

Softwaretechnik 1 - 3. Tutorium

Tutorium 17 Felix Bachmann | 04.06.2019

KIT - INSTITUT FÜR PROGRAMMSTRUKTUREN UND DATENORGANISATION (IPD)

Themen



- Evaluation
- 2 Motivation
- 3 Architekturstile
- 4 Java Swing
- 6 Entwurfsmuster
- 6 Ende

Evaluation



Tutorien-Halbzeit erreicht: Her mit der Kritik!



https://forms.gle/N7rjXDUu8BR6gx2KA



• die ersten 2 Phasen des Wasserfallmodells sind geschafft

04.06.2019

Ende



die ersten 2 Phasen des Wasserfallmodells sind geschafft

⇒ Welche waren das nochmal?

Architekturstile



- 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...)
- und welche Phasen gibt es sonst noch?



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

Wozu Entwurf?



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



Softwarearchitektur ist Grundlage für Implementierung!

Entwurf vs Planung/Definition



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

Entwurf vs Planung/Definition



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

Architekturstile



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

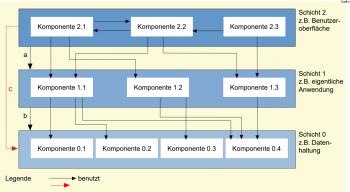
Schichtenarchitektur



- Funktionalität (Pakete, Klassen, ...) in Schichten getrennt
- Schichten "aufeinander gestapelt"
- jede Schicht
 - bietet seine Funktionalität den h\u00f6heren Schichten an
 - nutzt Funktionalität der tieferen Schichten
- Arten von Schichtenarchitekturen
 - intransparent: jede Schicht kann nur auf direkt darunterliegende Schicht zugreifen
 - transparent: jede Schicht kann auch weiter unten liegende Schichten benutzen

Schichtenarchitektur





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

Klient/Dienstgeber und Partnernetze



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

Klient/Dienstgeber und Partnernetze

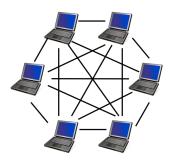


Klient/Dienstgeber



Vor- und Nachteile jeweils?

Partnernetz



Modell-Präsentation-Steuerung



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

Modell-Präsentation-Steuerung



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

Fließband



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



Fließband



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



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

Fließband bei Parallelität



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

	CPU1 (Stufe 1)	CPU2 (Stufe 2)	CPU3 (Stufe 3)
t=0	Х	Х	X
t=1	Daten 1	Х	Х
t=2	Daten 2	Daten 1	х
t=3	Daten 3	Daten 2	Daten 1
t=4	Daten 4	Daten 3	Daten 2
t=5	Daten 5	Daten 4	Daten 3

Rahmenarchitektur



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





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



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

Felix Bachmann - SWT1

04.06.2019



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



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



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



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



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



"Wir wollen unseren customern solutions anbieten, die möglichst extensible sind. Falls die Software successfull ist, können wir später neuen stuff implementen, ohne den alten Code zu ändern. Sheesh."



"Wir wollen unseren customern solutions anbieten, die möglichst extensible sind. Falls die Software successfull ist, können wir später neuen stuff implementen, ohne den alten Code zu ändern. Sheesh."

Framework!



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

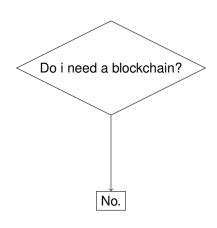


"Auf der letzten Conference waren viele Speaker, die etwas von einer "Blockkette" erzählt haben. Da wir Amazon und Co. nicht trusten, könnten wir sowas usen um unseren Service decentralized anzubieten. Besides können wir damit money einsparen."

Peer-to-Peer

Eine Entwurfsentscheidung:)





Welcher Architekturstil ist gemeint?



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

Felix Bachmann - SWT1

04.06.2019

Welcher Architekturstil ist gemeint?



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

MVC

Java Swing



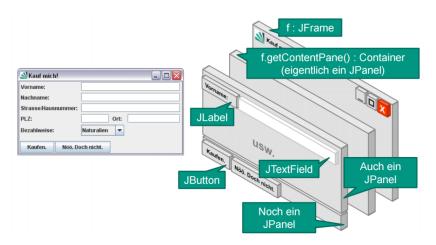
- mit Swing kann man GUIs bauen, ohne was anderes als Java zu benutzen
- graphische Elemente sind als Klassen implementiert

wichtige Klassen und Interfaces

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

Übersicht





Wie erstelle ich meine eigene GUI?



grundsätzlich 2 Möglichkeiten

JFrame erstellen und anpassen

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

Wie erstelle ich meine eigene GUI?



grundsätzlich 2 Möglichkeiten

JFrame erstellen und anpassen

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

von JFrame erben, Anpassung z.B. in Konstruktor

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

Wie erstelle ich eigene Elemente?



ebenfalls 2 Möglichkeiten (JFrame ist auch nur ein Element)

Instanzen erstellen und anpassen

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

Wie erstelle ich eigene Elemente?



ebenfalls 2 Möglichkeiten (JFrame ist auch nur ein Element)

Instanzen erstellen und anpassen

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

Klassen überschreiben, Anpassung z.B. in Konstruktor

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

Link-Sammlung



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

Praxis-Zeit



kleines Beispiel

04.06.2019

Empfehlenswerte Literatur (wirklich!)



knapp 700 Seiten

als interaktives Nachschlagewerk, falls man bestimmte Muster nicht





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 ⇒ 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 ⇒ Rad nicht neu erfinden!



Wozu Entwurfsmuster?



erleichtern Kommunikation

Architekturstile

Ende

Wozu Entwurfsmuster?



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

Geheimnisprinzip



Geheimnis- / Kapselungsprinzip

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

Geheimnisprinzip



Geheimnis- / Kapselungsprinzip

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

Warum eigentlich? Lokalitätsprinzip!

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

Beispiel?

Geheimnisprinzip



Geheimnis- / Kapselungsprinzip

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

Warum eigentlich? Lokalitätsprinzip!

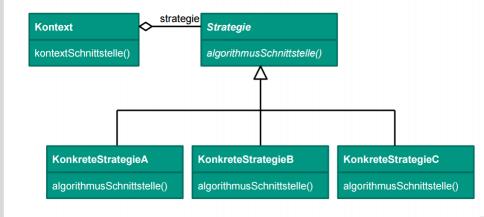
lokale Änderungen sollen sich nicht auf andere Teile auswirken weniger Fehler und Arbeit

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

Vorgriff: Entwurfsmuster Strategie



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



Felix Bachmann - SWT1

04.06.2019



Wahr oder falsch?

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

Felix Bachmann - SWT1

Architekturstile



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.
- 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.
- 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.
- Das Muster Strategie kapselt austauschbares Verhalten und verwendet Delegierung, um zu entscheiden, welches Verhalten verwendet wird. ✓
- Das Hinzufügen einer neuen konkreten Strategie erfordert keine Änderung existierender konkreter Strategien.



Wahr oder falsch?

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

Kategorien der Entwurfsmuster



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

Entkopplungs-Muster



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

Adapter



Problem

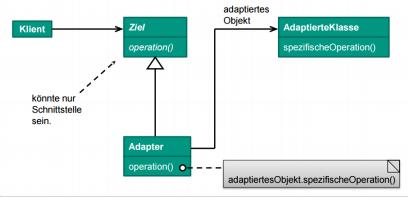
- Klassen mit inkompatiblen Schnittstellen, die wir aber zusammen benutzen wollen
- Schnittstellen nicht änderbar (z.B. externe Bibliotheken)

Adapter



Problem

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

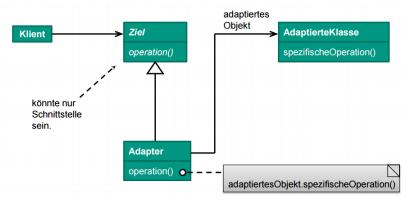


Felix Bachmann - SWT1

04.06.2019

Adapter (Objektadapter)





Wir sind bei Entkopplung-Mustern, Preisfrage:

Wo ist hier die Entkopplung?

Evaluation

Felix Bachmann - SWT1

Motivation

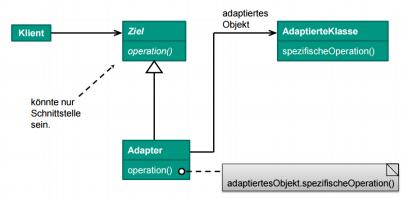
Architekturstile

Java Swing

Entwurfsmuster

Adapter (Objektadapter)





Wir sind bei Entkopplung-Mustern, Preisfrage:

Wo ist hier die Entkopplung?

der Klient ist von der adaptierten Klasse entkoppelt ⇒ austauschbar

Evaluation

Motivation

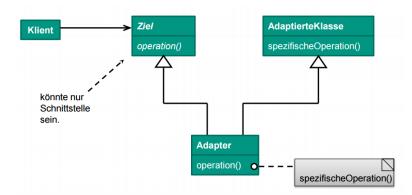
Architekturstile

Java Swing

Entwurfsmuster

Adapter - Alternative (Klassenadapter)





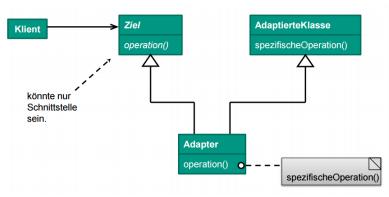
Felix Bachmann - SWT1

04.06.2019

Ende

Adapter - Alternative (Klassenadapter)

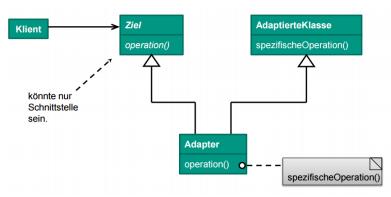




Was für ein Problem bekommt ihr, wenn ihr das auf einem ÜB implementieren müsst?

Adapter - Alternative (Klassenadapter)





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

⇒ keine Mehrfachvererbung in Java!

Felix Bachmann - SWT1

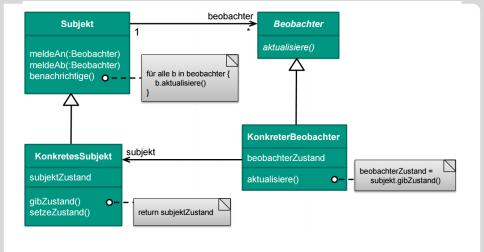
04.06.2019

Beobachter/Observer: abstrakt



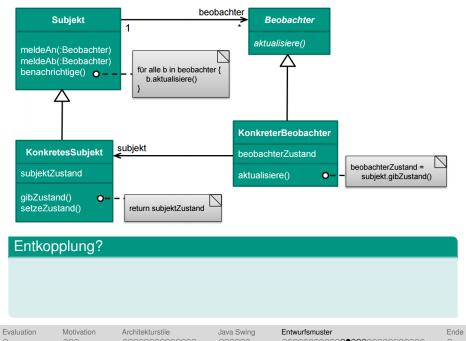
Problem

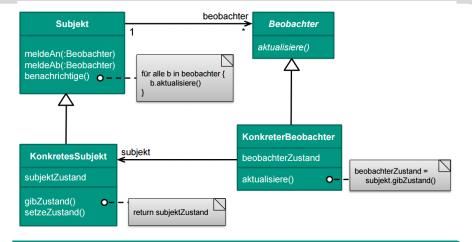
- ein Subjekt, viele Beobachter
- Subjekt ändert Zustand ⇒ Beobachter machen "irgendwas"











Entkopplung?

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

Evaluation 0

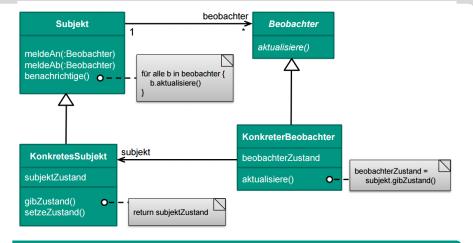
Felix Bachmann - SWT1

Motivation

Architekturstile

Java Swing

Entwurfsmuster

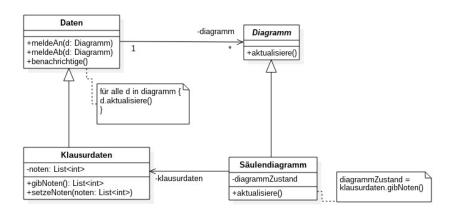


Entkopplung?

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

Beobachter/Observer: am Beispiel





Beobachter=Listener in Java Swing





Beobachter=Listener in Java Swing





Beobachter für einen JButton btn

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

Beobachter=Listener in Java Swing





Das Gleiche mit Lambdas

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

Felix Bachmann - SWT1

04.06.2019



Problem

wollen über Datenstruktur iterieren + Operationen ausführen

Architekturstile



Problem

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

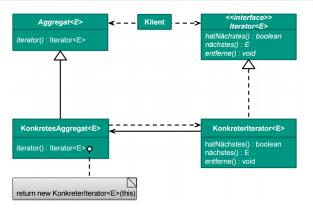
Felix Bachmann - SWT1

43/54

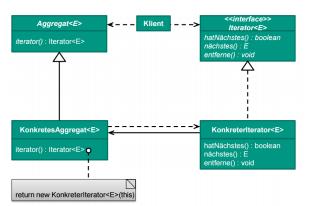


Problem

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







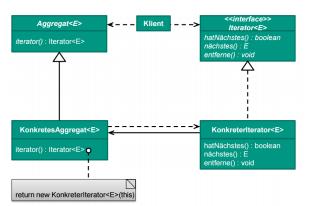
Entkopplung?



Felix Bachmann - SWT1

44/54

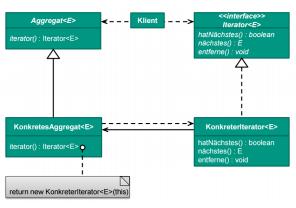




Entkopplung?

- Klient benutzt nur Methoden der Schnittstelle auf dem konkreten Iterator
 - ⇒ Implementierung austauschbar





Beispiel in Java: list.iterator();

Felix Bachmann - SWT1

45/54



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. X



Wahr oder falsch?

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



Problem

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



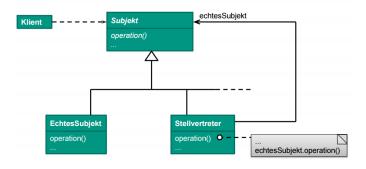
Problem

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

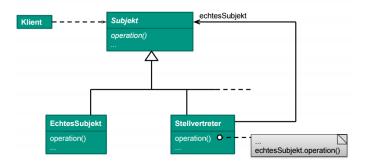


Problem

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

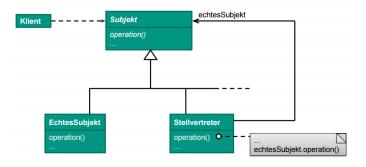






Entkopplung?





Entkopplung?

Klient hat keinen direkten Zugriff auf das echte Subjekt

Felix Bachmann - SWT1

Ende



Problem

mehrere voneinander abhängige Objekte



Problem

mehrere voneinander abhängige Objekte

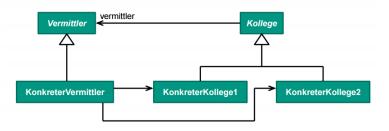
Architekturstile

⇒ Zustände der Objekte von anderen Zuständen abhängig



Problem

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

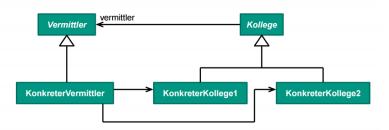






Entkopplung?

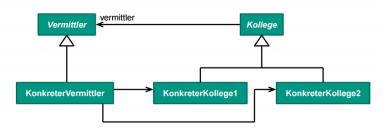




Entkopplung?

Kollegen kennen sich nicht direkt



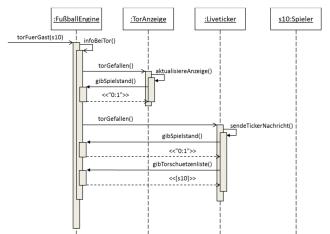


Entkopplung?

- 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?

Evaluation

Motivation

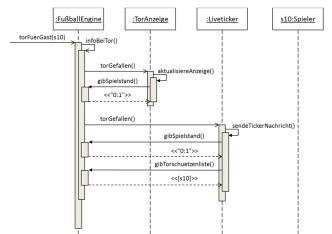
Architekturstile

Java Swing

Entwurfsmuster

Klausuraufgabe (Hauptklausur SS 2012)





Aufgabe 1

Welches Entwurfsmuster erkennen Sie in diesem Diagramm? Beobachter.

Evaluation Felix Bachmann - SWT1

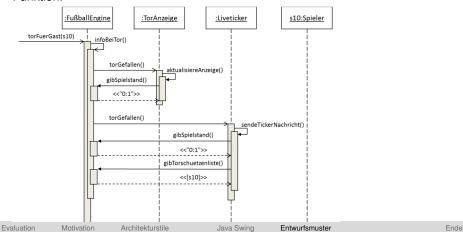
Motivation

Architekturstile

Java Swing

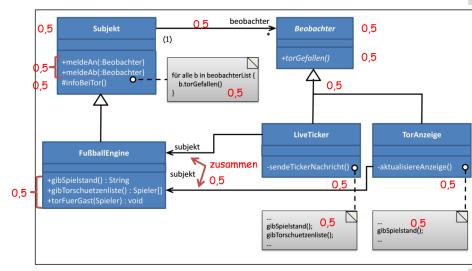
Entwurfsmuster

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.



Musterlösung





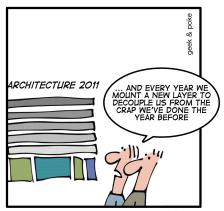
Architekturstile

Bis dann! (dann := 25.06.19)



BEST PRACTICES IN APPLICATION ARCHITECTURE

TODAY: USE LAYERS TO DECOUPLE



ANNUAL RINGS