

Softwaretechnik 1 - 4. Tutorium

Tutorium 03

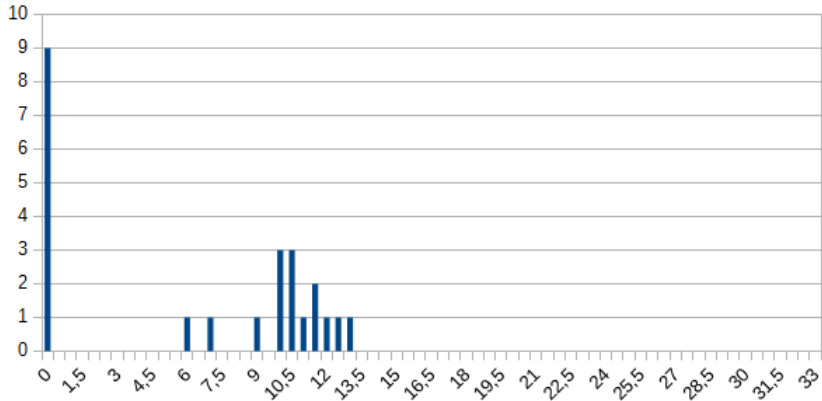
Felix Bachmann | 19.06.2018

KIT - INSTITUT FÜR PROGRAMMSTRUKTUREN UND DATENORGANISATION (IPD)



4. Übungsblatt Statistik

n=24



Ø von 25+3

Allgemein

- Form bei handschriftlichen Abgaben. . .

Aufgabe 1: GUI für iMage

Aufgabe 1: GUI für iMage

■ bla

Aufgabe 2: Zustandsdiagramm für Wasserzeichen

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

Aufgabe 2: Zustandsdiagramm für Wasserzeichen

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

Aufgabe 2: Zustandsdiagramm für Wasserzeichen

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

Aufgabe 3: git

Aufgabe 2: Zustandsdiagramm für Wasserzeichen

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

Aufgabe 3: git

- bla

Aufgabe 4: Architekturstile für JMJRST

Aufgabe 4: Architekturstile für JMJRST

■ bla

Was bisher geschah..

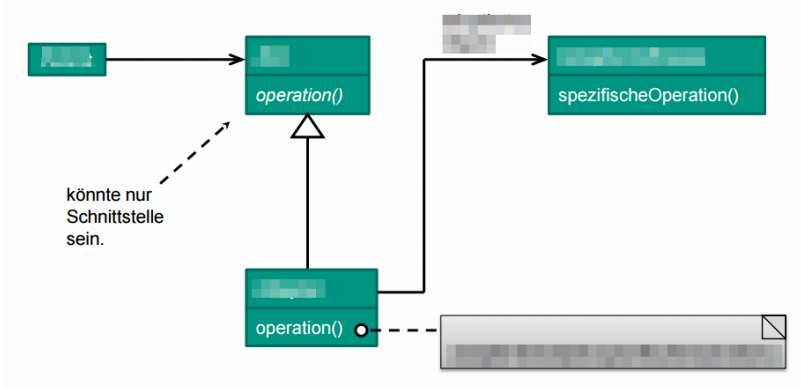
- haben uns Entkopplungsmuster angeschaut

Was bisher geschah..

- haben uns Entkopplungsmuster angeschaut
⇒ Beobachter, Iterator, Adapter, Stellvertreter, Vermittler

Was bisher geschah..

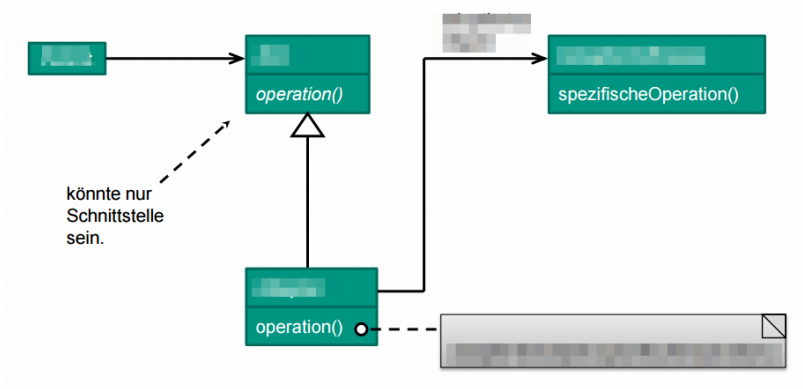
- haben uns Entkopplungsmuster angeschaut
⇒ Beobachter, Iterator, Adapter, Stellvertreter, Vermittler



Welches Entwurfsmuster?

Was bisher geschah..

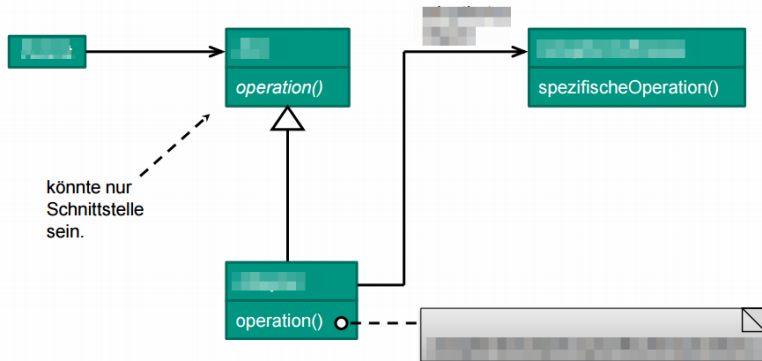
- haben uns Entkopplungsmuster angeschaut
⇒ Beobachter, Iterator, Adapter, Stellvertreter, Vermittler



Welches Entwurfsmuster? (Objekt-)Adapter

Was bisher geschah..

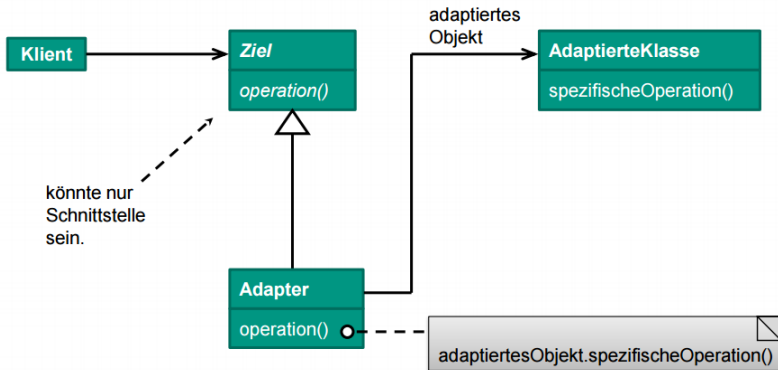
- haben uns Entkopplungsmuster angeschaut
⇒ Beobachter, Iterator, Adapter, Stellvertreter, Vermittler



Welche Klassen?

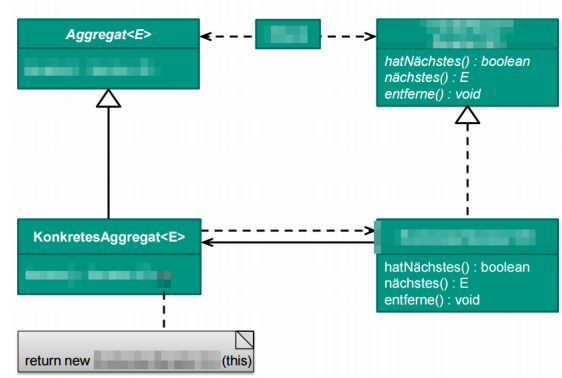
Was bisher geschah..

- haben uns Entkopplungsmuster angeschaut
 - ⇒ Beobachter, Iterator, Adapter, Stellvertreter, Vermittler



Was bisher geschah..

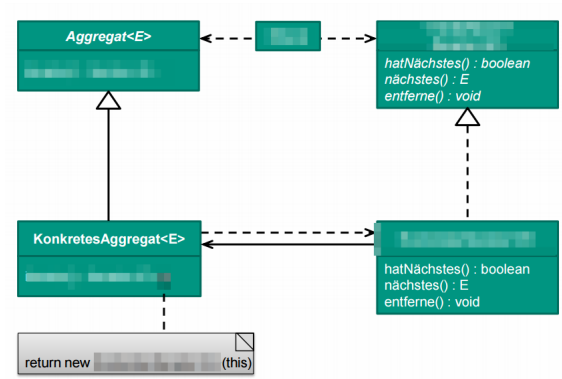
- haben uns Entkopplungsmuster angeschaut
⇒ Beobachter, Iterator, Adapter, Stellvertreter, Vermittler



Welches Entwurfsmuster?

Was bisher geschah..

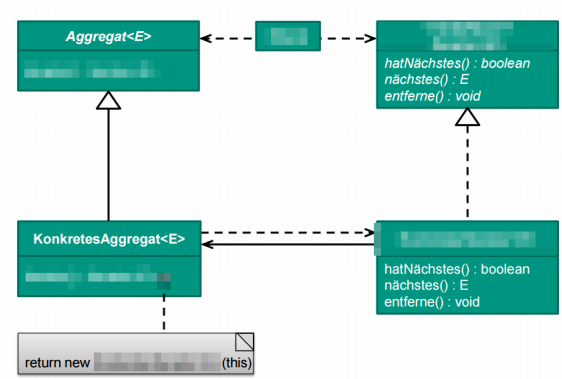
- haben uns Entkopplungsmuster angeschaut
⇒ Beobachter, Iterator, Adapter, Stellvertreter, Vermittler



Welches Entwurfsmuster? Iterator

Was bisher geschah..

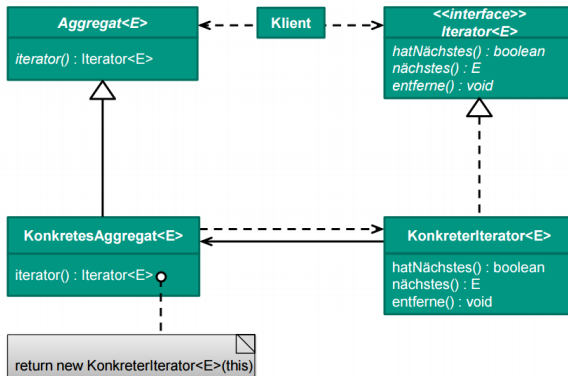
- haben uns Entkopplungsmuster angeschaut
⇒ Beobachter, Iterator, Adapter, Stellvertreter, Vermittler



Welche Klassen und Methoden?

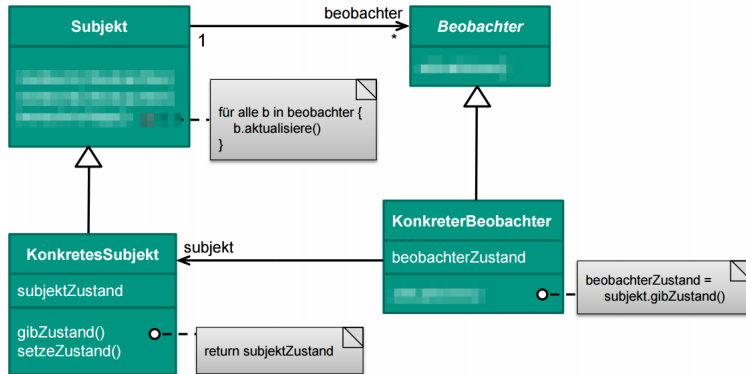
Was bisher geschah..

- haben uns Entkopplungsmuster angeschaut
⇒ Beobachter, Iterator, Adapter, Stellvertreter, Vermittler



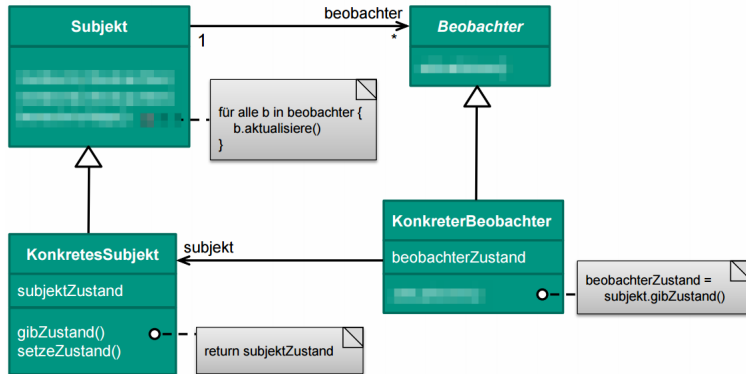
Was bisher geschah..

- haben uns Entkopplungsmuster angeschaut
⇒ Beobachter, Iterator, Adapter, Stellvertreter, Vermittler



Was bisher geschah..

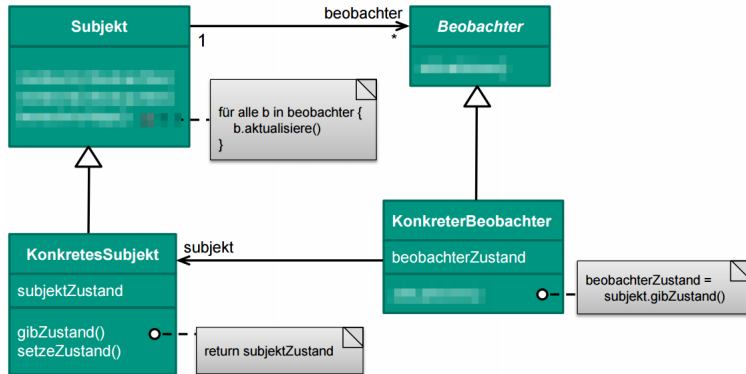
- haben uns Entkopplungsmuster angeschaut
⇒ Beobachter, Iterator, Adapter, Stellvertreter, Vermittler



Ist wohl ein Beobachter :)

Was bisher geschah..

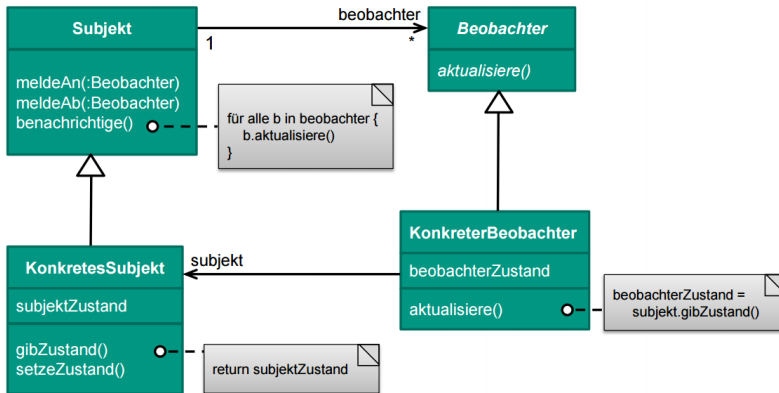
- haben uns Entkopplungsmuster angeschaut
⇒ Beobachter, Iterator, Adapter, Stellvertreter, Vermittler



Ist wohl ein Beobachter :) Methoden?

Was bisher geschah..

- haben uns Entkopplungsmuster angeschaut
⇒ Beobachter, Iterator, Adapter, Stellvertreter, Vermittler



■ Entkopplungs-Muster

- Adapter fertig
- Beobachter fertig
- Iterator fertig
- Stellvertreter fertig
- Vermittler fertig
- (Brücke)

■ Varianten-Muster

■ Zustandshandhabungs-Muster

■ Steuerungs-Muster

■ Bequemlichkeits-Muster

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

Übergeordnetes Ziel

- Gemeinsamkeiten herausziehen und an einer Stelle beschreiben

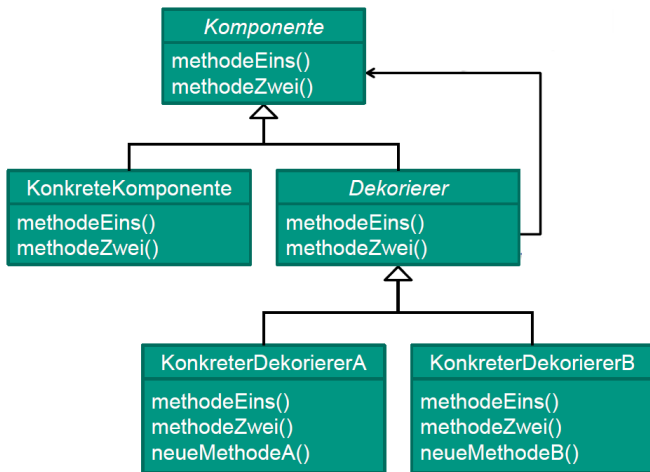
Übergeordnetes Ziel

- Gemeinsamkeiten herausziehen und an einer Stelle beschreiben
⇒ keine Wiederholung desselben Codes

Übergeordnetes Ziel

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

- 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



Wo Gemeinsamkeiten?

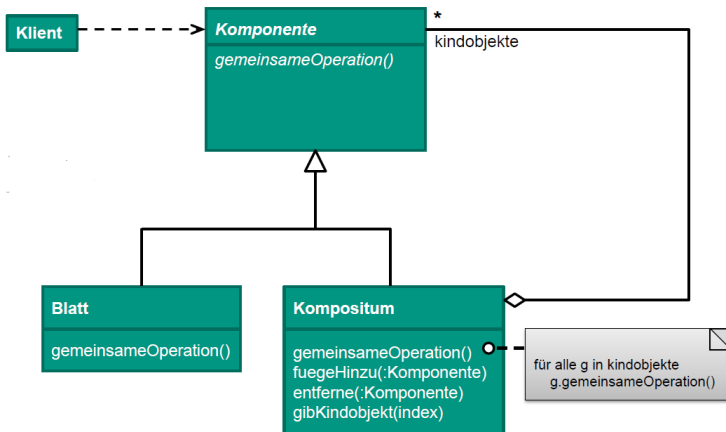
Die beiden Methoden `methodeEins()` und `methodeZwei()`.

Wo Variation?

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

Wozu Instanzvariable?

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



Wo Gemeinsamkeiten?

gemeinsameOperation().

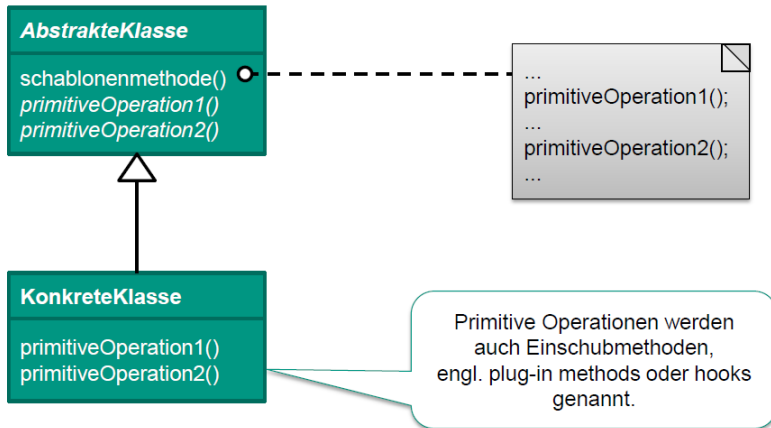
Wo Variation?

In Blatt/Kompositum-Klassen mit verschiedenen zusätzlichen Operationen.

Zusammengesetzt vs. nicht-zusammengesetzt

Kompositum = zusammengesetzt, Blatt = nicht-zusammengesetzt

Vorstellung Schablonenmethode



Wo Gemeinsamkeiten?

Reihenfolge der Methodenaufrufe in der Schablonenmethode.

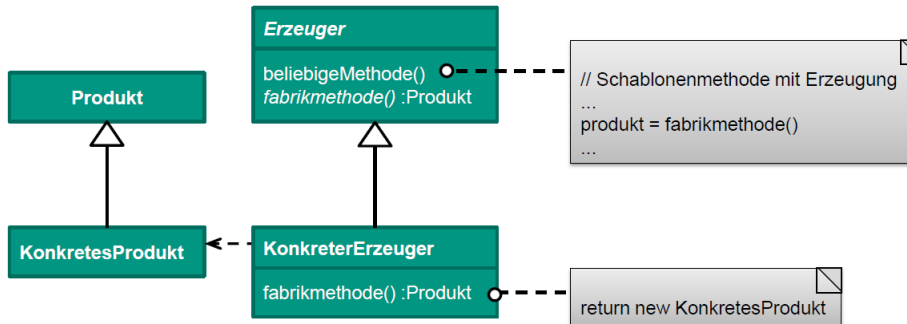
Wo Variation?

In den Einschubmethoden. (hier: `primitiveOperation1()` und `primitiveOperation2()`)

Schablonenmethode vs. Einschubmethode

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

Vorstellung 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.

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

Übergeordnetes Ziel

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

Übergeordnetes Ziel

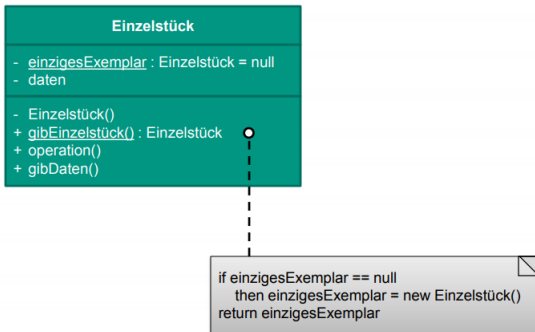
- den Zustand eines Objektes beschreiben (wer hätt's gedacht? :D)
- aber unabhängig von dem Zweck des Objekts!

Problem

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

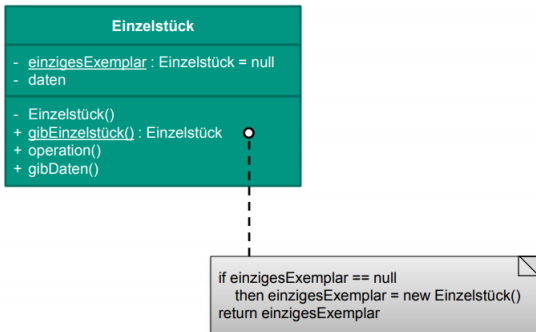
Problem

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



Problem

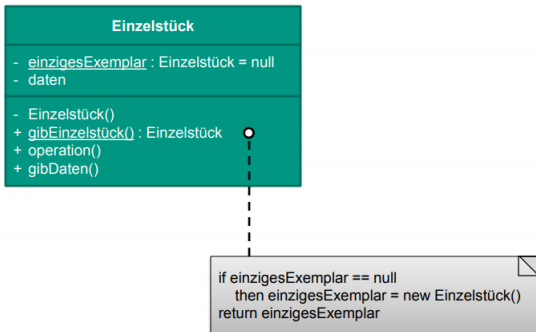
- von einer Klasse soll nur eine Instanz existieren
- Konstruktor könnte überall benutzt werden!



Aber warum nicht einfach statisch?

Problem

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



Aber warum nicht einfach statisch? Unterklassenbildung möglich!

Problem

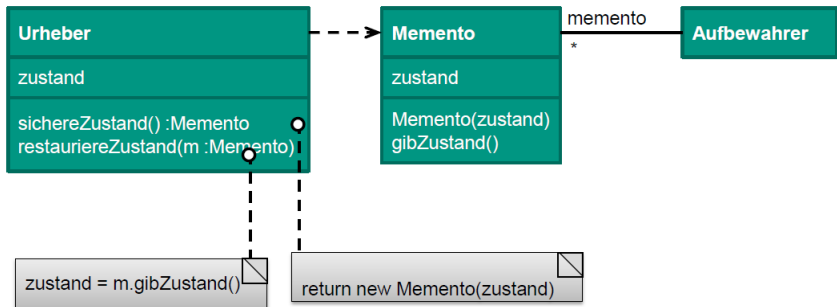
- internen Zustand eines Objekts “externalisieren“, um z.B. Zurücksetzen möglich zu machen

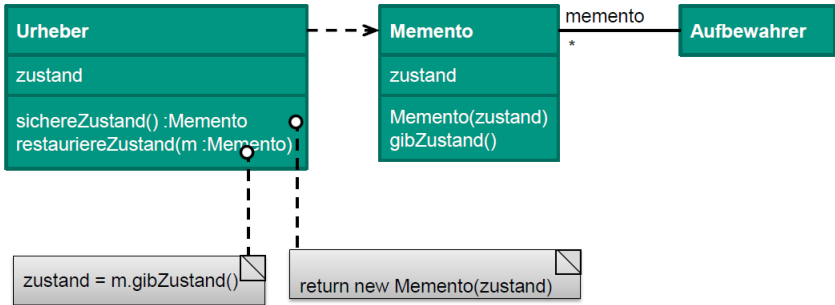
Problem

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

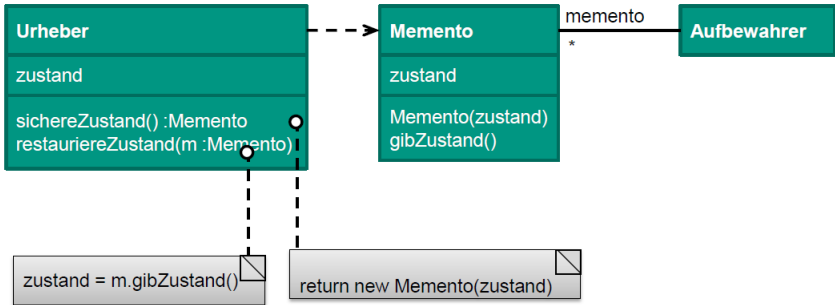
Problem

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



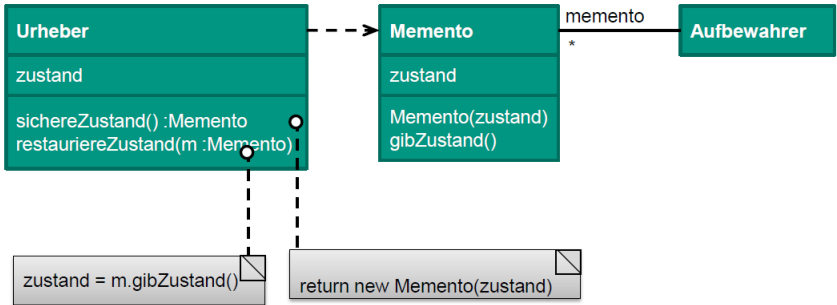


Problem gelöst?



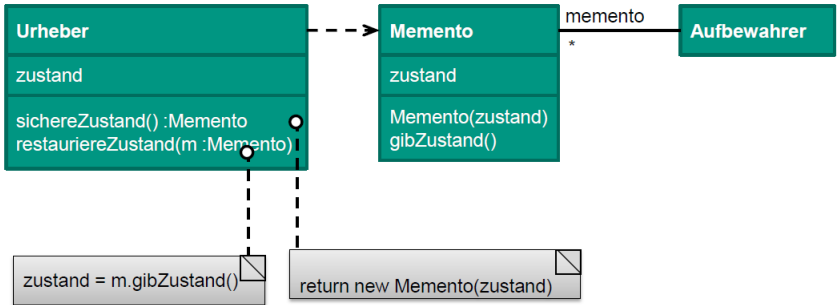
Problem gelöst?

■ Ja



Problem gelöst?

- Ja
 - Zustand durch Memento externalisiert



Problem gelöst?

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

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

Übergeordnetes Ziel

- steuern den Kontrollfluss

Übergeordnetes Ziel

- steuern den Kontrollfluss
⇒ zur richtigen Zeit richtige Methoden aufrufen

Problem

- Parametrisieren von Objekten mit einer auszuführenden Aktion

Problem

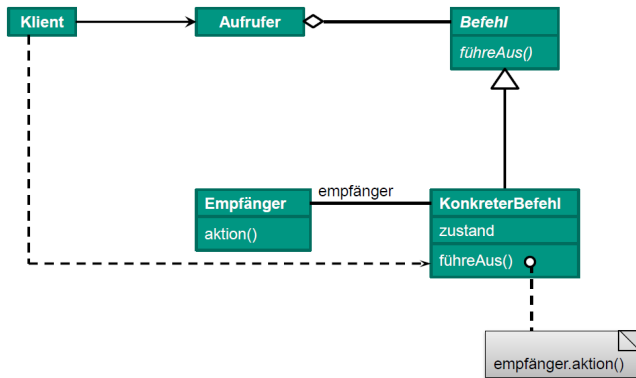
- Parametrisieren von Objekten mit einer auszuführenden Aktion
- komplexe Operationen aus primitiven Operationen aufbauen

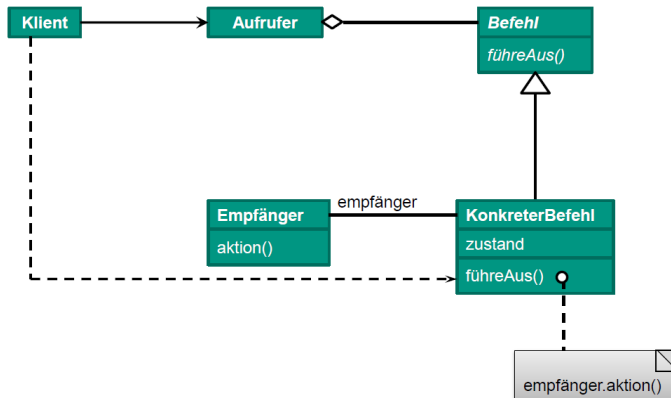
Problem

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

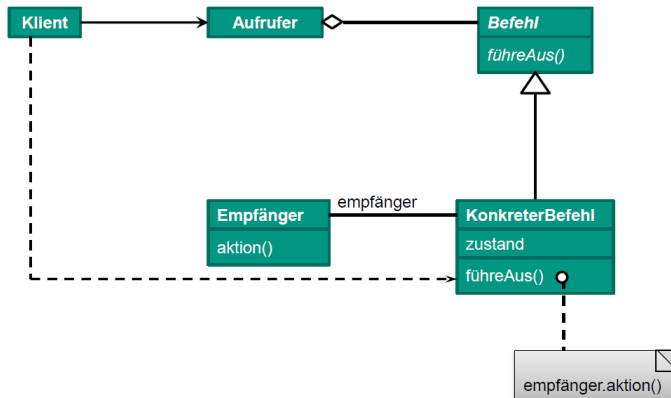
Problem

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



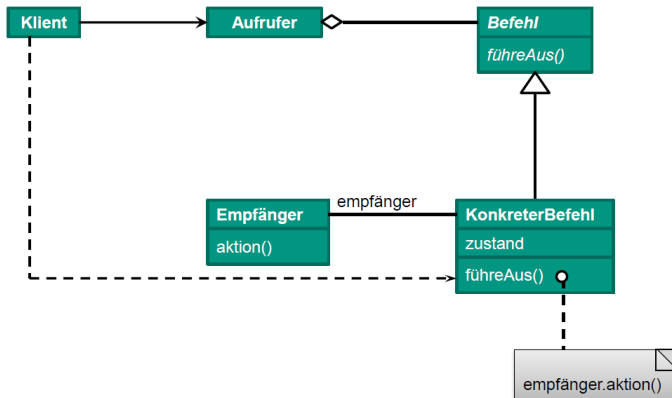


Was haben wir erreicht?



Was haben wir erreicht?

- Austauschbarkeit: Befehle unabhängig vom Aufrufer, universell einsetzbar



Beispiel!

Wahr oder falsch?

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

Wahr oder falsch?

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

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 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**

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.

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**

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.

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**

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.

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**

- Entwurfsmuster kommen sehr sehr sehr wahrscheinlich dran!

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

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

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

Feedback - Sagt mir eure Meinung



<https://goo.gl/forms/osMyz2hScqVPtmSr2>

Aufgabe 1: Shutterpile: Refaktorisierung + Entwurfsmuster anwenden

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

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`

Aufgabe 3: Wo sind Entwurfsmuster in Shutterpile?

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

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 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

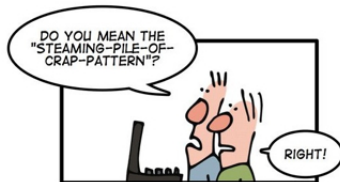
Aufgabe 5: Entwurfsmuster - Kaffeemaschine

- ein Muster anwenden

Abgabe

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

Bis dann! (dann := 03.07.18)



THE HYPE IS LONG GONE BUT
DESIGN PATTERNS ARE STILL USEFUL