

Softwaretechnik 1 - 4. Tutorium

Tutorium 17 Felix Bachmann | 25.06.2019

KIT - INSTITUT FÜR PROGRAMMSTRUKTUREN UND DATENORGANISATION (IPD)

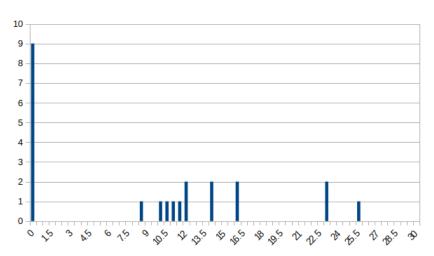
Was passiert heute?



- 1 Feedback + Eval
- 2 Recap
- 3 Vermittler
- 4 Aufgabe
- Rest Entwurfsmuster
- 6 Tipps

3. Übungsblatt Statistik







Programmieraufgaben generell

wie letztes Mal gesagt

Vermittler



Programmieraufgaben generell

- wie letztes Mal gesagt
- und das Mal davor

Vermittler



Programmieraufgaben generell

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



Programmieraufgaben generell

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



Programmieraufgaben generell

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



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)



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 Klassennamen vergleichen,
 - nicht Object#getName()
 - nicht PluginForJMJRST#getName()
 - z.B. Object#getSimpleName()



Aufgabe 2 (Instagrim Plug-In)

Felix Bachmann - SWT1



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



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 f
 ür configure()-Aufruf selbst gebaut
 - kein Abzug solange kein Quatsch passiert und sinnvoll programmiert
 - geht aber auch deutlich einfacher (siehe MuLö)



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 f
 ür 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



Aufgabe 4 (Aktivitätsdiagramm)

Felix Bachmann - SWT1

Aufgabe



Aufgabe 4 (Aktivitätsdiagramm)

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

Vermittler

"Startseite anzeigen" schon



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



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



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]



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
- lacktriang [Bedingung]
- Modellierung der Nutzerinteraktion
 - z.B. "Klick" nachdem "Startseite anzeigen" vorbei? nicht sinnvoll

Aufgabe



Aufgabe 5 (Zustandsdiagramm)

Vermittler



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!

25.06.2019



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



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.



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



Aufgabe 7 (Substitutionsprinzip: Logger)

Vermittler



Aufgabe 7 (Substitutionsprinzip: Logger)

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

Vermittler



Aufgabe 7 (Substitutionsprinzip: Logger)

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



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 Uberschreiben von Methoden :)
- bei solchen Aufgaben immer aus Klient-Sicht anschauen

Aufgabe



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

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

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

Vermittler



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

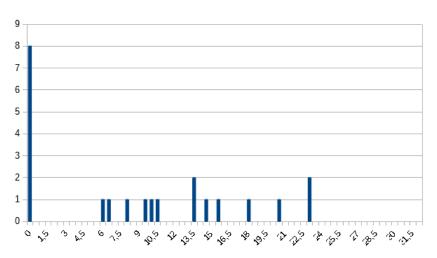
- zweiter Aufruf "muss sich so verhalten wie" erster Aufruf
- Vorbedingung schwächer/gleich
- Nachbedingung stärker/gleich
- "Unterklasse darf weniger verlangen, muss aber mehr leisten"

Aufgabe

- (oder gleich viel)
- Hintergrund: Klient muss Unterklasse so benutzen k\u00f6nnen wie Oberklasse. W\u00fcrde die Unterklasse bspw. mehr verlangen als Oberklasse m\u00fcsste Klient wissen, dass er Unterklasse benutzt, um sie korrekt aufrufen zu k\u00f6nnen.

4. Übungsblatt Statistik







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");
```

Vermittler



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



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
 - ist thread safe (siehe n\u00e4chstes Tut)



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
 - ist thread safe (siehe n\u00e4chstes Tut)
- nicht-navigierbare JFileChooser



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
 - ist thread safe (siehe n\u00e4chstes Tut)
- nicht-navigierbare JFileChooser
- Raw Types: JComboBox vs. JComboBox<String>
 - ersteres nicht typsicher, müssen theoretisches jedes Mal instanceof aufrufen

Felix Bachmann - SWT1



Aufgabe 2: Aktivitätsdiagramm → Zustandsdiagramm



Aufgabe 2: Aktivitätsdiagramm → Zustandsdiagramm

- entry/-Aktionen weggelassen, weil Zustand schon so heißt
 - z.B. "StartseiteAnzeigen"



Aufgabe 2: Aktivitätsdiagramm → Zustandsdiagramm

- entry/-Aktionen weggelassen, weil Zustand schon so heißt
 - z.B. "StartseiteAnzeigen"
- "Brühvorgang beendet" fehlt, aber ok wenn anders sinnvoll modelliert
 - sollte ja minimal sein

Vermittler

25.06.2019



Aufgabe 2: Aktivitätsdiagramm → Zustandsdiagramm

- entry/-Aktionen weggelassen, weil Zustand schon so heißt
 - z.B. "StartseiteAnzeigen"
- "Brühvorgang beendet" fehlt, aber ok wenn anders sinnvoll modelliert
 - sollte ja minimal sein

Aufgabe 3: Geheimnisprinzip



Aufgabe 2: Aktivitätsdiagramm → Zustandsdiagramm

- entry/-Aktionen weggelassen, weil Zustand schon so heißt
 - z.B. "StartseiteAnzeigen"
- "Brühvorgang beendet" fehlt, aber ok wenn anders sinnvoll modelliert
 - sollte ja minimal sein

Aufgabe 3: Geheimnisprinzip

ist nicht nur private Attribute und get(), set(x)



Aufgabe 2: Aktivitätsdiagramm → Zustandsdiagramm

- entry/-Aktionen weggelassen, weil Zustand schon so heißt
 - z.B. "StartseiteAnzeigen"
- "Brühvorgang beendet" fehlt, aber ok wenn anders sinnvoll modelliert
 - sollte ja minimal sein

Aufgabe 3: Geheimnisprinzip

- ist nicht nur private Attribute und get(), set(x)
- verborgen wird immer wie, Interfaces sagen nur was
 - auf Closeable-Objekt kann mittels close() Systemresource geschlossen werden
 - unabhängig wie Zugriff auf Resource funktioniert

Aufgabe

einige impl. Klassen: ZipFile, SSLSocket, AudioInputStream



Aufgabe 4: Modul-Abhängigkeiten



Aufgabe 4: Modul-Abhängigkeiten

Systemgrenze "iMage" hat gefehlt



Aufgabe 4: Modul-Abhängigkeiten

- Systemgrenze "iMage" hat gefehlt
- commons-*, ojalgo, hamcrest-core vergessen



Aufgabe 4: Modul-Abhängigkeiten

- Systemgrenze "iMage" hat gefehlt
- commons-*, ojalgo, hamcrest-core vergessen
- Name sollten Modulnamen (intern) oder artifactId (extern) sein
 - aber kein Abzug, solange erkennbar

25.06.2019



Aufgabe 4: Modul-Abhängigkeiten

- Systemgrenze "iMage" hat gefehlt
- commons-*, ojalgo, hamcrest-core vergessen
- Name sollten Modulnamen (intern) oder artifactId (extern) sein
 - aber kein Abzug, solange erkennbar

Aufgabe 5: Entwurfsmuster in HDrize



Aufgabe 4: Modul-Abhängigkeiten

- Systemgrenze "iMage" hat gefehlt
- commons-*, ojalgo, hamcrest-core vergessen
- Name sollten Modulnamen (intern) oder artifactId (extern) sein
 - aber kein Abzug, solange erkennbar

Aufgabe 5: Entwurfsmuster in HDrize

- Abbildung Klassen zu Elementen aus VL-Muster fehlte
 - z.B. GrayScaleDecorater ist KonkreterDekorierer

Evaluation der Evaluation



leider nur 5 Teilnehmer??

| Gut | Schlecht/Verbesserungswürdig |
|------------------------|------------------------------------|
| Folien (3) | |
| Beispiele und Code (3) | |
| Erklärungen (2) | |
| Tipps (2) | |
| Aufgaben (2) | zu viel Zeit für Aufgaben (2) |
| | Wahr/Falsch zu einfach (1) |
| | nicht immer Lösungen auf Folie (1) |
| | gleiche Beispiele wie in VL (1) |
| | Bewertungen zu kurz (1) |



haben uns Entkopplungmuster angeschaut

Tipps



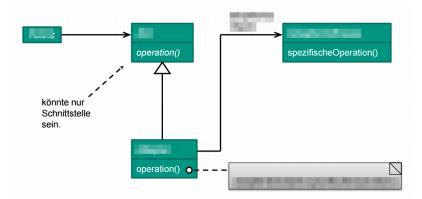
haben uns Entkopplungmuster angeschaut

⇒ Beobachter, Iterator, Adapter, Stellvertreter





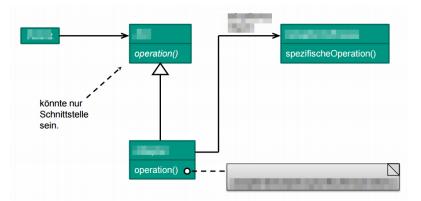
- haben uns Entkopplungmuster angeschaut
 - ⇒ Beobachter, Iterator, Adapter, Stellvertreter



Welches Entwurfsmuster?



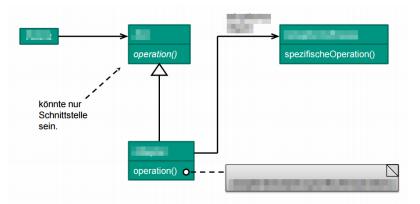
- haben uns Entkopplungmuster angeschaut
 - ⇒ Beobachter, Iterator, Adapter, Stellvertreter



Welches Entwurfsmuster? (Objekt-)Adapter



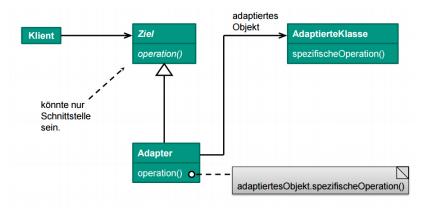
- haben uns Entkopplungmuster angeschaut
 - ⇒ Beobachter, Iterator, Adapter, Stellvertreter



Welche Klassen?

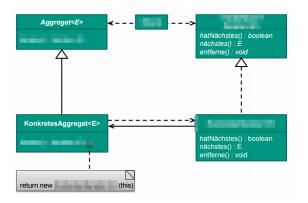


- haben uns Entkopplungmuster angeschaut
 - ⇒ Beobachter, Iterator, Adapter, Stellvertreter





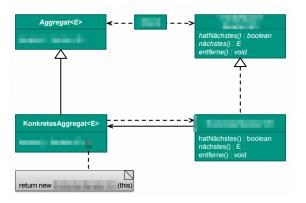
- haben uns Entkopplungmuster angeschaut
 - ⇒ Beobachter, Iterator, Adapter, Stellvertreter



Welches Entwurfsmuster?



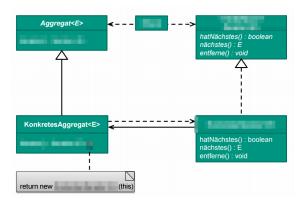
- haben uns Entkopplungmuster angeschaut
 - ⇒ Beobachter, Iterator, Adapter, Stellvertreter



Welches Entwurfsmuster? Iterator



- haben uns Entkopplungmuster angeschaut
 - ⇒ Beobachter, Iterator, Adapter, Stellvertreter



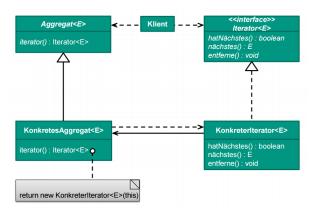
Welche Klassen und Methoden?

Vermittler

25.06.2019

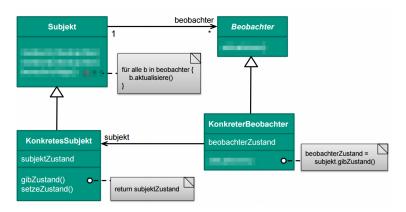


- haben uns Entkopplungmuster angeschaut
 - ⇒ Beobachter, Iterator, Adapter, Stellvertreter





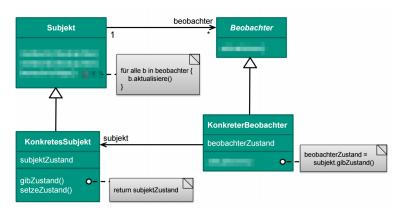
- haben uns Entkopplungmuster angeschaut
 - ⇒ Beobachter, Iterator, Adapter, Stellvertreter



22/66



- haben uns Entkopplungmuster angeschaut
 - ⇒ Beobachter, Iterator, Adapter, Stellvertreter

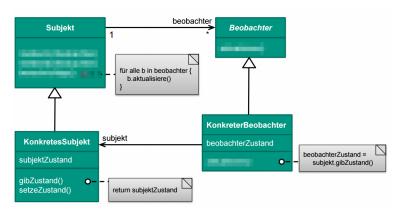


Ist wohl ein Beobachter:)

| Feed | back | + | Eval | |
|------|------|----|------|-----|
| 000 | 000 | 00 | 00 | 000 |



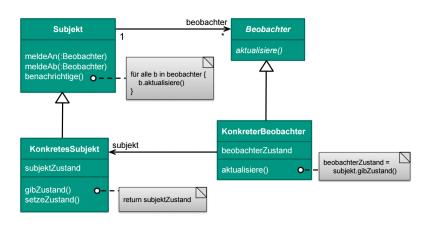
- haben uns Entkopplungmuster angeschaut
 - ⇒ Beobachter, Iterator, Adapter, Stellvertreter



Ist wohl ein Beobachter :) Methoden?



- haben uns Entkopplungmuster angeschaut
 - ⇒ Beobachter, Iterator, Adapter, Stellvertreter





Problem

mehrere voneinander abhängige Objekte



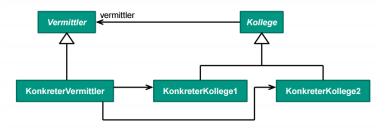
Problem

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

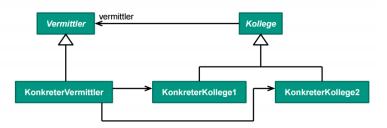


Problem

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



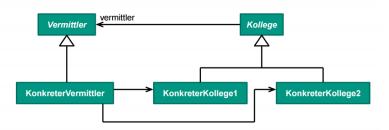




Entkopplung?

000

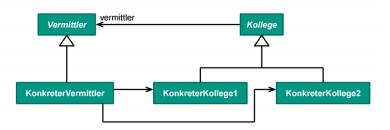




Entkopplung?

Kollegen kennen sich nicht direkt





Entkopplung?

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

Aufgabe

Beobachter vs. Vermittler



- wirken ähnlich
 - ein Vermittler, viele Kollegen
 - ein Subjekt, viele Beobachter

Beobachter vs. Vermittler



- wirken ähnlich
 - ein Vermittler, viele Kollegen
 - ein Subjekt, viele Beobachter

Beobachter

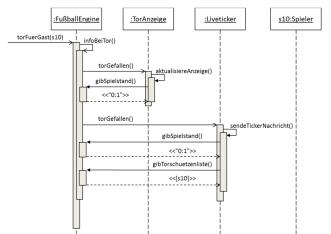
Beobachter interessieren sich nicht füreinander. Nur für das Subjekt

Vermittler

Kollegen interessieren sich füreinander, kommunizieren über den Vermittler miteinander.

Klausuraufgabe (Hauptklausur SS 2012)



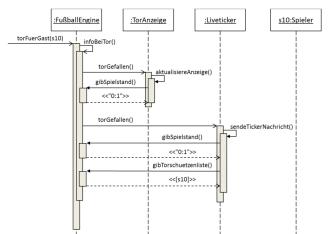


Aufgabe 1

Welches Entwurfsmuster erkennen Sie in diesem Diagramm?

Klausuraufgabe (Hauptklausur SS 2012)





Aufgabe 1

Welches Entwurfsmuster erkennen Sie in diesem Diagramm? Beobachter.

Feedback + Eval Felix Bachmann - SWT1

Vermittler Recap

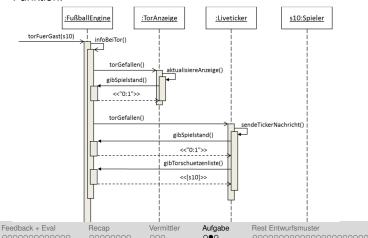
Aufgabe ●○○

Rest Entwurfsmuster

25.06.2019

27/66

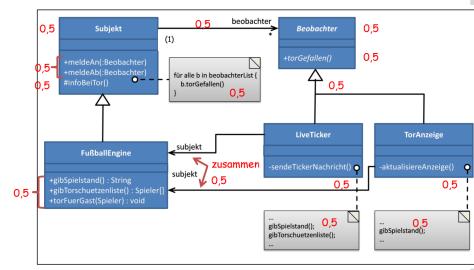
Entwerfen Sie das folgende Klassendiagramm passend zu dem Sequenzdiagramm; es soll alle verwendeten Klassen und Methoden enthalten. Kennzeichnen Sie die Zugreifbarkeiten der Methoden mit den Symbolen +, -, #; seien Sie dabei möglichst restriktiv. Verzichten Sie auf die Modellierung von Attributen. Kennzeichnen Sie die Elemente des Entwurfsmusters und deren Funktion.



Felix Bachmann - SWT1

Musterlösung





Feedback + Eval

Recap

Vermittler

Aufgabe 000

Rest Entwurfsmuster

29/66

Tipps

Kategorien der Entwurfsmuster



Entkopplungs-Muster

- Adapter
- Beobachter
- Iterator
- Stellvertreter
- Vermittler
- (Brücke)
- Varianten-Muster
- Zustandshandhabungs-Muster
- Steuerungs-Muster
- Bequemlichkeits-Muster

Kategorien der Entwurfsmuster



- Entkopplungs-Muster <
- Varianten-Muster
 - (Abstrakte Fabrik)
 - **Besucher**
 - Schablonenmethode
 - Fabrikmethode
 - Kompositum
 - Strategie
 - 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

⇒ keine Wiederholung desselben Codes

Varianten-Muster



Übergeordnetes Ziel

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

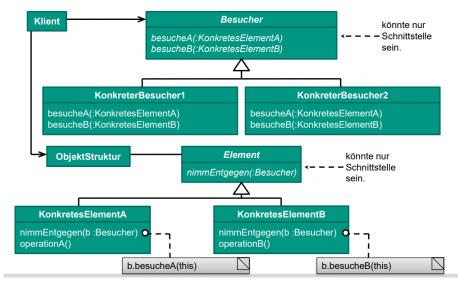


Problem

- Objekte sollen verarbeitet werden
- Verarbeitung der Objekte nicht als Methoden des Objektes
 - Kapselung der Verarbeitung in Besucher

- Besucher "stattet Objekten einen Besuch ab" und verarbeitet sie dabei
- Vorteile: gut neue Methoden hinzufügbar (durch neue Besucher)







Code-Beispiel

25.06.2019



Code-Beispiel

Achtung

- Was ist ein großer Nachteil des Besucher-Musters?
- Womit wird die gute Erweiterbarkeit um Methoden erkauft?



Code-Beispiel

Achtung

- Was ist ein großer Nachteil des Besucher-Musters?
- Womit wird die gute Erweiterbarkeit um Methoden erkauft?
- schlecht um neue (Element-)Klassen erweiterbar!

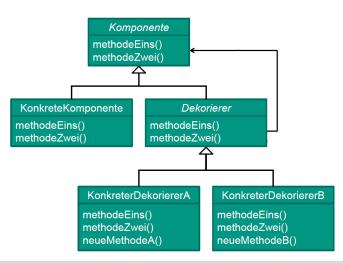
25.06.2019



Problem

- dynamisch neue Funktionalität (=Methoden) zu Klasse hinzufügen
 - aka dekorieren
- ohne Klasse zu ändern oder von ihr zu erben
- vermeidet unübersichtliche Vererbungshierarchien
 - durch rekursives Dekorieren







Beispiel aus java.io

```
// erstelle KonkreteKomponente
FileInputStream fis = new FileInputStream("/objects.gz");
// ab jetzt: dekorieren bis zum get no :)
BufferedInputStream bis = new BufferedInputStream(fis);
GzipInputStream gis = new GzipInputStream(bis);
ObjectInputStream ois = new ObjectInputStream(gis);
// Aufruf von neueMethode()
SomeObject someObject = (SomeObject) ois.readObject();
// ...
ois.close(); // delegiert an die "richtige Stelle" (fis)
```

25.06.2019



Beispiel aus java.io

```
// erstelle KonkreteKomponente
FileInputStream fis = new FileInputStream("/objects.gz");
// ab jetzt: dekorieren bis zum get no :)
BufferedInputStream bis = new BufferedInputStream(fis);
GzipInputStream gis = new GzipInputStream(bis);
ObjectInputStream ois = new ObjectInputStream(gis);
// Aufruf von neueMethode()
SomeObject someObject = (SomeObject) ois.readObject();
// ...
ois.close(); // delegiert an die "richtige Stelle" (fis)
```

Ohne Dekorierer

z.B. Klasse ObjectGzipBufferedFileInputStream nötig

Kompositum

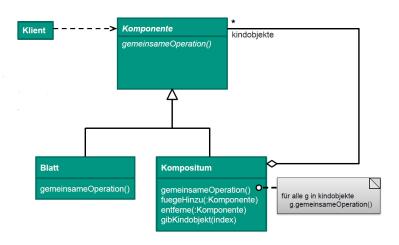


Problem

- Bestandshierarchie modellieren
 - zusammengesetzte und nicht-zusammengesetzte Objekte
 - beide sollen einheitlich behandelt werden können
 - leichtes Hinzufügen neuer Objekte

Kompositum

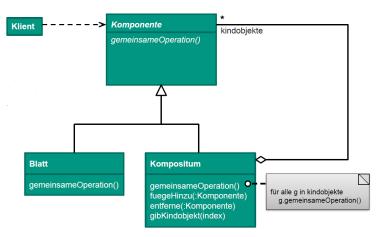




40/66

Kompositum





Beispiel: java.awt.Container#add(Component comp)

Schablonenmethode

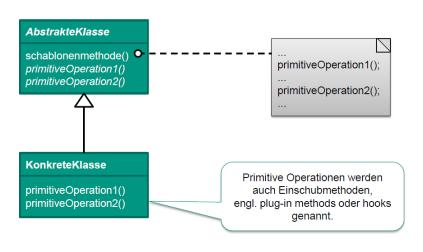


Problem

- eine Methode hat in allen Klassen, die sie implementieren, den gleichen Ablauf
- Teilschritte variieren aber
- nicht komplett in abstrakter Oberklasse implementierbar

Schablonenmethode





Schablonenmethode



Code-Beispiel

Fabrikmethode

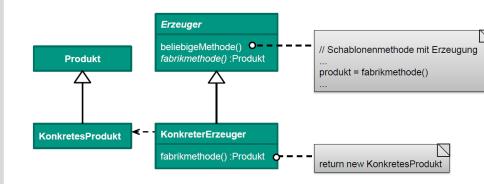


Problem

- Methode soll als Teil-Schritt ein Objekt erzeugen
- Unterklasse soll entscheiden, welches erzeugt wird

Fabrikmethode







Wahr oder falsch?

 Die Fabrikmethode sorgt dafür, dass nur eine einzige Instanz einer Klasse fabriziert wird.

Vermittler

25.06.2019



Wahr oder falsch?

 Die Fabrikmethode sorgt dafür, dass nur eine einzige Instanz einer Klasse fabriziert wird. X



Wahr oder falsch?

- Die Fabrikmethode sorgt dafür, dass nur eine einzige Instanz einer Klasse fabriziert wird. X
- Eine Schablonenmethode ist immer auch eine Fabrikmethode.



Wahr oder falsch?

- Die Fabrikmethode sorgt dafür, dass nur eine einzige Instanz einer Klasse fabriziert wird. X
- Eine Schablonenmethode ist immer auch eine Fabrikmethode. X



Wahr oder falsch?

- Die Fabrikmethode sorgt dafür, dass nur eine einzige Instanz einer Klasse fabriziert wird. X
- Eine Schablonenmethode ist immer auch eine Fabrikmethode. X
- Eine Komponente kann immer nur mit einem einzigen Dekorierer versehen werden.



Wahr oder falsch?

- Die Fabrikmethode sorgt dafür, dass nur eine einzige Instanz einer Klasse fabriziert wird. X
- Eine Schablonenmethode ist immer auch eine Fabrikmethode. X
- Eine Komponente kann immer nur mit einem einzigen Dekorierer versehen werden. X

Kategorien der Entwurfsmuster



- Entkopplungs-Muster ✓
- Varianten-Muster
- 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

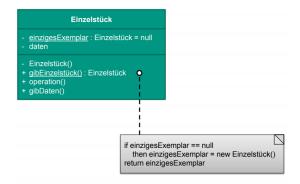
- von einer Klasse soll nur eine Instanz existieren
- Konstruktor könnte überall benutzt werden!

Einzelstück/Singleton



Problem

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



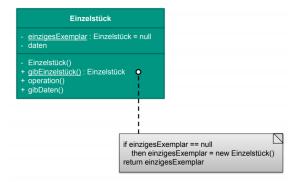
Aufgabe

Einzelstück/Singleton



Problem

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



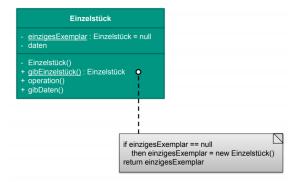
Aber warum nicht einfach statisch?

Einzelstück/Singleton



Problem

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



Aber warum nicht einfach statisch? Unterklassenbildung möglich



Problem

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



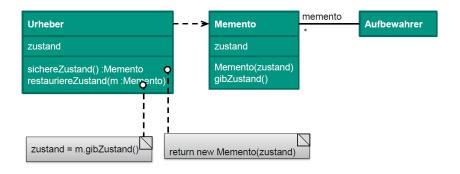
Problem

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



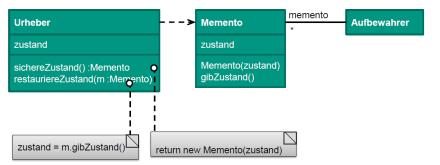
Problem

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



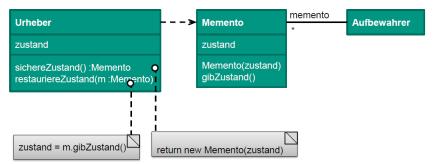
Aufgabe





Problem gelöst?



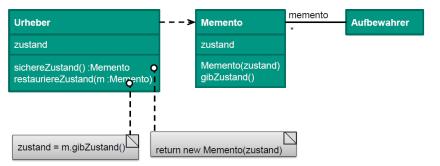


Problem gelöst?

Ja :)



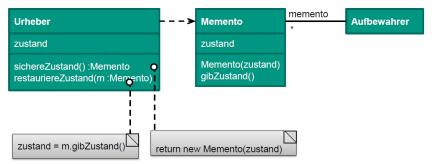




Problem gelöst?

- Ja:)
 - Zustand durch Memento externalisiert





Problem gelöst?

- Ja :)
 - Zustand durch Memento externalisiert
 - Kapselung nicht verletzt (kein direkter Zugriff auf Zustand)
 - Nutzer ruft nur sichereZustand() auf und kriegt neuen Memento

Memento - Beispiel



SuperMario

playerPos: Coordinate

coins: int level: Level

save():Savegame load(s:Savegame)

Savegame

playerPos: Coordinate

coins: int level: I evel

<<create>> Savegame(playerPos: Coordinate, coins:int, level:Level)

getPlayerPos(): Coordinate

getCoins(): int getLevel(): Level

Aufgabe

Kategorien der Entwurfsmuster



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

Steuerungs-Muster



Übergeordnetes Ziel

steuern den Kontrollfluss

Felix Bachmann - SWT1

Steuerungs-Muster



Übergeordnetes Ziel

steuern den Kontrollfluss

⇒ zur richtigen Zeit richtige Methoden aufrufen



Problem

■ Parametrisieren von Objekten mit einer auszuführenden Aktion

Vermittler

Tipps

55/66



Problem

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



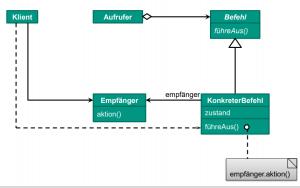
Problem

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

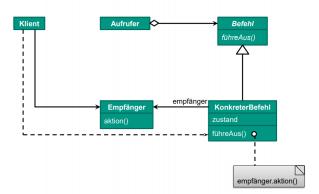


Problem

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



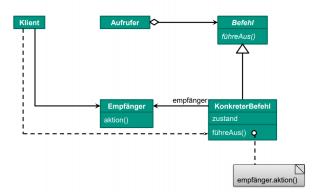




Was haben wir erreicht?

Felix Bachmann - SWT1





Was haben wir erreicht?

 Austauschbarkeit: Befehle unabhängig vom Aufrufer, universell einsetzbar



Code-Beispiel!

Kategorien der Entwurfsmuster



- Entkopplungs-Muster
- Varianten-Muster
- Zustandshandhabungs-Muster
- Steuerungs-Muster
- Bequemlichkeits-Muster
 - (Bequemlichkeits-Klasse)
 - (Bequemlichkeits-Methode)
 - (Fassade)
 - (Null-Objekt)

Kategorien der Entwurfsmuster



- Entkopplungs-Muster <
- Varianten-Muster
- Zustandshandhabungs-Muster
- Bequemlichkeits-Muster <:)



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 oder falsch?

- Bei dem Entwurfsmuster Befehl kennt der Empfänger den Befehl nicht, jedoch der Befehl den Empfänger. ✓
- 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. ✓
- Ein Aufbewahrer im Entwurfsmuster Memento kann beliebig viele Mementos verwalten. Für die Restauration im Falle eines Reset ist er allerdings nicht verantwortlich. ✓

Weitere Beispiele aus Java-Bibliotheken



https://stackoverflow.com/questions/1673841/ examples-of-gof-design-patterns-in-javas-core-libraries/

Für die Klausur



Entwurfsmuster kommen sehr sehr sehr wahrscheinlich dran

Felix Bachmann - SWT1

25.06.2019

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
 - nicht nur die Klassen und Methoden auswendig lernen, sondern das Prinzip verstehen



Aufgabe 1: Architekturstile

- MVC als Schichtenmodell
- wurde mal in der Vorlesung angesprochen, wie man das machen kann



Aufgabe 1: Architekturstile

- MVC als Schichtenmodell
- wurde mal in der Vorlesung angesprochen, wie man das machen kann

Aufgabe 2: Entwurfsmuster in API finden

keine Tipps, sonst zu einfach :)



Aufgabe 1: Architekturstile

- MVC als Schichtenmodell
- wurde mal in der Vorlesung angesprochen, wie man das machen kann

Aufgabe 2: Entwurfsmuster in API finden

keine Tipps, sonst zu einfach :)

Aufgabe 3: Iterator und Besucher selbst schreiben

- Tiefensuche und Breitensuche als Iterator schreiben
- Verarbeitung der Baum-Elemente über Besucher realisieren
- Testfälle diesmal verpflichtend



Aufgabe 4: Code zu Klassendiagramm

• für Umsetzung von Set, Map nochmal Assoziationen anschauen





Aufgabe 4: Code zu Klassendiagramm

• für Umsetzung von Set, Map nochmal Assoziationen anschauen

Aufgabe 5: Zustandsautomat implementieren

- Tests wieder verpflichtend
- Folien zu Zustands-Muster arg knapp
- die im Implementierungskapitel sind deutlich ausführlicher

Felix Bachmann - SWT1



Aufgabe 4: Code zu Klassendiagramm

für Umsetzung von Set, Map nochmal Assoziationen anschauen

Aufgabe 5: Zustandsautomat implementieren

- Tests wieder verpflichtend
- Folien zu Zustands-Muster arg knapp
- die im Implementierungskapitel sind deutlich ausführlicher

Aufgabe 6: Git

- git merge vs. git rebase
- "Merge preserves history whereas rebase rewrites it."
- https://learngitbranching.js.org/
 - interaktives, visuelles Tool zum tieferen Verständnis von git
 - merge und rebase: Level 1.3 und 1.4

Das vorletzte Blatt..



Abgabe

- Deadline am 03.07.19 um 12:00
- Aufgaben 1,2,4,6 handschriftlich

Bis dann! (dann := 09.07.19)



Servicehinweis: Steam Summer Sale ab heute :)

