

# Softwaretechnik 1 - 3. Tutorium

Tutorium 03

Felix Bachmann | 12.06.2017

KIT - INSTITUT FÜR PROGRAMMSTRUKTUREN UND DATENORGANISATION (IPD)



- 1 Orga
- 2 Motivation
- 3 Entwurfsmuster
- 4 Adapter
- 5 Beobachter
- 6 Iterator
- 7 Stellvertreter
- 8 Vermittler
- 9 Klausuraufgabe
- 10 Tipps

## 2. Übungsblatt Statistik

## Allgemein

- verspätete Abgaben bekomme ich erst beim jeweils nächsten Tutorsmeeting  
⇒ Rückgabe dauert länger; gibt keine Punkte, nur grobe Korrektur

## Aufgabe 1 (Plug-In-Architektur): Ø 2,25 bzw. 3,86 von 5

- keine riesigen switch-cases/if-else Anweisungen in `compareTo()` benutzen  
⇒ Erweiterbarkeit wird dadurch eingeschränkt

## Aufgabe 1 (Plug-In-Architektur): Ø 2,25 bzw. 3,86 von 5

- keine riesigen switch-cases/if-else Anweisungen in compareTo() benutzen
  - ⇒ Erweiterbarkeit wird dadurch eingeschränkt
  - ⇒ Java vergleicht Enum-Elemente anhand ihrer Position in dem Enum
- orientiert euch nicht am JMJRST-Stil

Aufgabe 2 (Plug-In): Ø 1,79 bzw. 3,58 von 4

- keine :D

Aufgabe 3 (iMage-Bundle): Ø 0,44 bzw. 1,75 von 2

- 6 Abgaben ...

## Aufgabe 4 (Aktivitätsdiagramme (Geometrfy)): Ø von 10





## Aufgabe 5 (Sequenzdiagramm (main-Methode)): Ø von 5



## Aufgabe 6 (Substitutionsprinzip): 0 von 3



- die ersten 2 Phasen des Wasserfallmodells sind geschafft

- die ersten 2 Phasen des Wasserfallmodells sind geschafft  
⇒ Welche waren das nochmal?

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

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

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

- die ersten 2 Phasen des Wasserfallmodells sind geschafft
  - ⇒ Welche waren das nochmal? Planung, Definition!
  - ⇒ Dokumente? Lastenheft, Pflichtenheft (+ andere. . .)
- jetzt: Entwurf!



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



**Entwurfsprozess**



Softwarearchitektur

Softwarearchitektur ist Grundlage für Implementierung!

- Definition: **Was** ist zu implementieren?

- Definition: **Was** ist zu implementieren?
- Entwurf: **Wie** ist das System zu implementieren?

# Empfehlenswerte Literatur (wirklich!)

knapp 700 Seiten

⇒ als interaktives Nachschlagewerk, falls man bestimmte Muster nicht versteht



## Entwurfsmuster

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

## Entwurfsmuster

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

- schematische Klassendiagramme zur Lösung von häufig auftretenden Problemen

## Entwurfsmuster

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

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

## Entwurfsmuster

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

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





- erleichtern Kommunikation

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

## Geheimnis- / Kapselungsprinzip

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

## Geheimnis- / Kapselungsprinzip

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

Sinn?

## Geheimnis- / Kapselungsprinzip

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

Sinn?  $\implies$  Änderungen ohne Risiko durchführen

## Geheimnis- / Kapselungsprinzip

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

Sinn?  $\implies$  Änderungen ohne Risiko durchführen  
Beispiel?

## Geheimnis- / Kapselungsprinzip

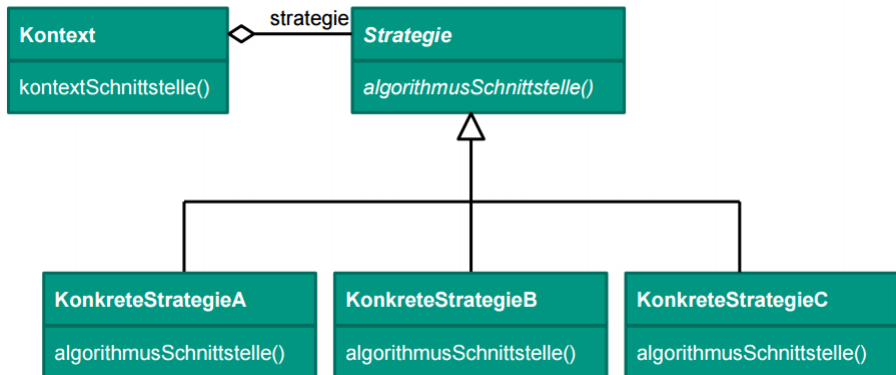
Jedes Modul verbirgt eine wichtige Entwurfsentscheidung hinter einer wohldefinierten Schnittstelle, die sich bei einer Änderung der Entscheidung nicht mit ändert.

Sinn?  $\implies$  Änderungen ohne Risiko durchführen

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

# Vorgriff: Entwurfsmuster Strategie

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





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

Wahr oder falsch?

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

Wahr oder falsch?

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

wahr

Wahr oder falsch?

- Das Entwurfsmuster Strategie bietet die Möglichkeit, eine Klasse mit einer von mehreren möglichen Verhaltensweisen zu konfigurieren.  
**wahr**
- Das Strategiemuster erfüllt das Geheimnisprinzip, indem es Datenstrukturen, die in einer konkreten Strategie enthalten sind, vor dem Klienten verbirgt.

Wahr oder falsch?

- Das Entwurfsmuster Strategie bietet die Möglichkeit, eine Klasse mit einer von mehreren möglichen Verhaltensweisen zu konfigurieren. **wahr**
- Das Strategiemuster erfüllt das Geheimnisprinzip, indem es Datenstrukturen, die in einer konkreten Strategie enthalten sind, vor dem Klienten verbirgt. **wahr**

Wahr oder falsch?

- Das Entwurfsmuster Strategie bietet die Möglichkeit, eine Klasse mit einer von mehreren möglichen Verhaltensweisen zu konfigurieren.  
**wahr**
- Das Strategiemuster erfüllt das Geheimnisprinzip, indem es Datenstrukturen, die in einer konkreten Strategie enthalten sind, vor dem Klienten verbirgt. **wahr**
- Das Muster Strategie kapselt austauschbares Verhalten und verwendet Delegierung, um zu entscheiden, welches Verhalten verwendet wird.

Wahr oder falsch?

- Das Entwurfsmuster Strategie bietet die Möglichkeit, eine Klasse mit einer von mehreren möglichen Verhaltensweisen zu konfigurieren. **wahr**
- Das Strategiemuster erfüllt das Geheimnisprinzip, indem es Datenstrukturen, die in einer konkreten Strategie enthalten sind, vor dem Klienten verbirgt. **wahr**
- Das Muster Strategie kapselt austauschbares Verhalten und verwendet Delegation, um zu entscheiden, welches Verhalten verwendet wird. **wahr**

Wahr oder falsch?

- Das Entwurfsmuster Strategie bietet die Möglichkeit, eine Klasse mit einer von mehreren möglichen Verhaltensweisen zu konfigurieren.  
**wahr**
- Das Strategiemuster erfüllt das Geheimnisprinzip, indem es Datenstrukturen, die in einer konkreten Strategie enthalten sind, vor dem Klienten verbirgt. **wahr**
- Das Muster Strategie kapselt austauschbares Verhalten und verwendet Delegierung, um zu entscheiden, welches Verhalten verwendet wird. **wahr**
- Das Hinzufügen einer neuen konkreten Strategie erfordert keine Änderung existierender konkreter Strategien.



Wahr oder falsch?

- Das Entwurfsmuster Strategie bietet die Möglichkeit, eine Klasse mit einer von mehreren möglichen Verhaltensweisen zu konfigurieren. **wahr**
- Das Strategiemuster erfüllt das Geheimnisprinzip, indem es Datenstrukturen, die in einer konkreten Strategie enthalten sind, vor dem Klienten verbirgt. **wahr**
- Das Muster Strategie kapselt austauschbares Verhalten und verwendet Delegierung, um zu entscheiden, welches Verhalten verwendet wird. **wahr**
- Das Hinzufügen einer neuen konkreten Strategie erfordert keine Änderung existierender konkreter Strategien. **wahr**

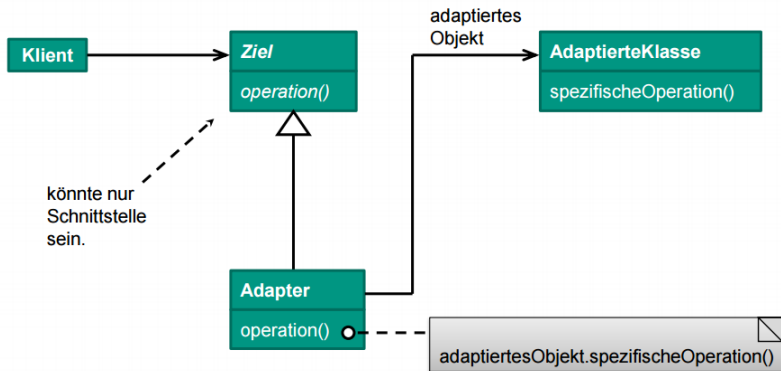
- übergeordnetes Ziel: System in Teile aufspalten, die unabhängig voneinander sind  
⇒ Teile austauschbar bzw. veränderbar

## Problem

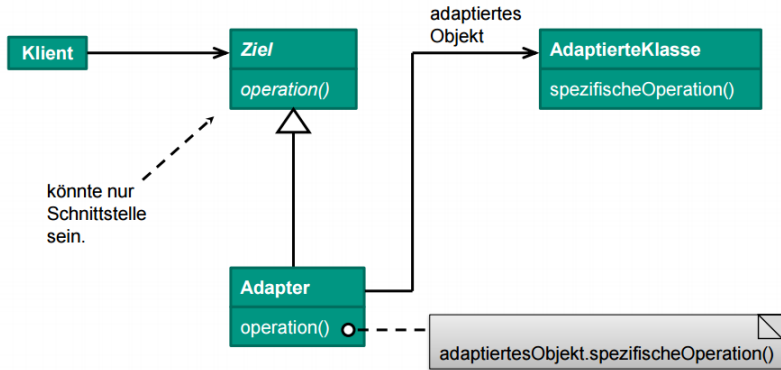
- Klassen mit inkompatiblen Schnittstellen, die wir aber zusammen benutzen wollen
- Schnittstellen nicht änderbar (z.B. externe Bibliotheken)

## Problem

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



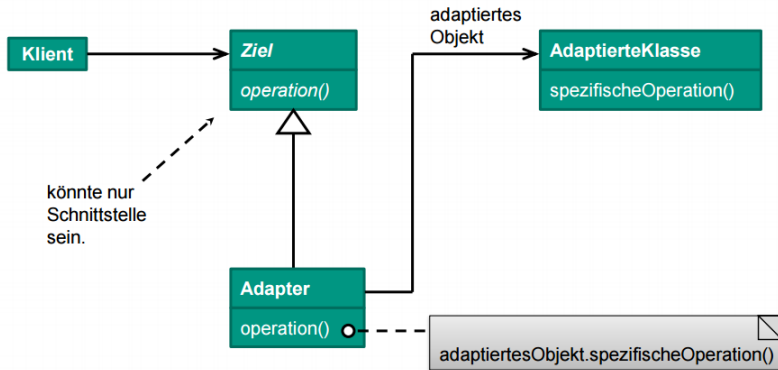
# Adapter (Objektadapter)



Wir sind bei Entkopplung-Mustern, Preisfrage:

Wo ist hier die Entkopplung?

# Adapter (Objektadapter)

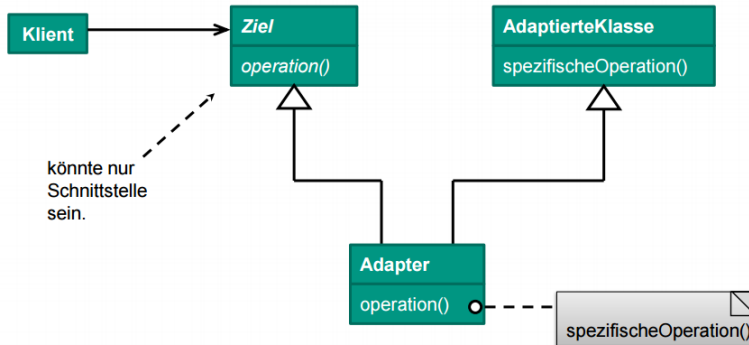


Wir sind bei Entkopplung-Mustern, Preisfrage:

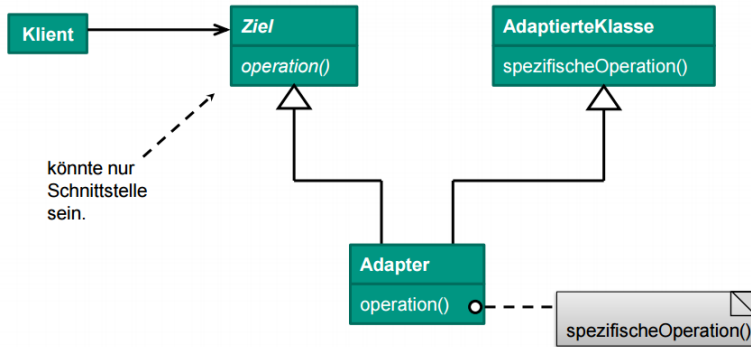
Wo ist hier die Entkopplung?

der Klient ist von der adaptierten Klasse entkoppelt  $\implies$  austauschbar

# Adapter - Alternative (Klassenadapter)



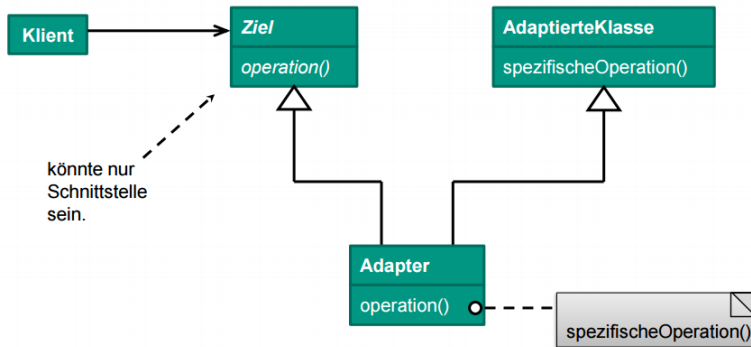
# Adapter - Alternative (Klassenadapter)



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



# Adapter - Alternative (Klassenadapter)

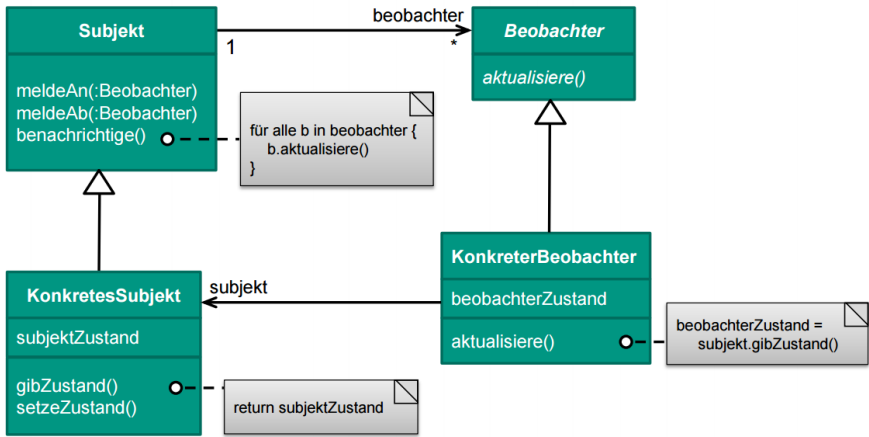


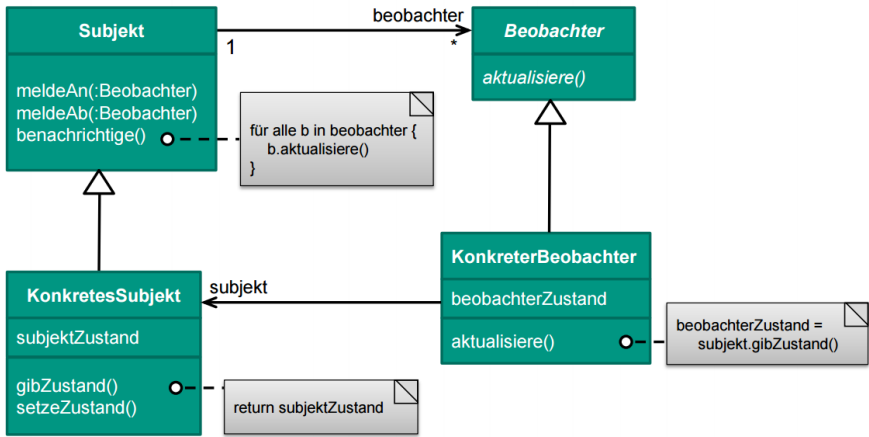
Was für ein Problem bekommt ihr, wenn ihr das auf einem ÜB implementieren müsst?

⇒ keine Mehrfachvererbung in Java!

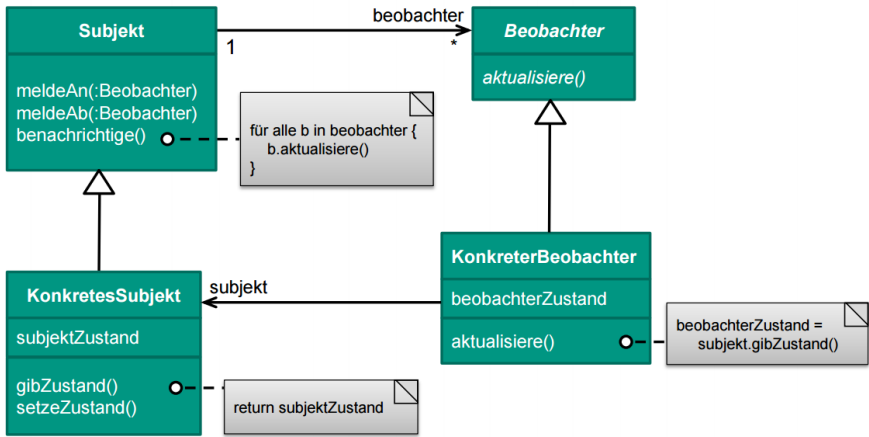
## Problem

- ein Subjekt, viele Beobachter
- Subjekt ändert Zustand  $\implies$  Beobachter machen "irgendwas"



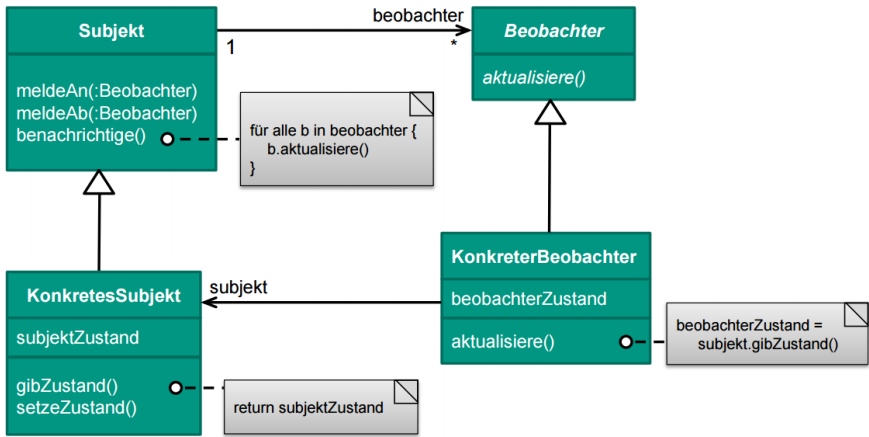


## Entkopplung?



## Entkopplung?

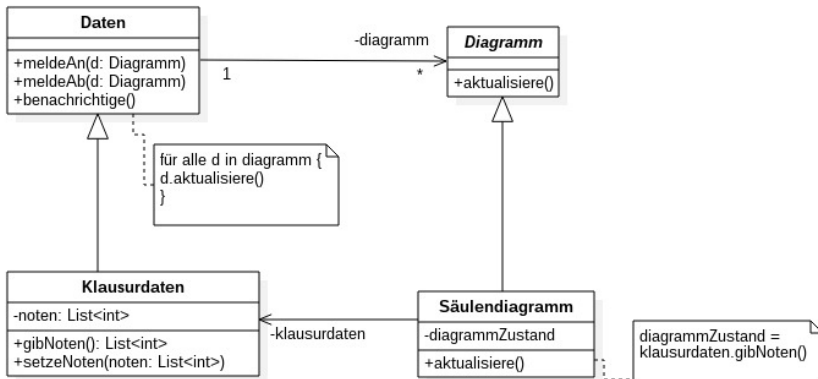
- jeder Beobachter definiert, was bei Benachrichtigung passiert, Subjekt kriegt davon nichts mit



## Entkopplung?

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

# Beobachter/Observer: am Beispiel



## Problem

- wollen über Datenstruktur iterieren + Operationen ausführen  
⇒ Hinzufügen, Löschen...

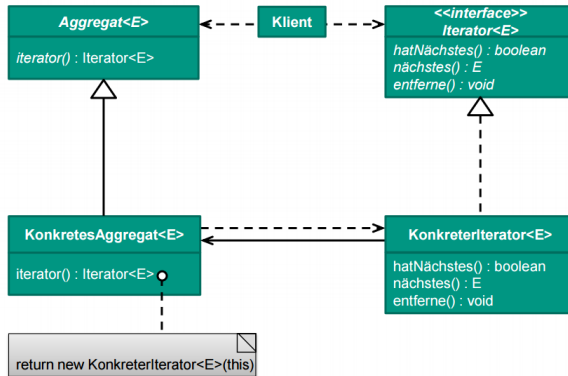


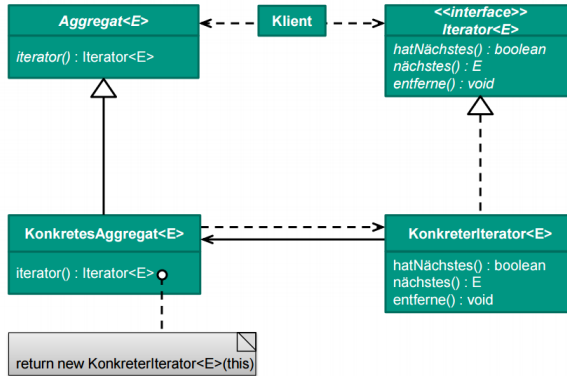
## Problem

- wollen über Datenstruktur iterieren + Operationen ausführen  
⇒ Hinzufügen, Löschen. . .
- das Ganze ohne Kenntnis des internen Aufbaus der Datenstruktur

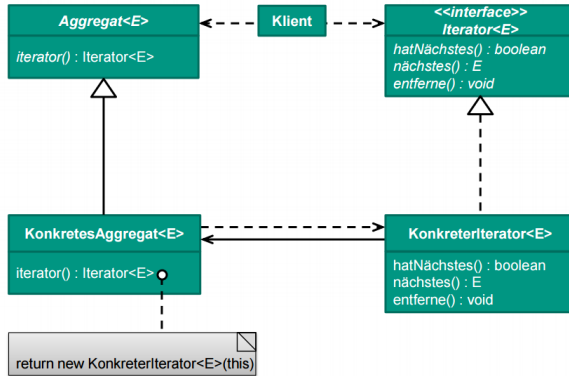
## Problem

- wollen über Datenstruktur iterieren + Operationen ausführen  
⇒ Hinzufügen, Löschen...
- das Ganze ohne Kenntnis des internen Aufbaus der Datenstruktur





## Entkopplung?



## Entkopplung?

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

Wahr oder falsch?

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

Wahr oder falsch?

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

falsch

Wahr oder falsch?

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

Wahr oder falsch?

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



## Problem

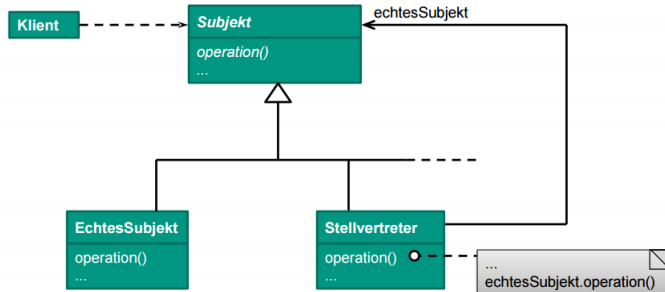
- wollen Zugriff auf ein Objekt kontrollieren, ohne seine Klasse zu ändern

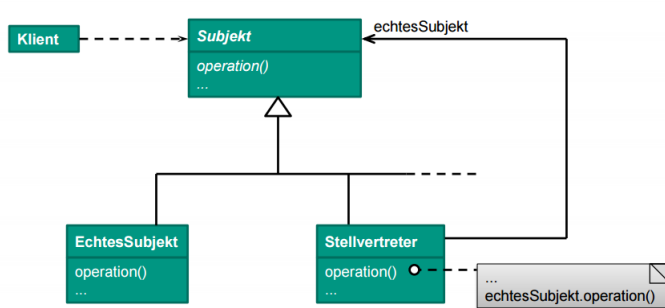
## Problem

- wollen Zugriff auf ein Objekt kontrollieren, ohne seine Klasse zu ändern  
⇒ Stellvertreter macht Zugriffskontrolle

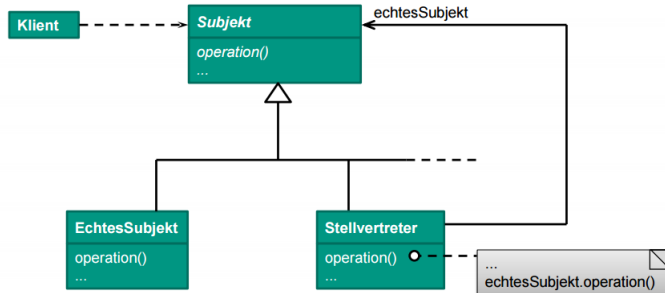
## Problem

- wollen Zugriff auf ein Objekt kontrollieren, ohne seine Klasse zu ändern  
⇒ Stellvertreter macht Zugriffskontrolle





## Entkopplung?



## Entkopplung?

- Klient hat keinen direkten Zugriff auf das echte Subjekt

## Problem

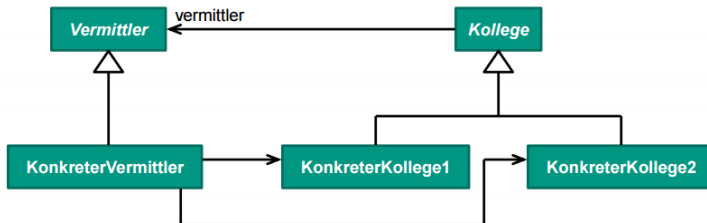
- mehrere voneinander abhängige Objekte

## Problem

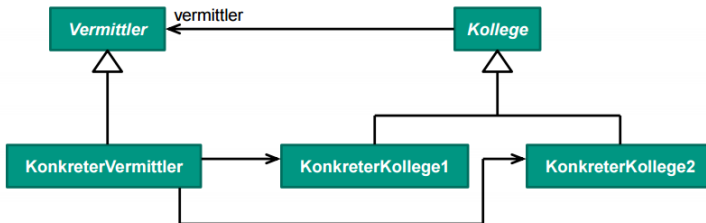
- mehrere voneinander abhängige Objekte  
⇒ Zustände der Objekte von anderen Zuständen abhängig

## Problem

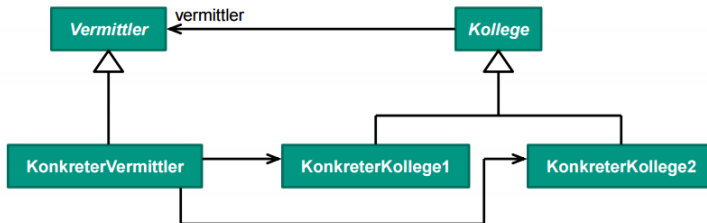
- mehrere voneinander abhängige Objekte  
⇒ Zustände der Objekte von anderen Zuständen abhängig





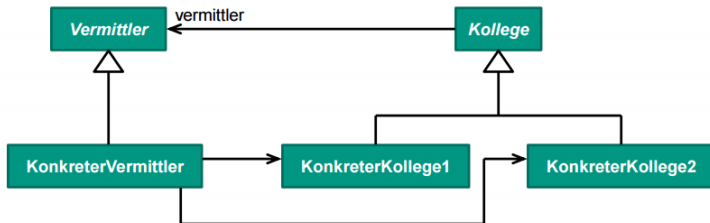


## Entkopplung?



## Entkopplung?

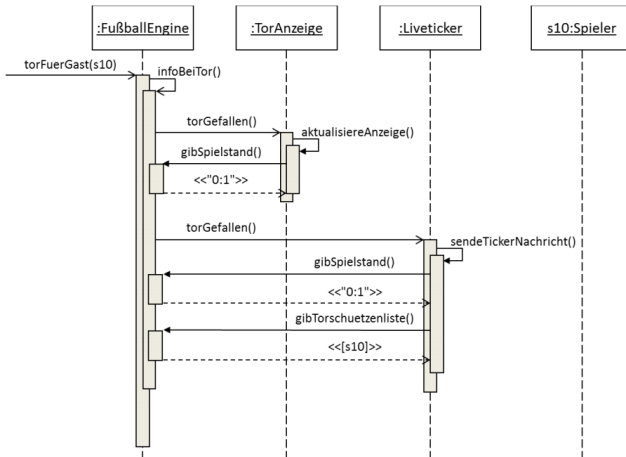
- Kollegen kennen sich nicht direkt



## Entkopplung?

- Kollegen kennen sich nicht direkt  
⇒ Hinzufügen eines Kollegen erfordert keine Änderung der alten Kollegen

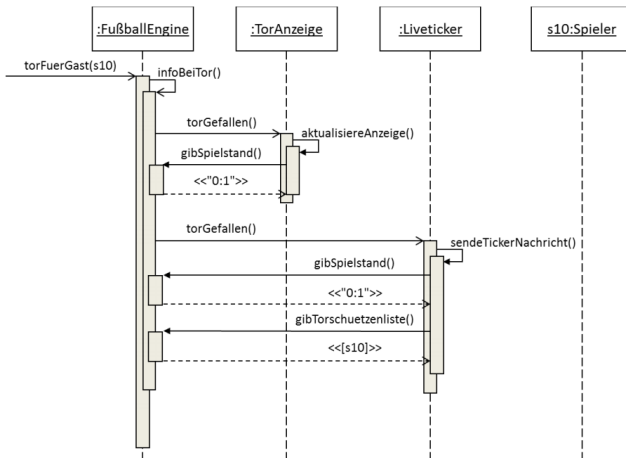
# Klausuraufgabe (Hauptklausur SS 2012)



## Aufgabe 1

Welches Entwurfsmuster erkennen Sie in diesem Diagramm?

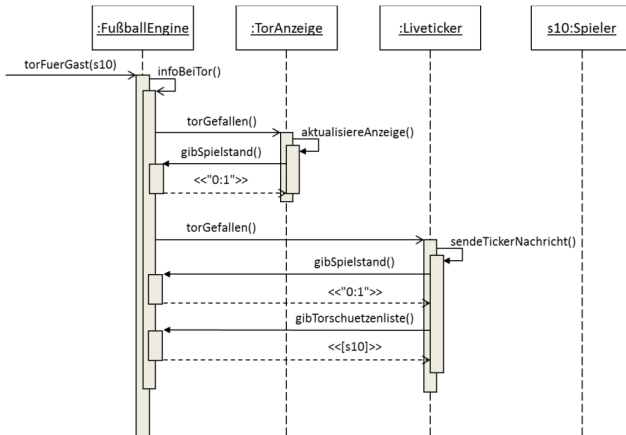
# Klausuraufgabe (Hauptklausur SS 2012)

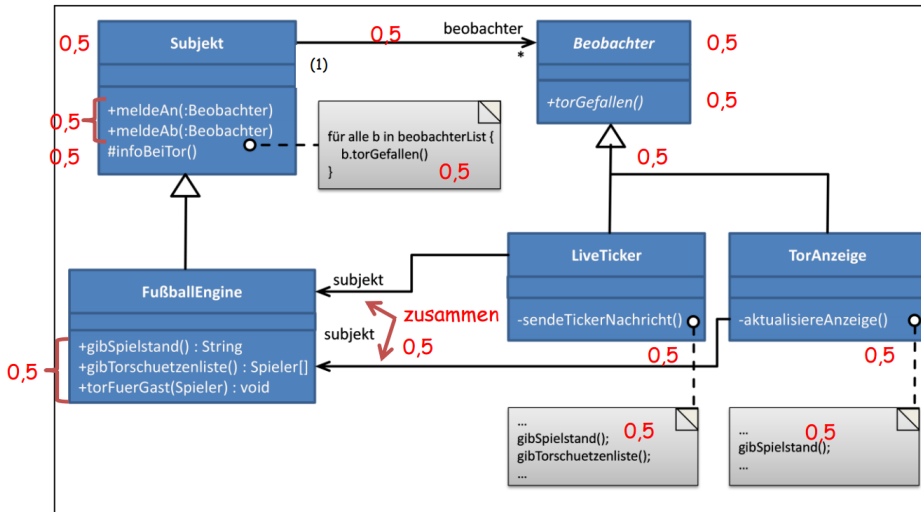


## Aufgabe 1

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.





## Aufgabe





Aufgabe



Aufgabe



## Aufgabe



Aufgabe



Aufgabe



## Abgabe

- Deadline am 21.6 um 12:00
- A{1,2,4} handschriftlich!

# Bis dann! (dann := 26.06.17)