

# Softwaretechnik 1 - 3. Tutorium

Tutorium 03

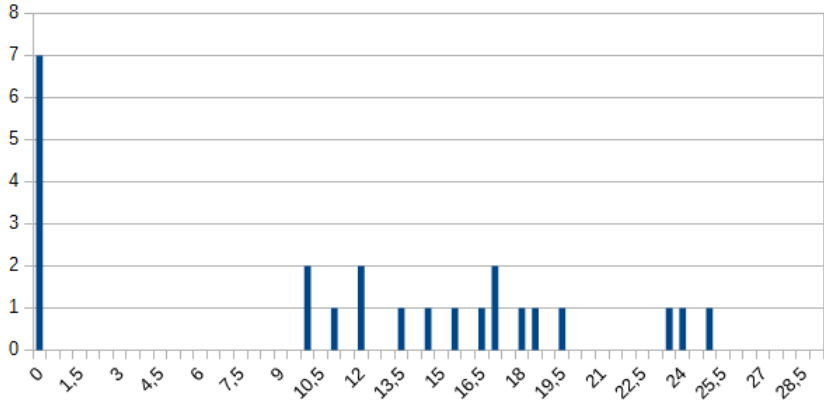
Felix Bachmann | 05.06.2018

KIT - INSTITUT FÜR PROGRAMMSTRUKTUREN UND DATENORGANISATION (IPD)



# 3. Übungsblatt Statistik

n=24



Ø 11,56 bzw. 16,32 von 28+1

## Allgemein

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

## Aufgabe 1 (Plug-In-Architektur)

## Aufgabe 1 (Plug-In-Architektur)

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

## Aufgabe 1 (Plug-In-Architektur)

- keine riesigen switch-cases/if-else Anweisungen in `compareTo()` benutzen
  - ⇒ Erweiterbarkeit wird dadurch eingeschränkt
  - ⇒ Java vergleicht Enum-Elemente anhand ihrer Position in dem Enum

## Aufgabe 1 (Plug-In-Architektur)

- 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)

- keine :D

## Aufgabe 3 (iMage-Bundle)

- 6 Abgaben ...



## Aufgabe 4 (Aktivitätsdiagramm)

## Aufgabe 4 (Aktivitätsdiagramm)

- denkt an die Rauten!

## Aufgabe 4 (Aktivitätsdiagramm)

- denkt an die Rauten!
- [Bedingung]

## Aufgabe 4 (Aktivitätsdiagramm)

- denkt an die Rauten!
- [Bedingung]
- verschachtelte Aktivitäten  $\implies$  irgendwo passender Kasten dazu

## Aufgabe 5 (Zustandsdiagramm)

## Aufgabe 6 (Sequenzdiagramm)

## Aufgabe 6 (Sequenzdiagramm)

- bzgl. Konstruktor sind VL-Folien etwas blöd

## Aufgabe 6 (Sequenzdiagramm)

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



## Aufgabe 6 (Sequenzdiagramm)

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

## 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

## Aufgabe 7 (Testen mit Nachahmungen)

- 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 + Hilfekonzpt



**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  $\implies$  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.

## Warum eigentlich?

lokale Änderungen sollen sich nicht auf andere Teile auswirken  
⇒ weniger Fehler und Arbeit

Beispiel?

## Geheimnis- / Kapselungsprinzip

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

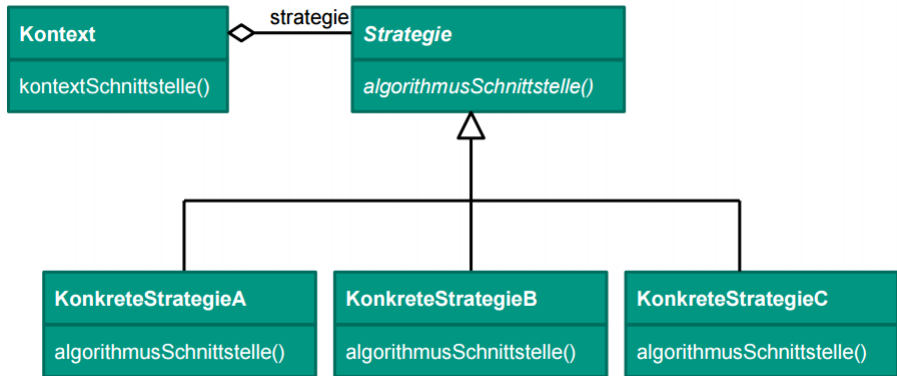
## Warum eigentlich?

lokale Änderungen sollen sich nicht auf andere Teile auswirken  
⇒ weniger Fehler und Arbeit

Beispiel? ⇒ private Attribute mit get()- und set()-Methoden

# Vorgriff: Entwurfsmuster Strategie

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



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**

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

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

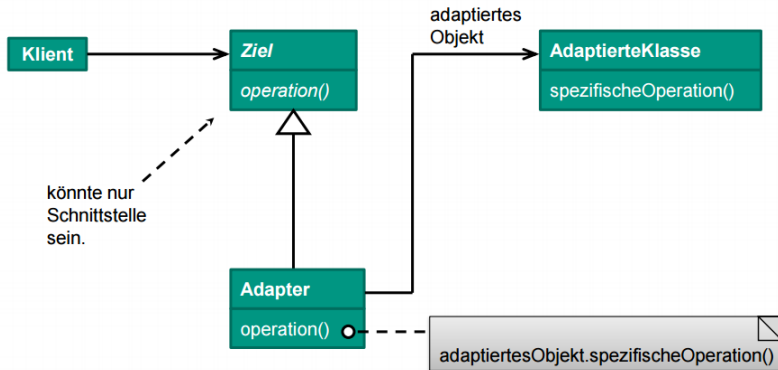


## Problem

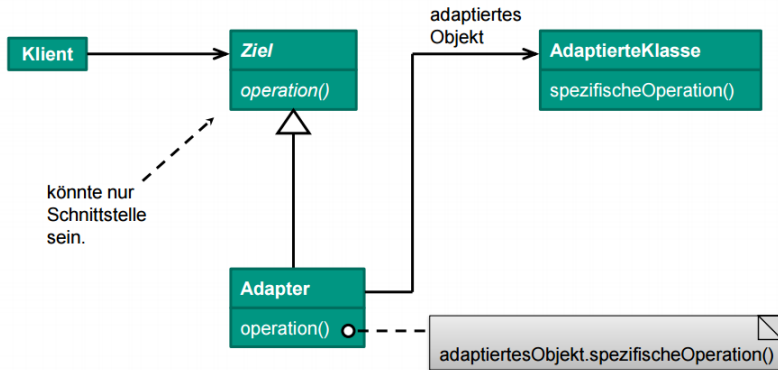
- Klassen mit inkompatiblen Schnittstellen, die wir aber zusammen benutzen wollen
- Schnittstellen nicht änderbar (z.B. externe Bibliotheken)

## Problem

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



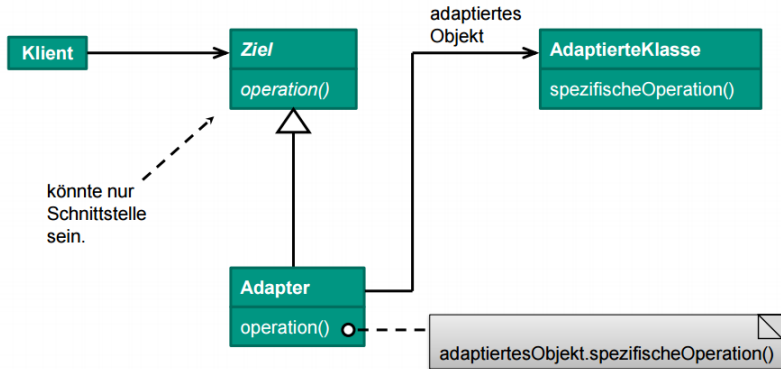
# Adapter (Objektadapter)



Wir sind bei Entkopplung-Mustern, Preisfrage:

Wo ist hier die Entkopplung?

# Adapter (Objektadapter)

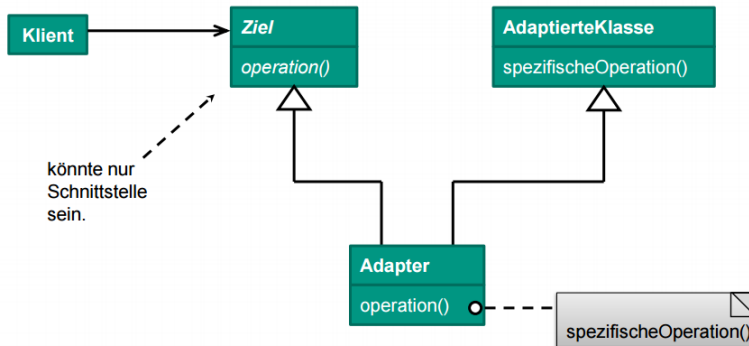


Wir sind bei Entkopplung-Mustern, Preisfrage:

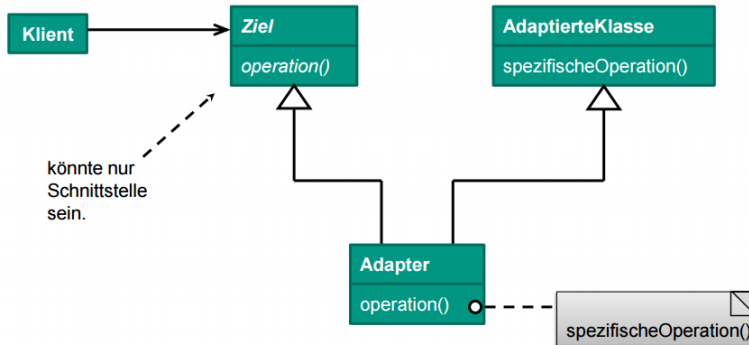
Wo ist hier die Entkopplung?

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

# Adapter - Alternative (Klassenadapter)

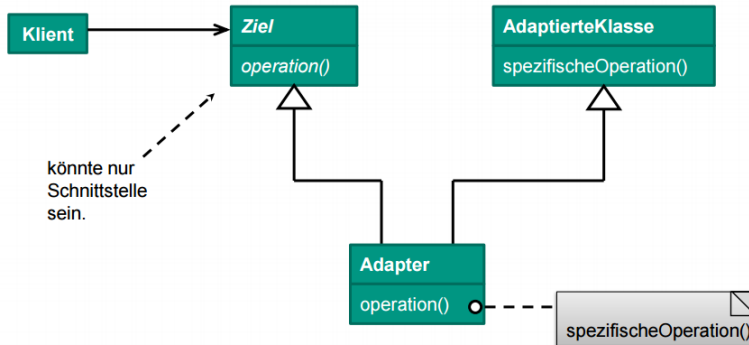


# Adapter - Alternative (Klassenadapter)



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

# Adapter - Alternative (Klassenadapter)



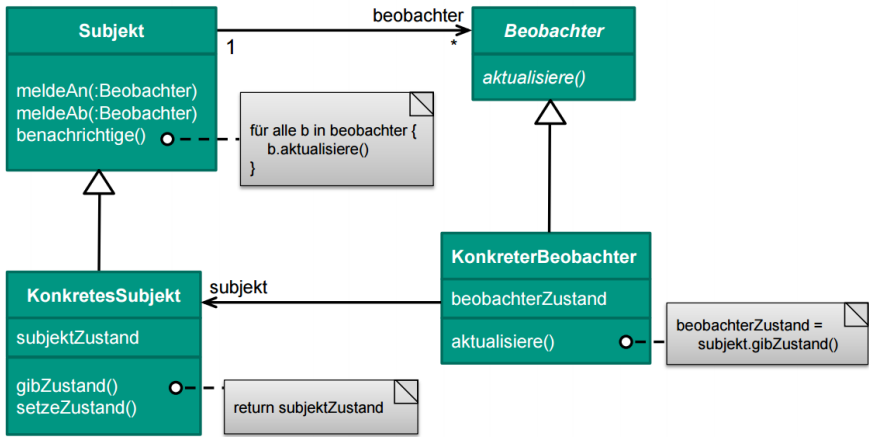
Was für ein Problem bekommt ihr, wenn ihr das auf einem ÜB implementieren müsst?

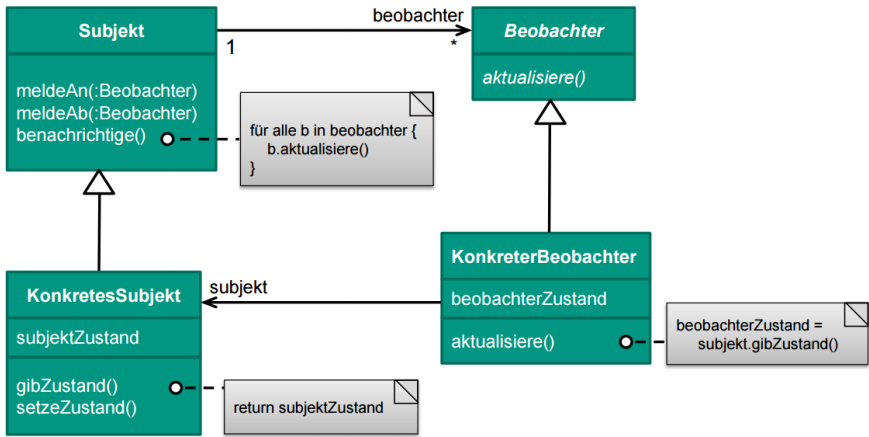
⇒ keine Mehrfachvererbung in Java!

## Problem

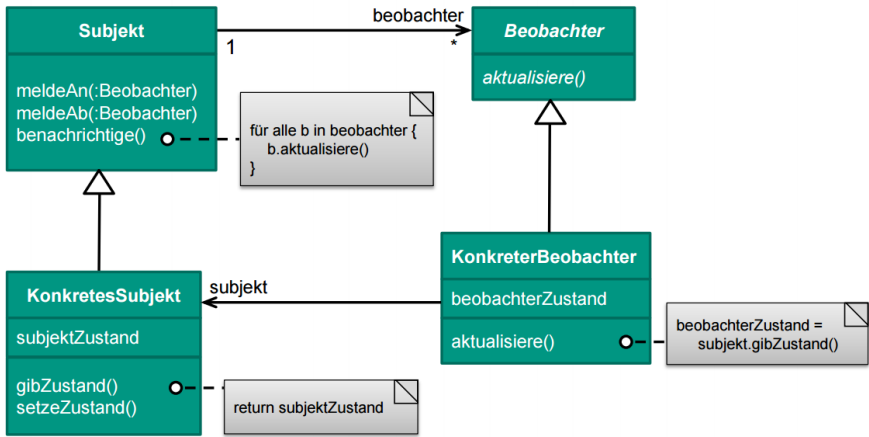
- ein Subjekt, viele Beobachter
- Subjekt ändert Zustand  $\implies$  Beobachter machen "irgendwas"





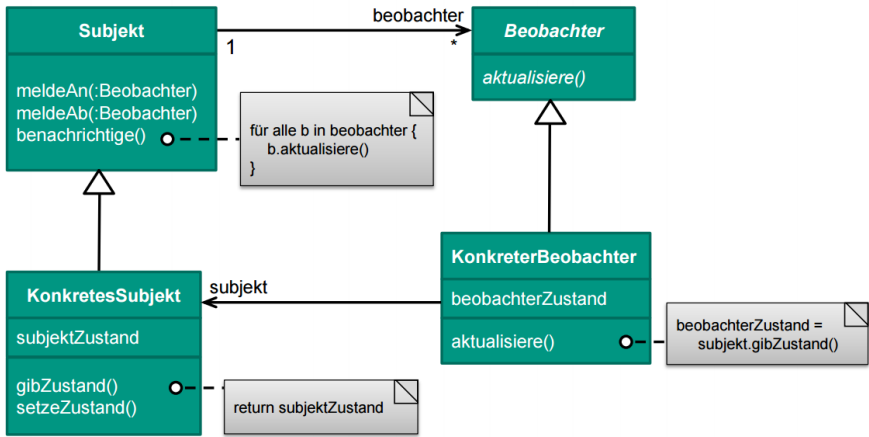


## Entkopplung?



## Entkopplung?

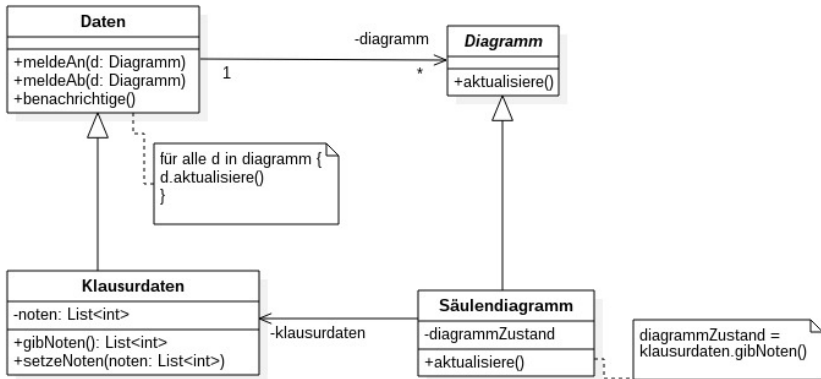
- jeder Beobachter definiert, was bei Benachrichtigung passiert, Subjekt kriegt davon nichts mit



## Entkopplung?

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

# Beobachter/Observer: am Beispiel



## Problem

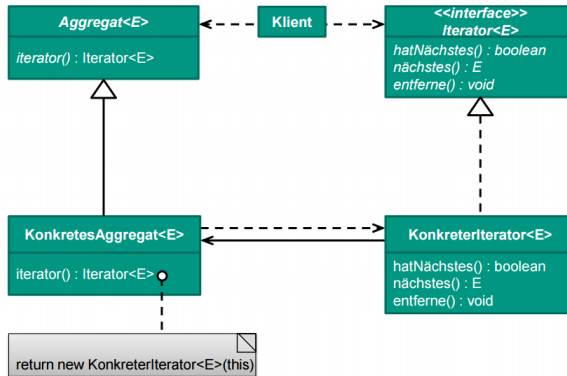
- wollen über Datenstruktur iterieren + Operationen ausführen  
⇒ Hinzufügen, Löschen...

## Problem

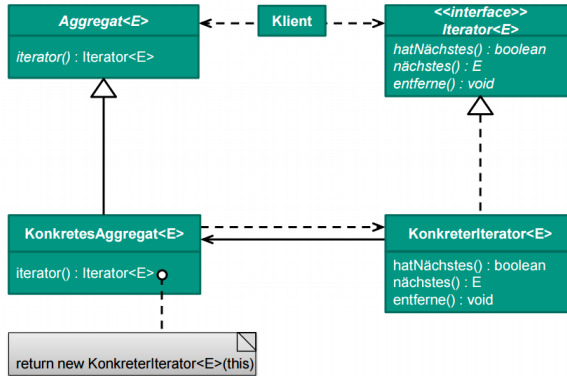
- wollen über Datenstruktur iterieren + Operationen ausführen  
⇒ Hinzufügen, Löschen. . .
- das Ganze ohne Kenntnis des internen Aufbaus der Datenstruktur

## Problem

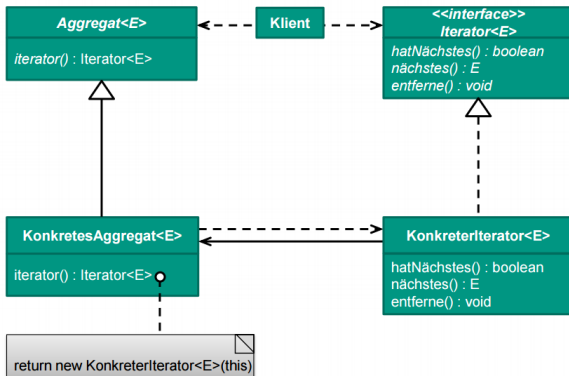
- wollen über Datenstruktur iterieren + Operationen ausführen  
⇒ Hinzufügen, Löschen...
- das Ganze ohne Kenntnis des internen Aufbaus der Datenstruktur





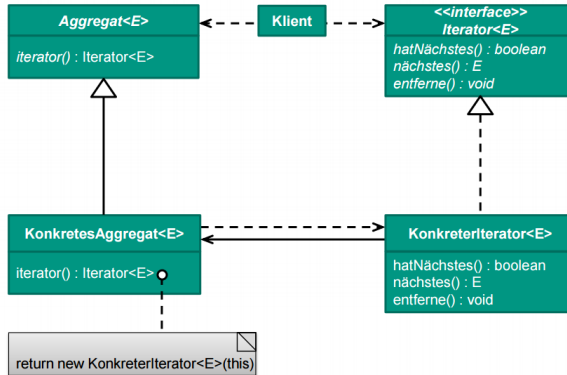


## Entkopplung?



## Entkopplung?

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



Beispiel in Java: `list.iterator();`

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

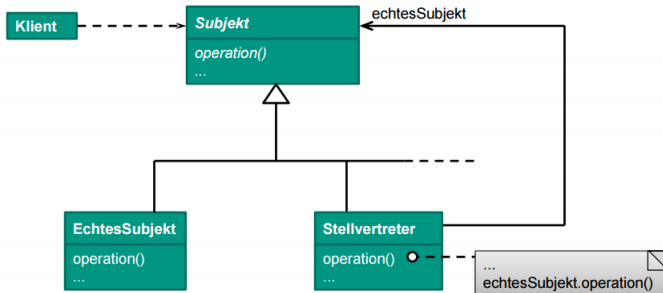


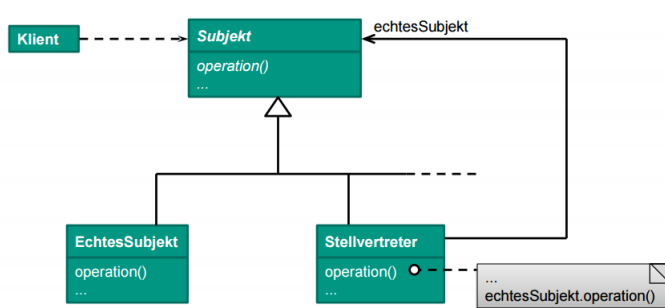
## Problem

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

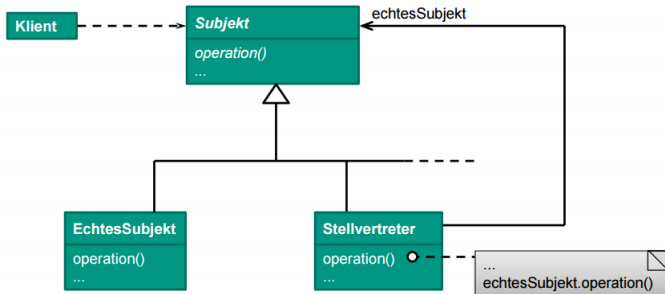
## Problem

- 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

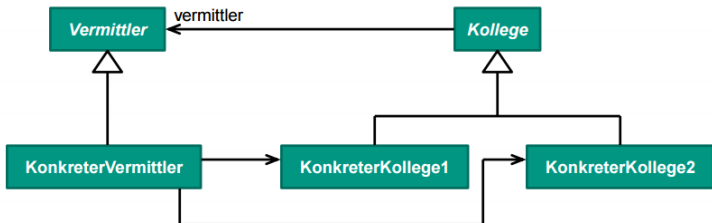
- mehrere voneinander abhängige Objekte

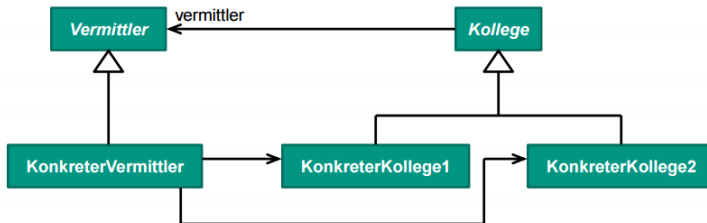
## Problem

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

## Problem

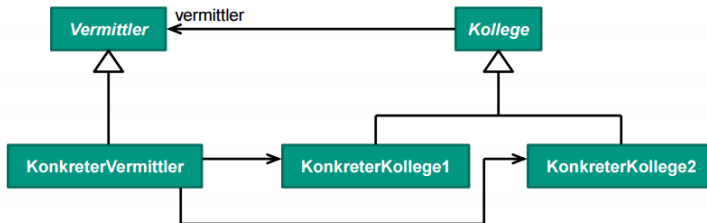
- mehrere voneinander abhängige Objekte  
⇒ Zustände der Objekte von anderen Zuständen abhängig





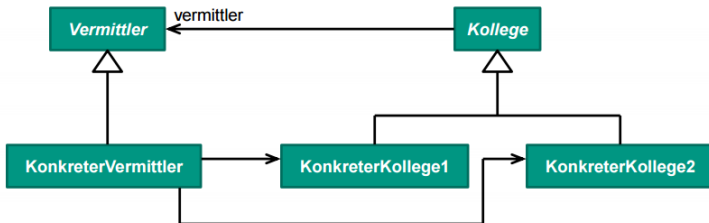
## Entkopplung?





## Entkopplung?

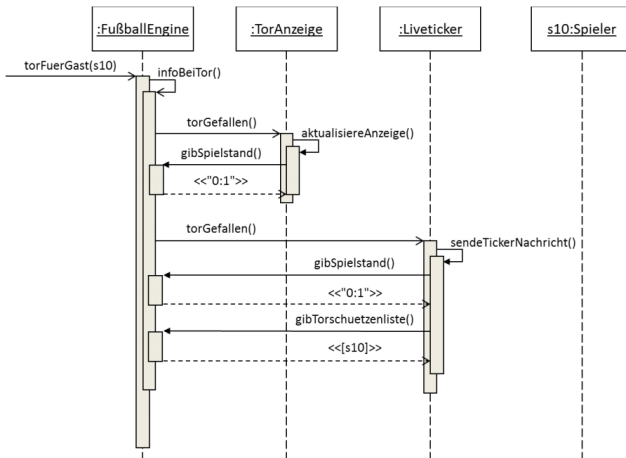
- Kollegen kennen sich nicht direkt



## Entkopplung?

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

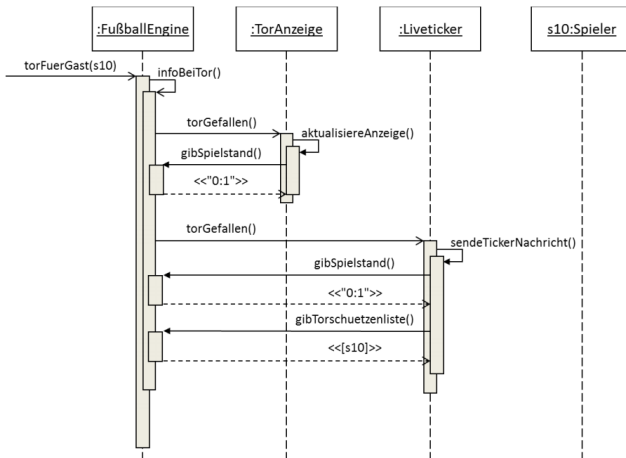
# Klausuraufgabe (Hauptklausur SS 2012)



## Aufgabe 1

Welches Entwurfsmuster erkennen Sie in diesem Diagramm?

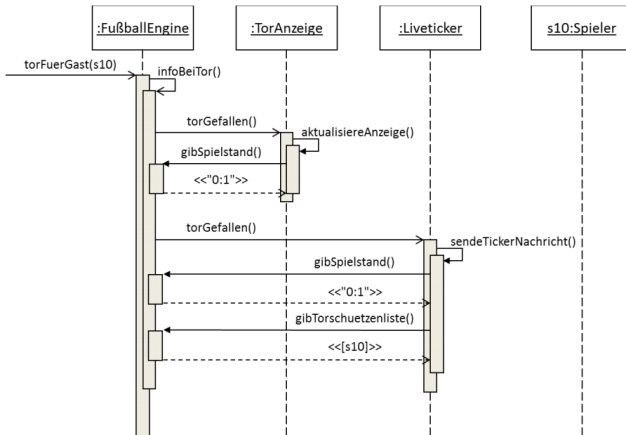
# Klausuraufgabe (Hauptklausur SS 2012)

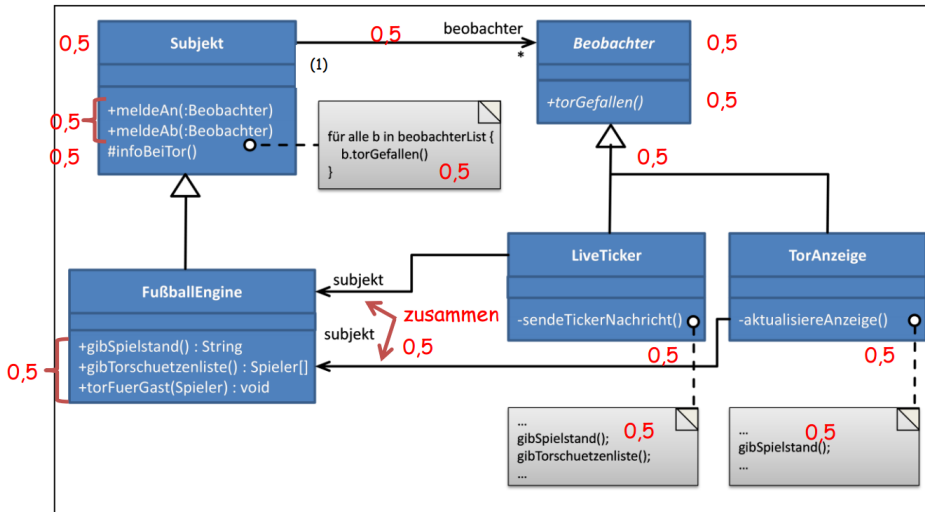


## Aufgabe 1

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 1: iMage-GUI

- macht die “kleinen” Bonusaufgaben  
⇒ relativ leichte Punkte

## Aufgabe 1: iMage-GUI

- macht die “kleinen” Bonusaufgaben  
⇒ relativ leichte Punkte
- schaut euch die verschiedenen LayoutManager aus Java Swing an  
⇒ verschiedene LayoutManager möglich (via mehrerer Container, z.B. JPanel)



## Aufgabe 1: iMage-GUI

- macht die “kleinen” Bonusaufgaben  
⇒ relativ leichte Punkte
- schaut euch die verschiedenen LayoutManager aus Java Swing an  
⇒ verschiedene LayoutManager möglich (via mehrerer Container, z.B. JPanel)

## Aufgabe 2: Zustandsdiagramm (LEZ)

- nochmal Syntax anschauen  
⇒ Was darf in Zustandsdiagramm, was nicht? (laut VL)
- von Hand!

## Aufgabe 3: Git

- `echo "hallo">>> test.txt` schreibt hallo in test.txt
- git-Dokumentation anschauen

## Aufgabe 3: Git

- `echo "hallo">> test.txt` schreibt hallo in test.txt
- git-Dokumentation anschauen

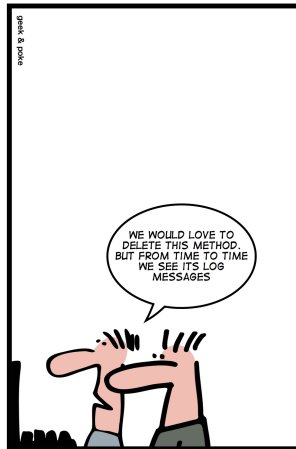
## Aufgabe 4: Architekturstile

- Jmjrst (gedanklich) umbauen
- Zusammenhang der Klassen anschauen (z.B. Main-Klasse)

## Abgabe

- Deadline am 13.6 um 12:00
- A2-4 handschriftlich!

# Bis dann! (dann := 19.06.17)



UNDEAD CODE