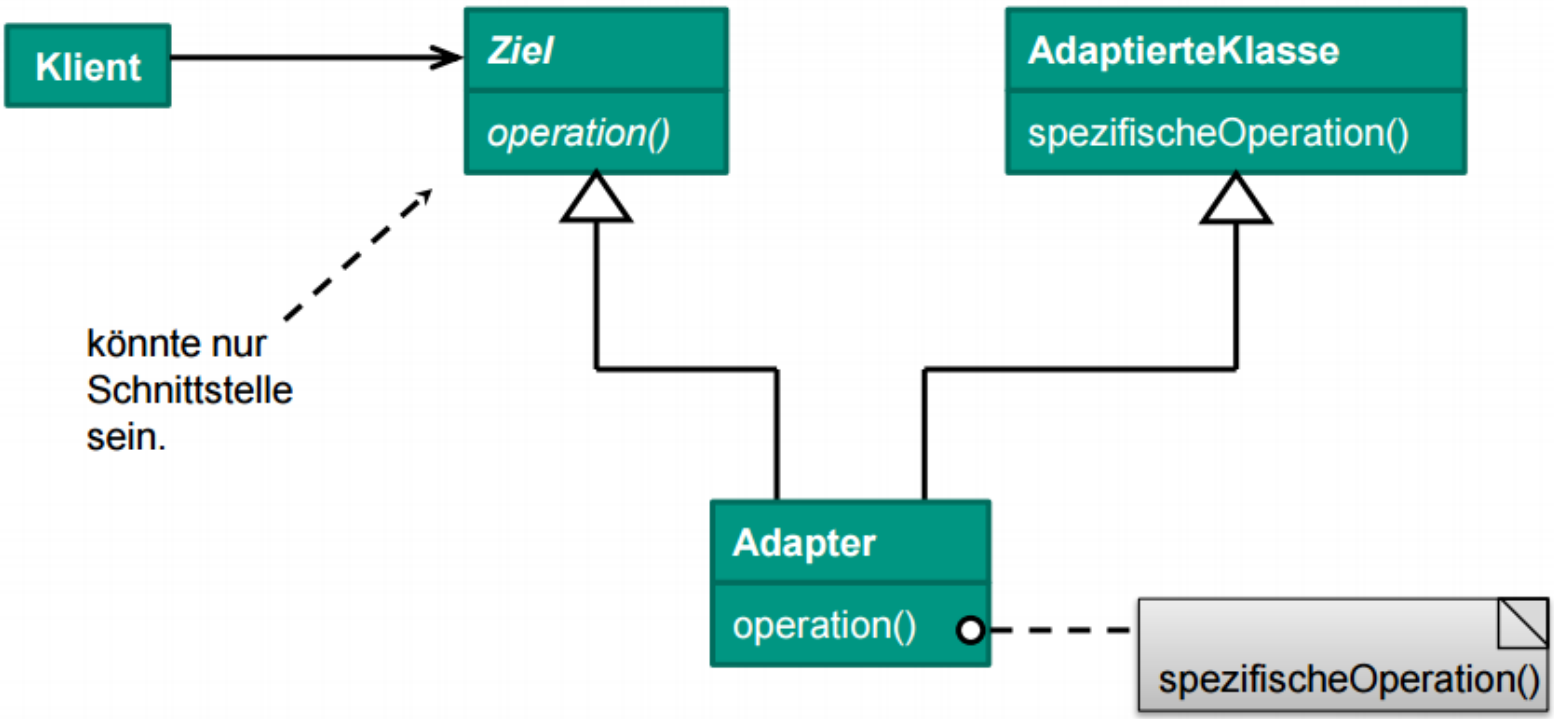
Adapter (Objektadapter)



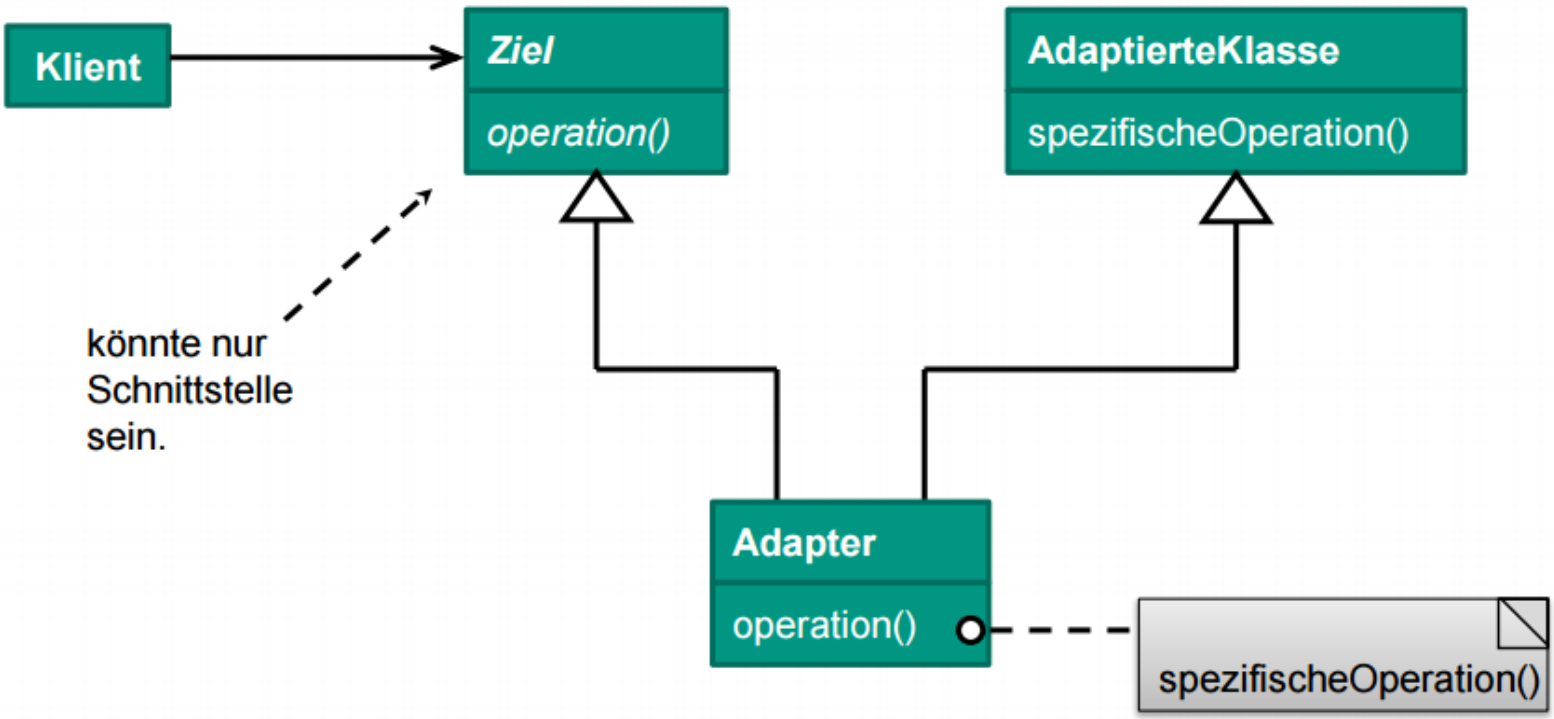
Preisfrage: Wo ist die Entkopplung?

der Klient ist von der adaptierten Klasse (insb. von seiner Schnittstelle) entkoppelt

Adapter - Alternative (Klassenadapter)

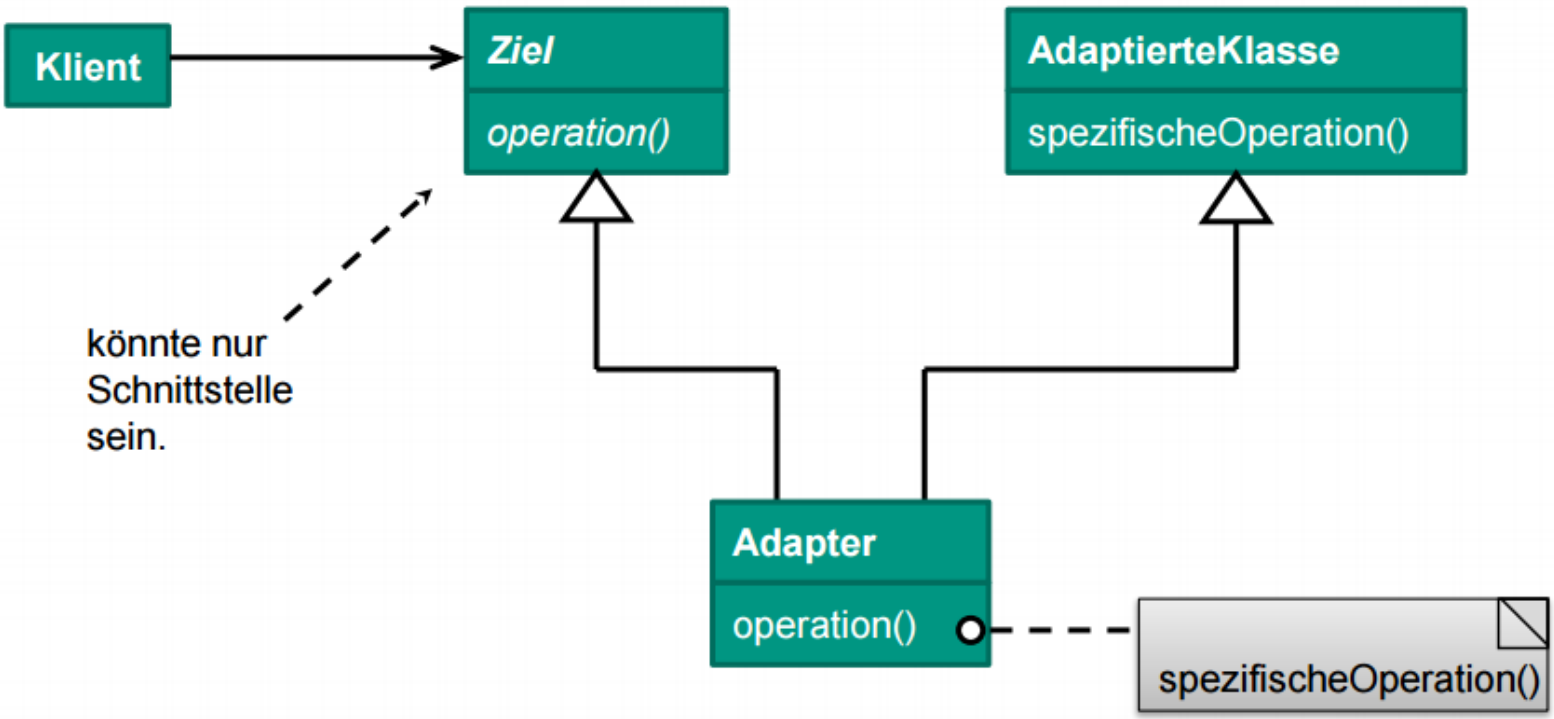


Adapter - Alternative (Klassenadapter)



Was für ein Problem bekommt ihr, wenn ihr das auf einem ÜB implementieren müsst?

Adapter - Alternative (Klassenadapter)



Was für ein Problem bekommt ihr, wenn ihr das auf einem ÜB implementieren müsst?

$\Longrightarrow$ keine Mehrfachvererbung in Java

Adapter - Beispiel

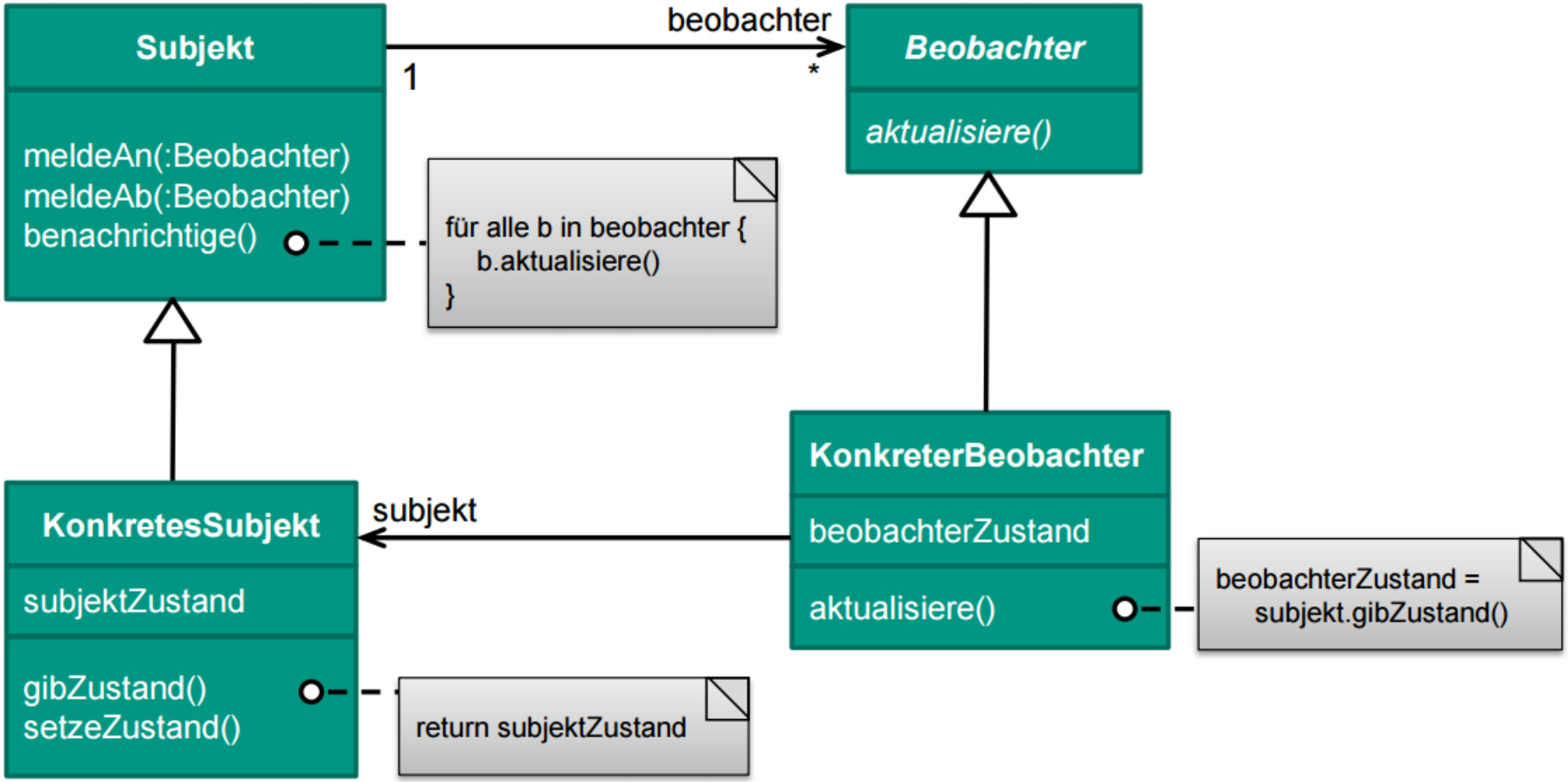
Karlsruher Institut für Technologie

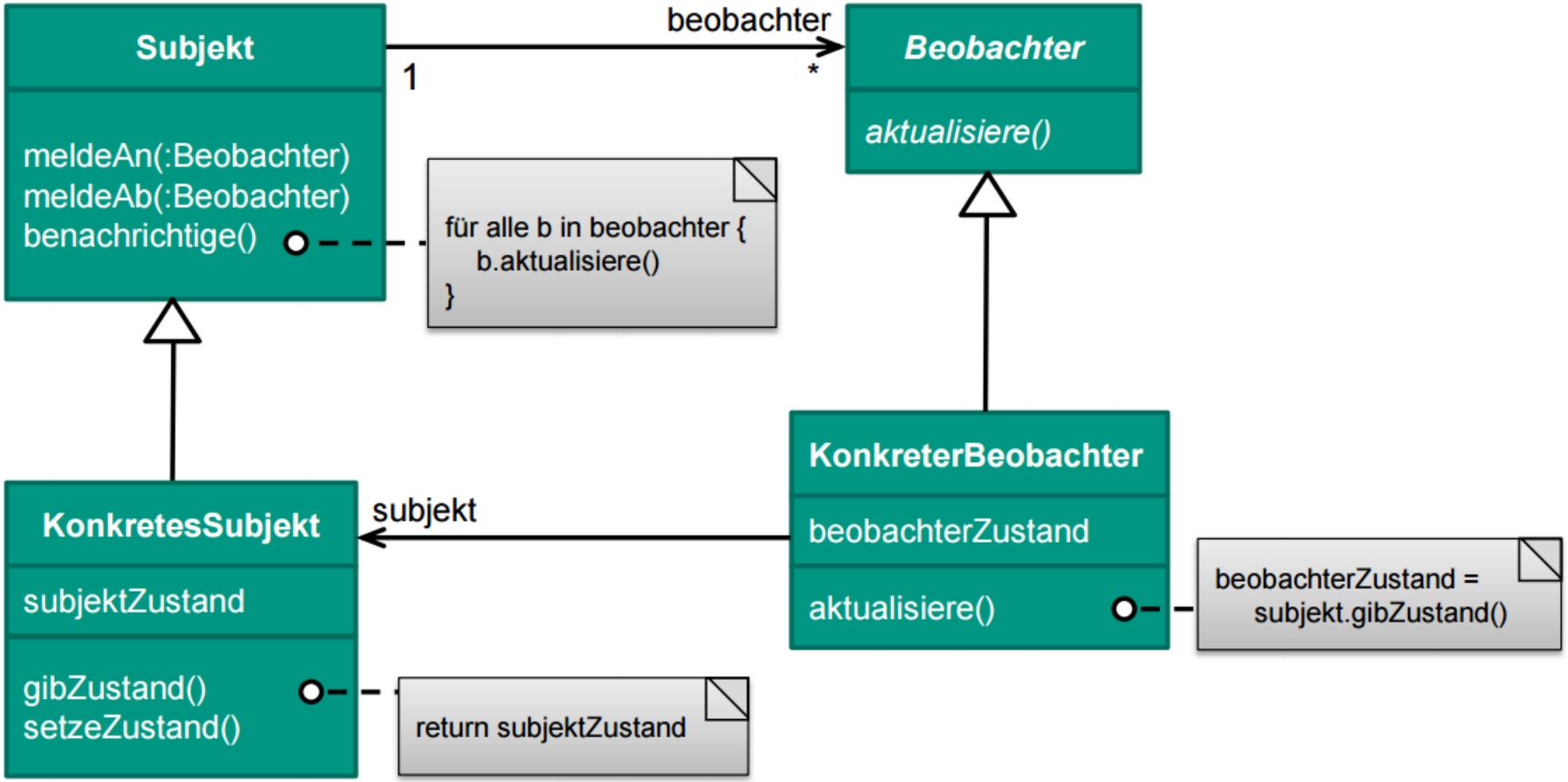
Beispiel

Beobachter/Observer

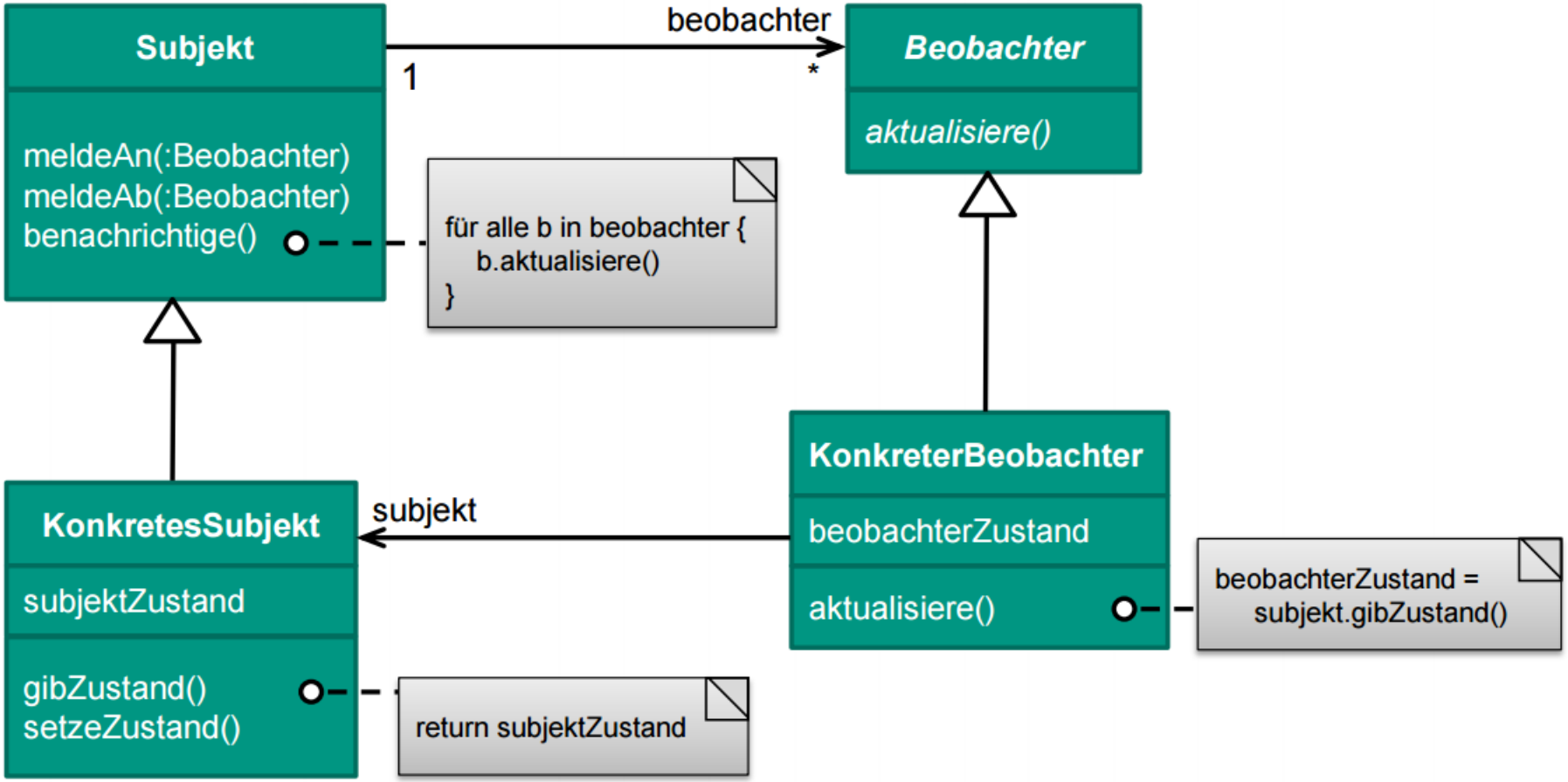
Problem

- ein Subjekt, viele Beobachter
- Subjekt ändert Zustand $\Longrightarrow$ Beobachter machen “irgendwas”


Subjekt
melde An(:Beobachter)
melde Ab(:Beobachter)
benachrichtige()
beobachter
1
*
Beobachter
aktualisiere()
für alle b in beobachter {
b.aktualisiere()
}
KonkreterBeobachter
beobachterZustand
aktualisiere()
beobachterZustand =
subjekt.gibZustand()
subjekt
KonkretesSubjekt
subjektZustand
gibZustand()
setzeZustand()
return subjektZustand

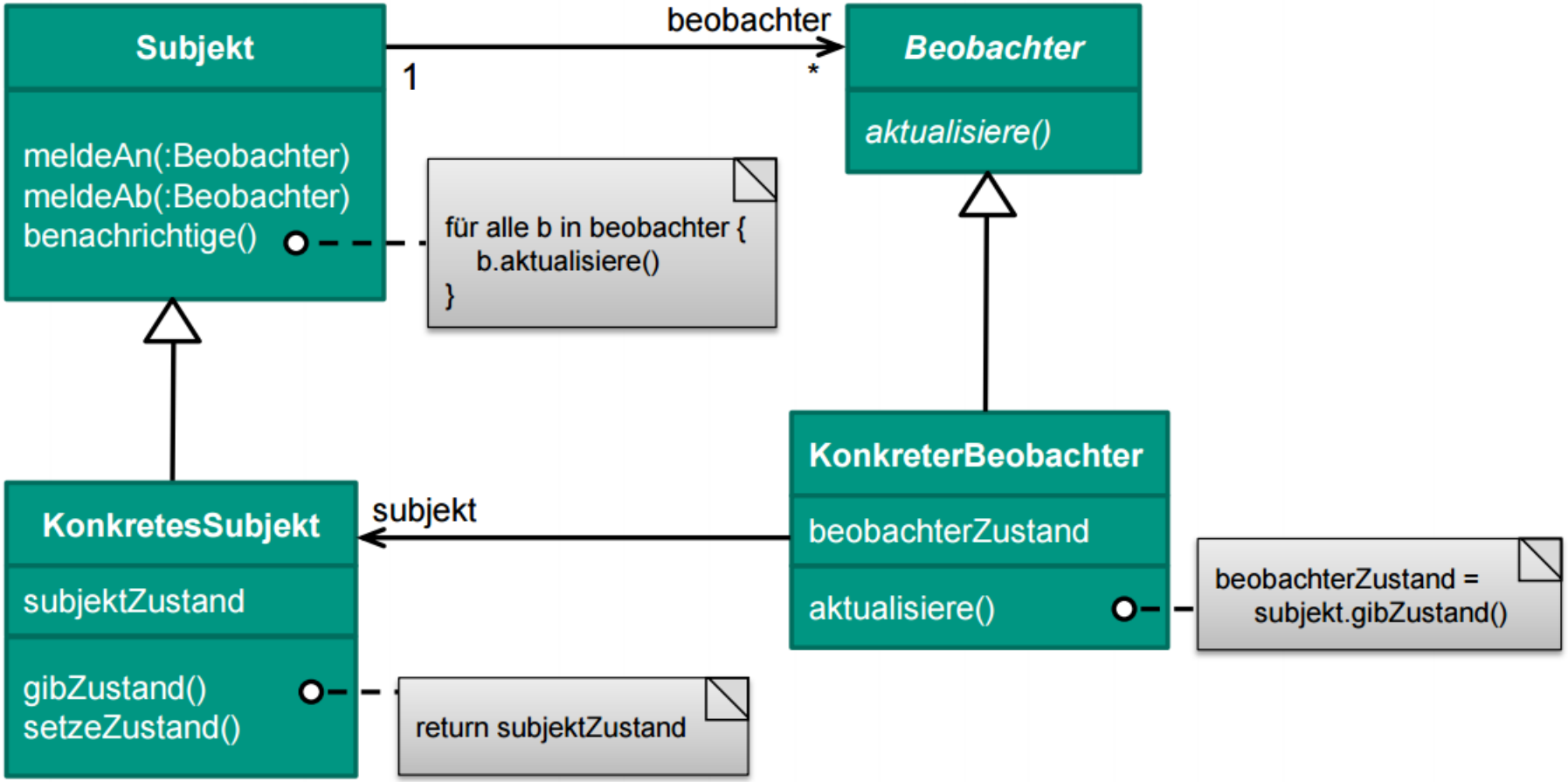


Entkopplung?



Entkopplung?

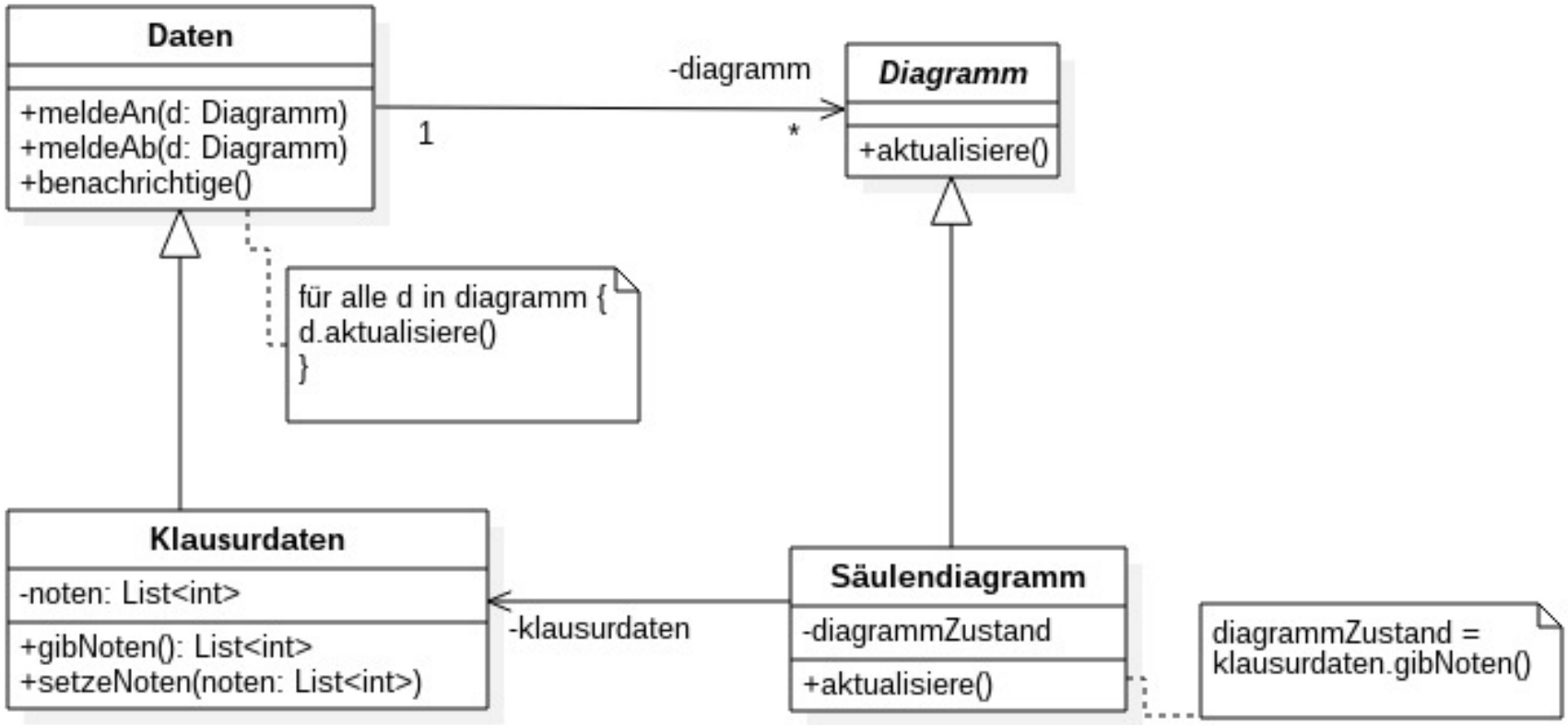
- jeder Beobachter definiert, was bei Benachrichtigung passiert, Subjekt kriegt davon nichts mit



Entkopplung?

- jeder Beobachter definiert, was bei Benachrichtigung passiert, Subjekt kriegt davon nichts mit
- zur Laufzeit änderbar: Anzahl der Beobachter

Beobachter/Observer: am Beispiel



Beobachter=Listener in Java Swing

Beobachter=Listener in Java Swing

Beobachter für einen `JButton btn`

```
btn.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        System.out.print("clicked");
    }
});
```

Beobachter=Listener in Java Swing

Das Gleiche mit Lambdas

```
btn.addActionListener(e -> System.out.print("clicked"));
```

Iterator

Problem

- wollen über Datenstruktur iterieren + Operationen ausführen

Iterator

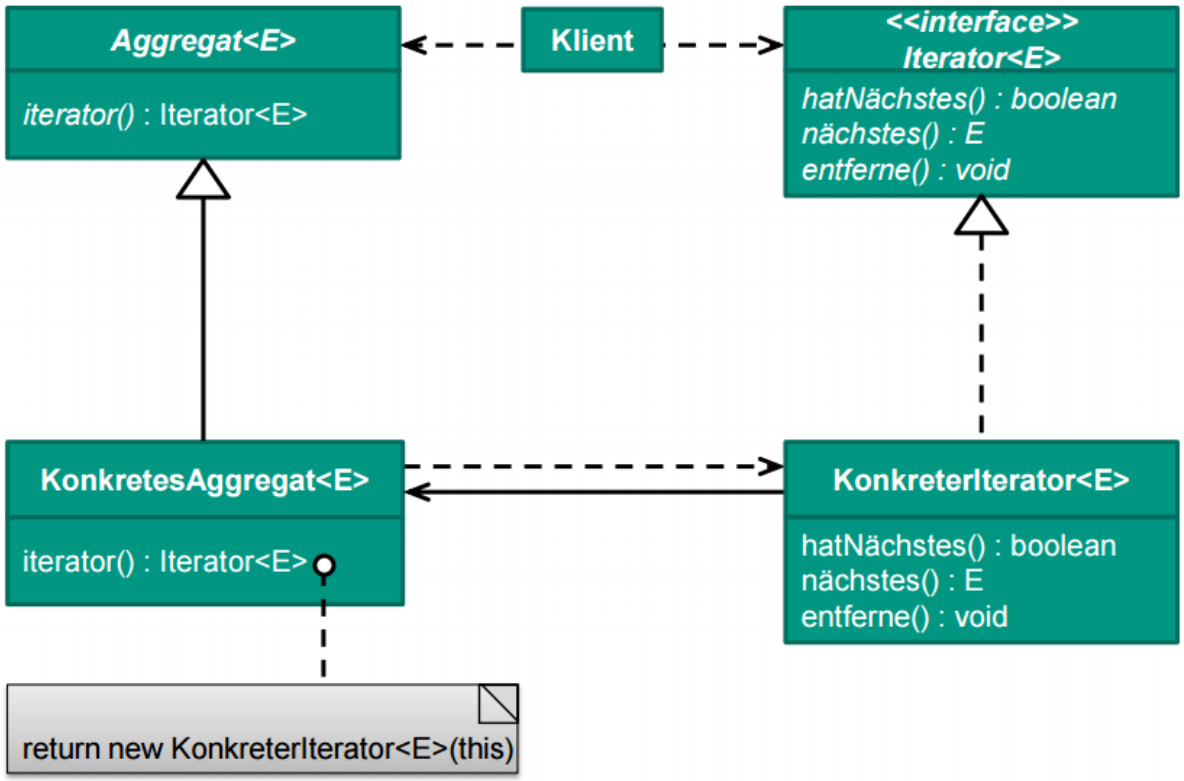
Problem

- wollen über Datenstruktur iterieren + Operationen ausführen
- das Ganze ohne Kentniss des internen Aufbaus der Datenstruktur

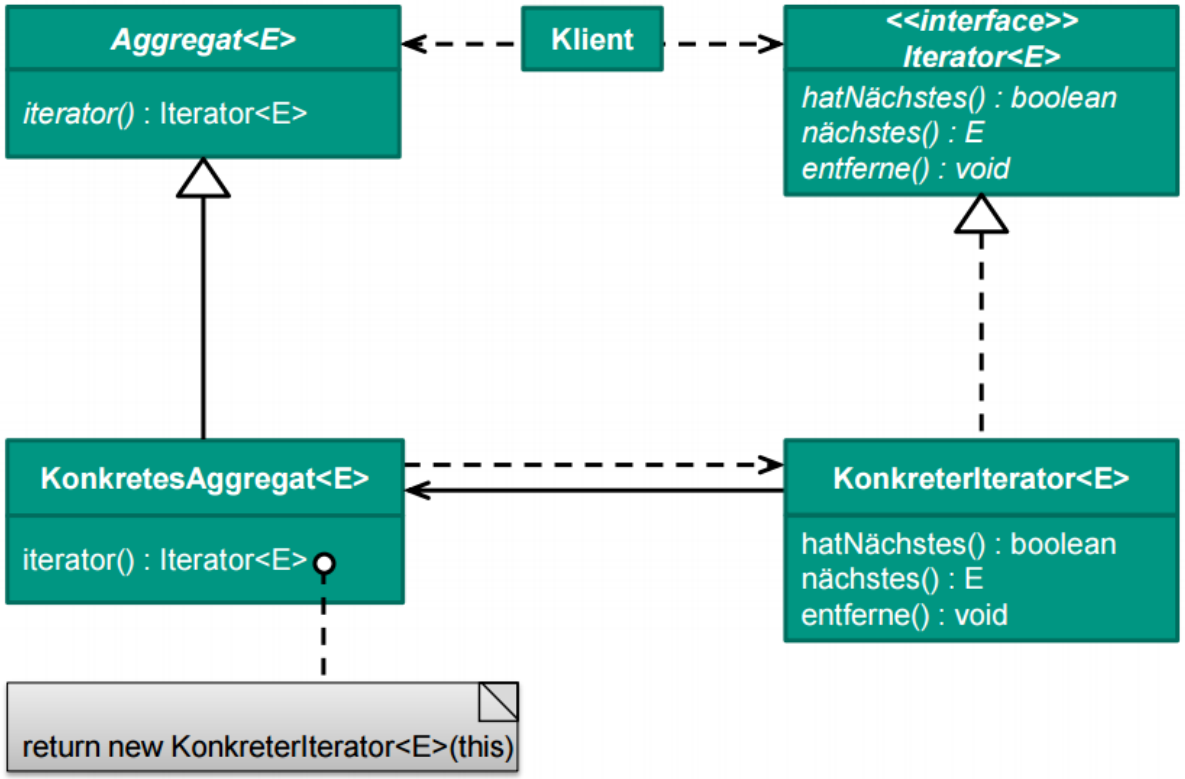
Iterator

Problem

- wollen über Datenstruktur iterieren + Operationen ausführen
- das Ganze ohne Kentniss des internen Aufbaus der Datenstruktur

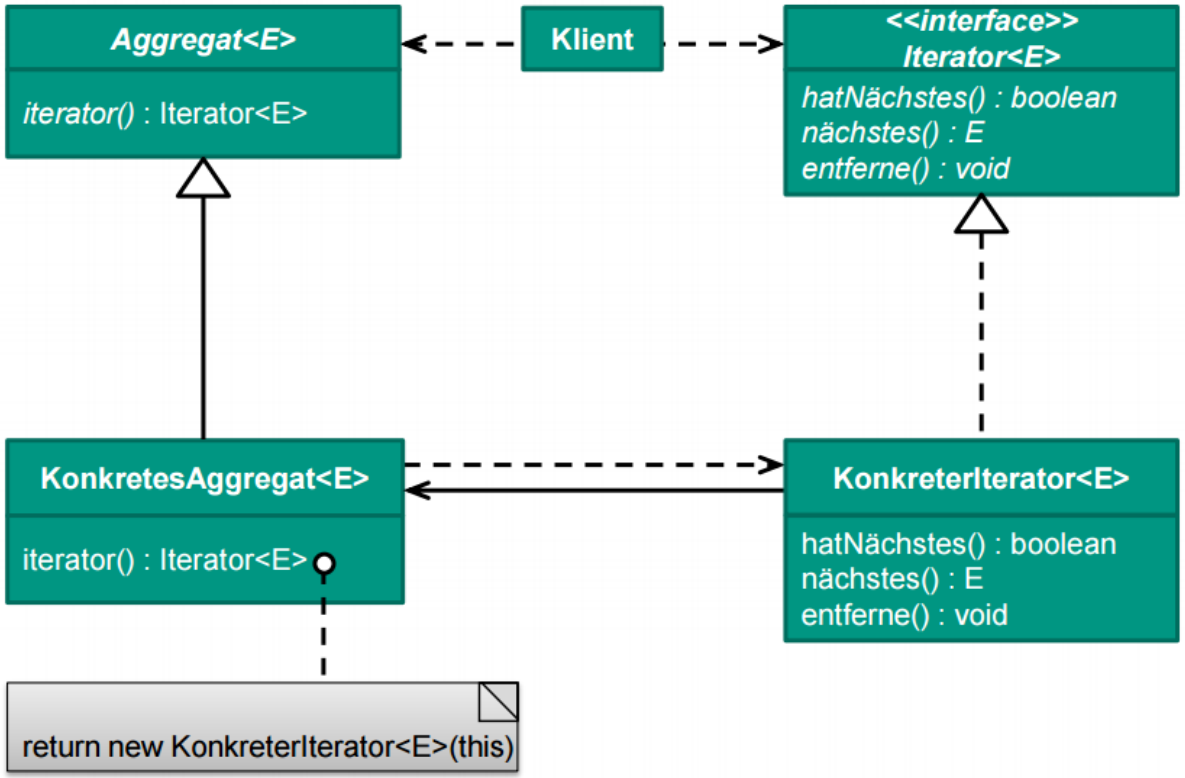


Iterator



Entkopplung?

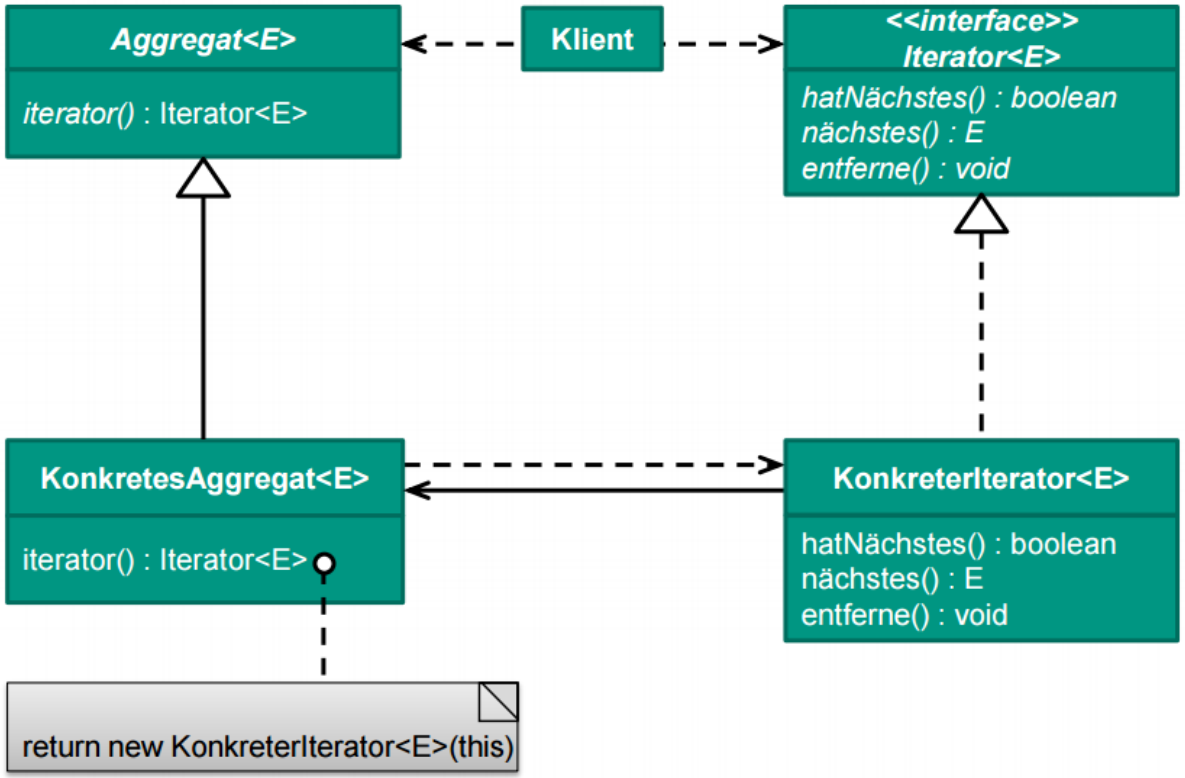
Iterator



Entkopplung?

- Klient benutzt nur Methoden der Schnittstelle auf dem konkreten Iterator
  $\Longrightarrow$ Implementierung austauschbar

Iterator



In Java

- `List l = new ArrayList(); l.iterator();`
- `for(String s : stringList) //...`

Quiz (Ankreuzaufgaben aus Klausuren)

Wahr oder falsch?

- Klienten können mithilfe des Iterator-Musters Sammlungen von Objekten und einzelne Objekte auf die gleiche Weise behandeln.

Quiz (Ankreuzaufgaben aus Klausuren)

Wahr oder falsch?

- Klienten können mithilfe des Iterator-Musters Sammlungen von Objekten und einzelne Objekte auf die gleiche Weise behandeln. ✗

Quiz (Ankreuzaufgaben aus Klausuren)

Wahr oder falsch?

- Klienten können mithilfe des Iterator-Musters Sammlungen von Objekten und einzelne Objekte auf die gleiche Weise behandeln. ✗
- Das Entwurfsmuster Iterator ist den Variantenmustern zuzuordnen.

Quiz (Ankreuzaufgaben aus Klausuren)

Wahr oder falsch?

- Klienten können mithilfe des Iterator-Musters Sammlungen von Objekten und einzelne Objekte auf die gleiche Weise behandeln. ✗
- Das Entwurfsmuster Iterator ist den Variantenmustern zuzuordnen. ✗

Stellvertreter

Problem

- wollen Zugriff auf ein Objekt kontrollieren, ohne seine Klasse zu ändern

Stellvertreter

Problem

- wollen Zugriff auf ein Objekt kontrollieren, ohne seine Klasse zu ändern

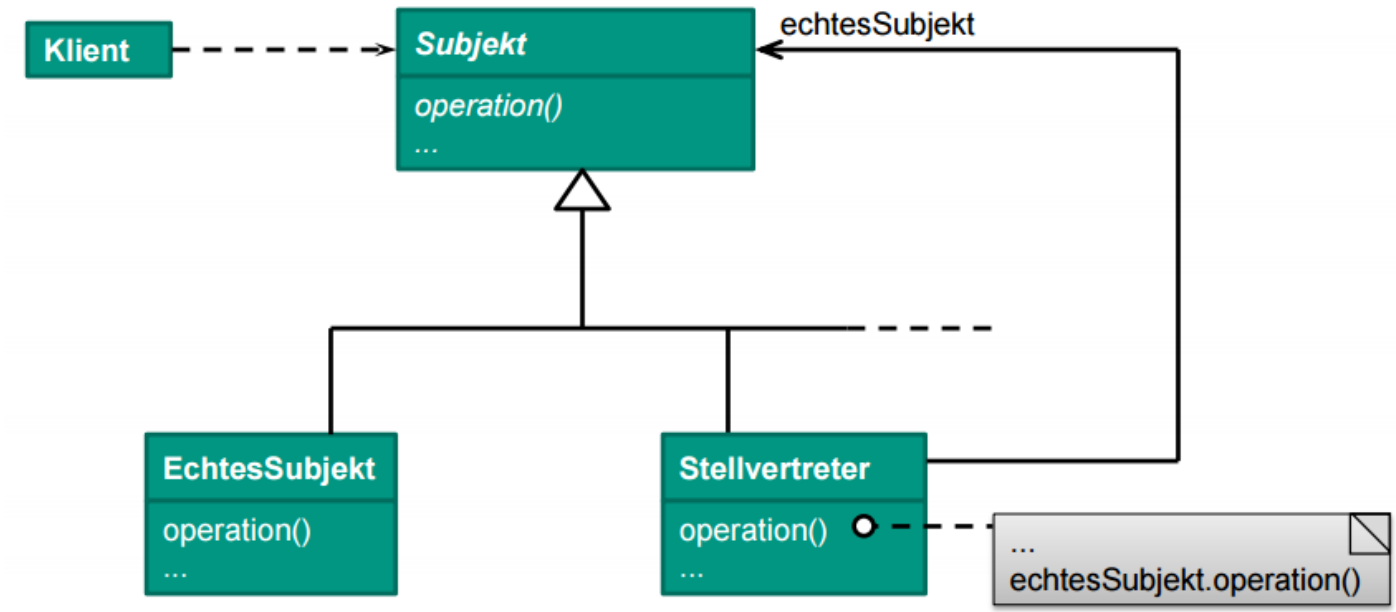
  $\Longrightarrow$ Stellvertreter macht Zugriffskontrolle

Stellvertreter

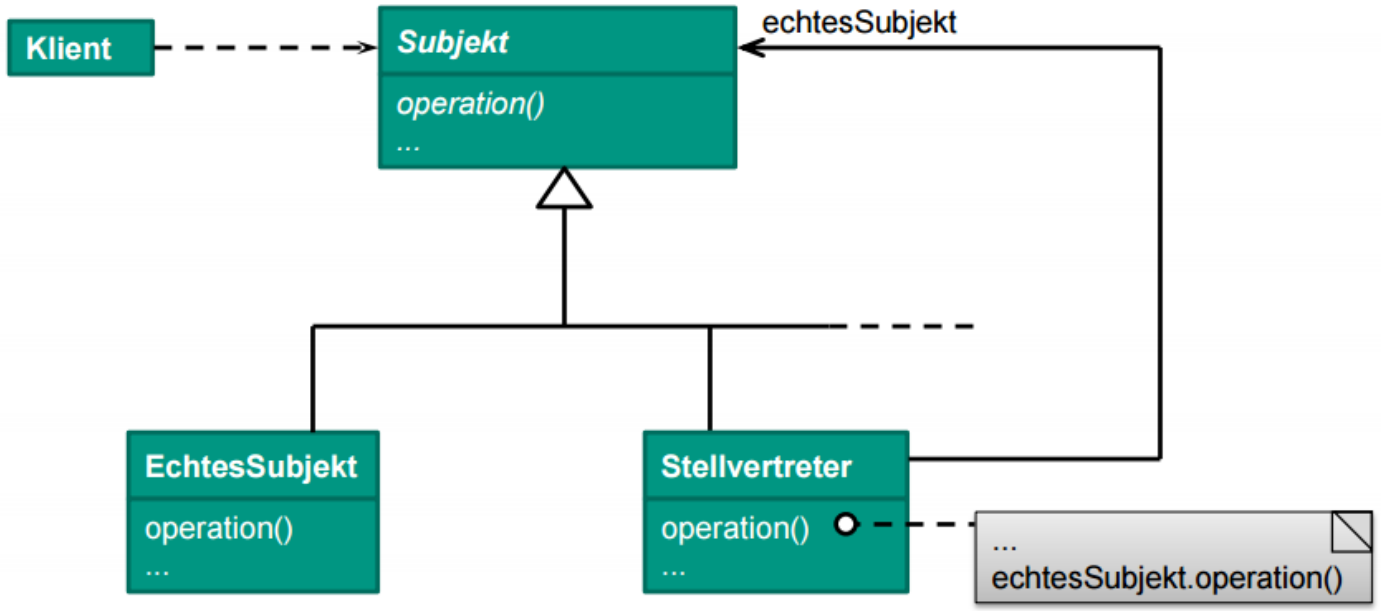
Problem

- wollen Zugriff auf ein Objekt kontrollieren, ohne seine Klasse zu ändern

  $\Longrightarrow$ Stellvertreter macht Zugriffskontrolle

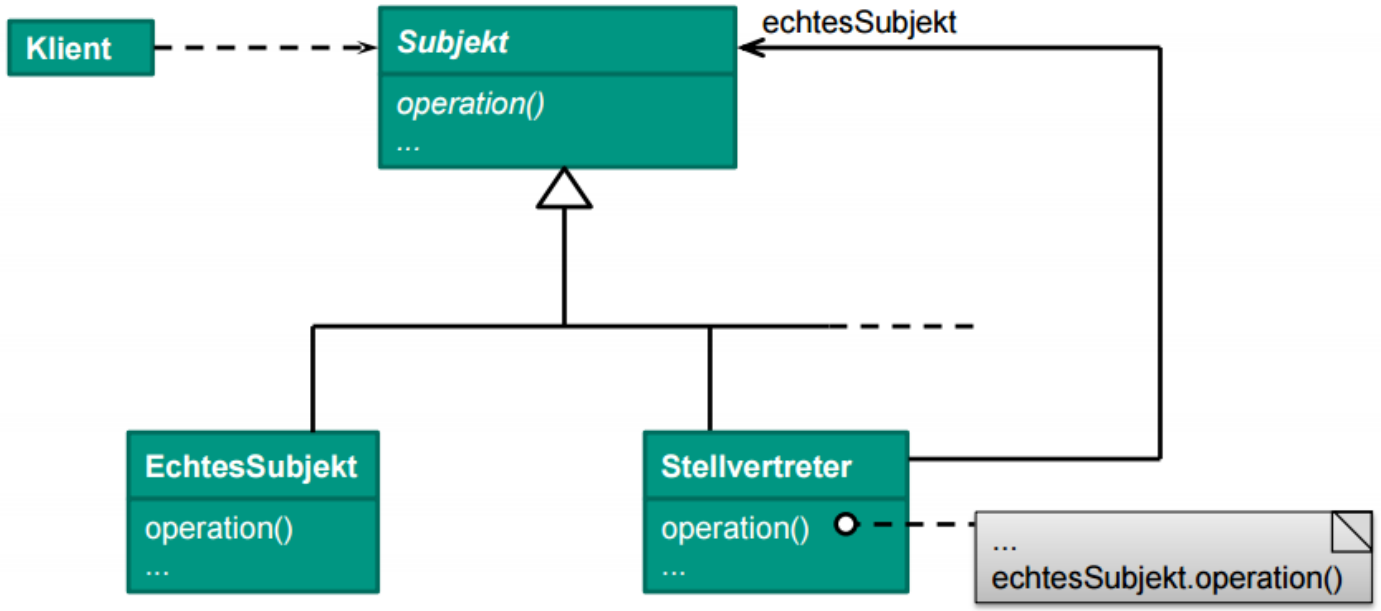


Stellvertreter



Entkopplung?

Stellvertreter



Entkopplung?

- Klient hat keinen direkten Zugriff auf das echte Subjekt
- Zugriffskontrolle und Funktionalität getrennt

Adapter vs. Stellvertreter

Adapter

Schnittstelle wird angepasst, Verhalten bleibt gleich

Stellvertreter

Schnittstelle bleibt gleich, Verhalten wird angepasst (z.B. Zugriffskontrolle)

Bis dann! (dann := 25.06.19)