Woche 07

ISW-Tutorium

Xel Pratscher

Orga

! Stellt Fragen !

- We all want to see you succeed
- Im Testat ist zu spät
- Ihr erreicht mich meistens über Discord
  - Wenn ich nach 6h nicht geantwortet habe schreibt nochmal
  - Auf dem Sever antworten auch andere Tutor:innen

Feedback

- Evaluation bis **18.12.** offen
- erst ab 5 Evaluationen sehe ich Ergebnisse

Weihnachtsmarkt

- 18.12.
- Uhrzeit steht noch nicht fest, wahrscheinlich abends
- Ich schreibe noch eine Mail und auf Discord

Vorlesung

1. Zustandsdiagramme
2. Entwurfsmuster

Zustandsdiagramme

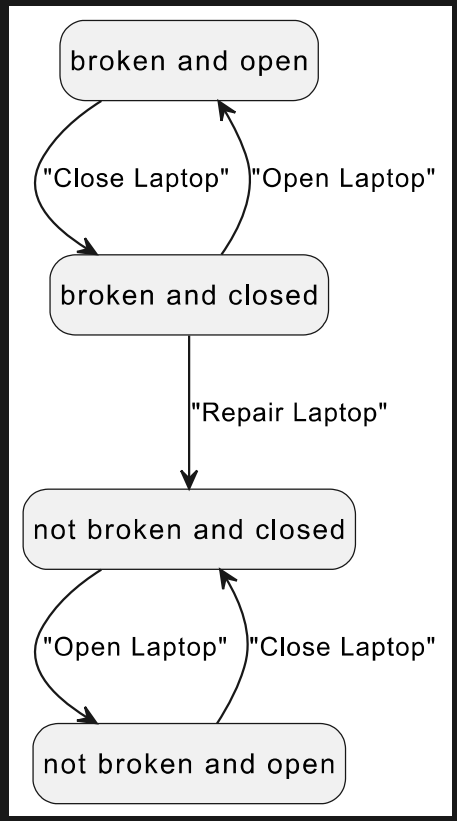
- Beschreiben Verhalten
  - Verhalten Objekt/Klasse
  - Verhalten Komponente/System
  - Verhalten Benutzungsschnittstelle

Wichtige Begriffe

- **Verhalten** = Menge der möglichen Zustandsfolgen
- **Zustand** charakterisiert durch Bedingung über Daten
- **Transition** = Übergang zw. Zuständen

Beispiel Aufbau

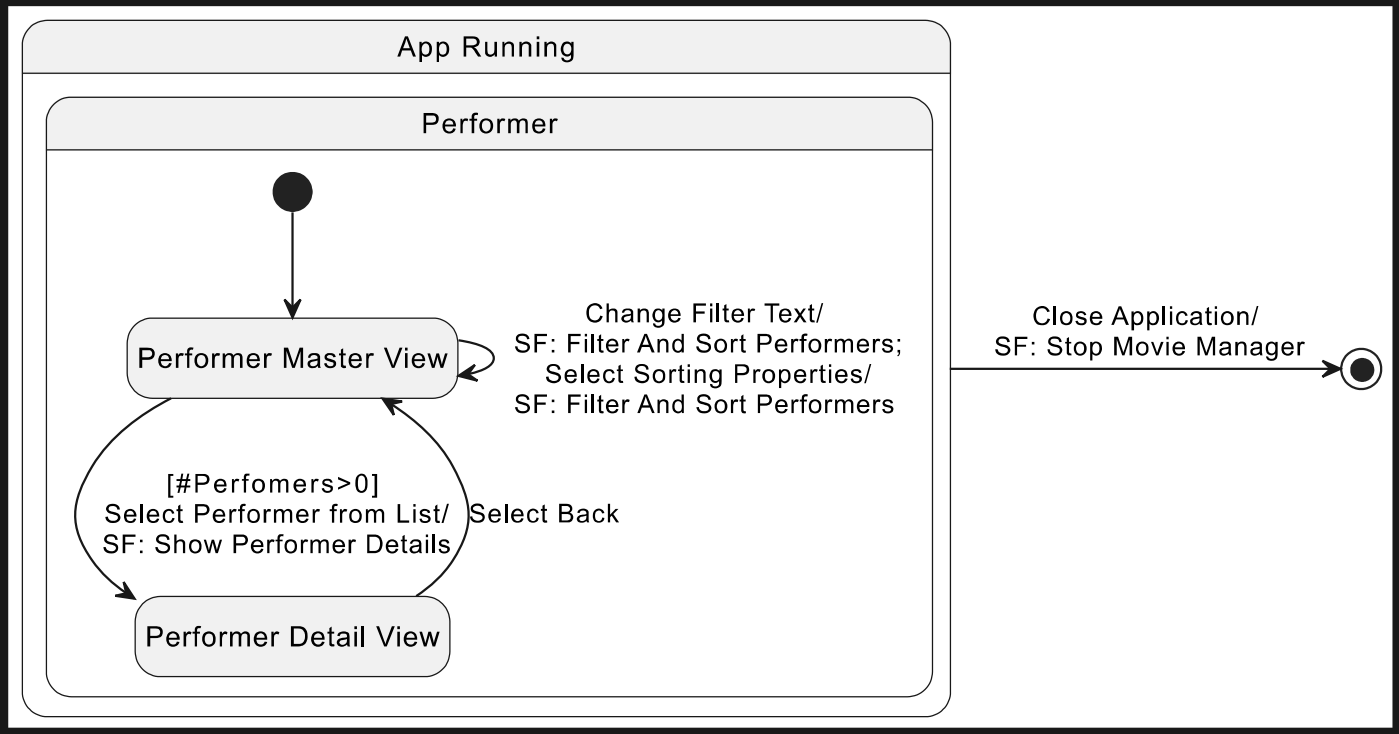
Laptop kann kaputt und/oder aufgeklappt sein



Dialogmodell

- **Dialog** beschreibt Abfolge von Sichten bei Aufgabendurchführung
- **Zustand** = Workspaces
- **Transitionen** = Funktionsausführung
- **Ziel:** Möglichst *wenige* Sichten (niedrige Kopplung), möglichst viel Wiederverwendung

Beispiel


App Running
Performer
Performer Master View
Change Filter Text/
SF: Filter And Sort Performers;
Select Sorting Properties/
SF: Filter And Sort Performers
Close Application/
SF: Stop Movie Manager
[#Perfomers>0]
Select Performer from List/
SF: Show Performer Details
Select Back
Performer Detail View

H vs H*

- **H:** Historie, letzter Oberzustand
- **H*:** letzter innerer Zustand

Entwurfsmuster

- **Muster** = schematische Lösung für verwandte Probleme
- **Arten**
  - Erzeugungsmuster
  - Strukturmuster
  - Verhaltensmuster

Vorteile

- Wiederverwendung bewährter Lösungen
- bessere Lesbarkeit/Wartbarkeit
- einfachere Kommunikation

Nachteile

- Anwendung von Muster im falschen Kontext
  - Overhead, schlechtere Lesbarkeit/Wartbarkeit

Beschreibung

- Name
- Problem
- Lösung
  - Struktur
  - Bestandteile
  - Objektinteraktion
- Diskussion
  - Vor-/Nachteile
  - Abhängigkeiten, Einschränkungen
  - Spezialfälle
  - Bekannte Verwendung

Erzeugungsmuster

- Behandeln Erzeugung von Objekten
- Versuchen Erzeugung zu verstecken/vereinheitlichen/vereinfachen

Beispiele

- Singleton (Einzelstück)
- Abstrakte Oberklasse
- Factory Method -> Technologie-VL
- Builder -> Technologie-VL

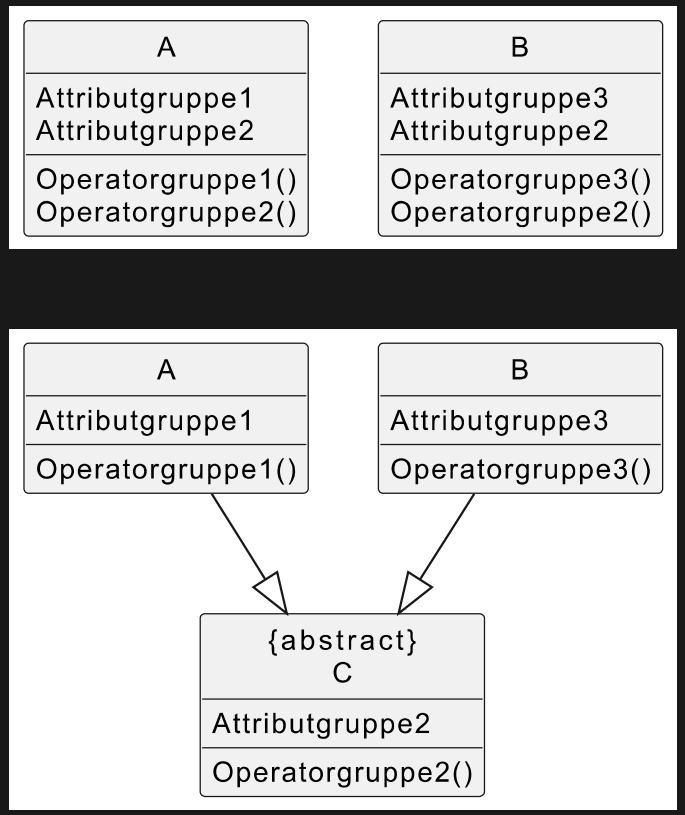
Singleton (Einzelstück)

- **Problem:** Von Klasse max. eine Instanz
- **Lösung:** Klasse aufbauen nach Singleton-Schema

```
1  Public final class Singleton {
2      private static Singleton instance;
3      private Singleton()
4      public static Singleton getInstance() {
5          if (instance==null) {
6              instance = new Singleton();
7          }
8          return instance;
9      }
10 }
```

Abstrakte Oberklasse

- **Problem:** Klassen enthalten Gruppen identischer Attribute / Operationen
- **Lösung:** identische Bestandteile in abstrakte Oberklasse separieren



Strukturmuster

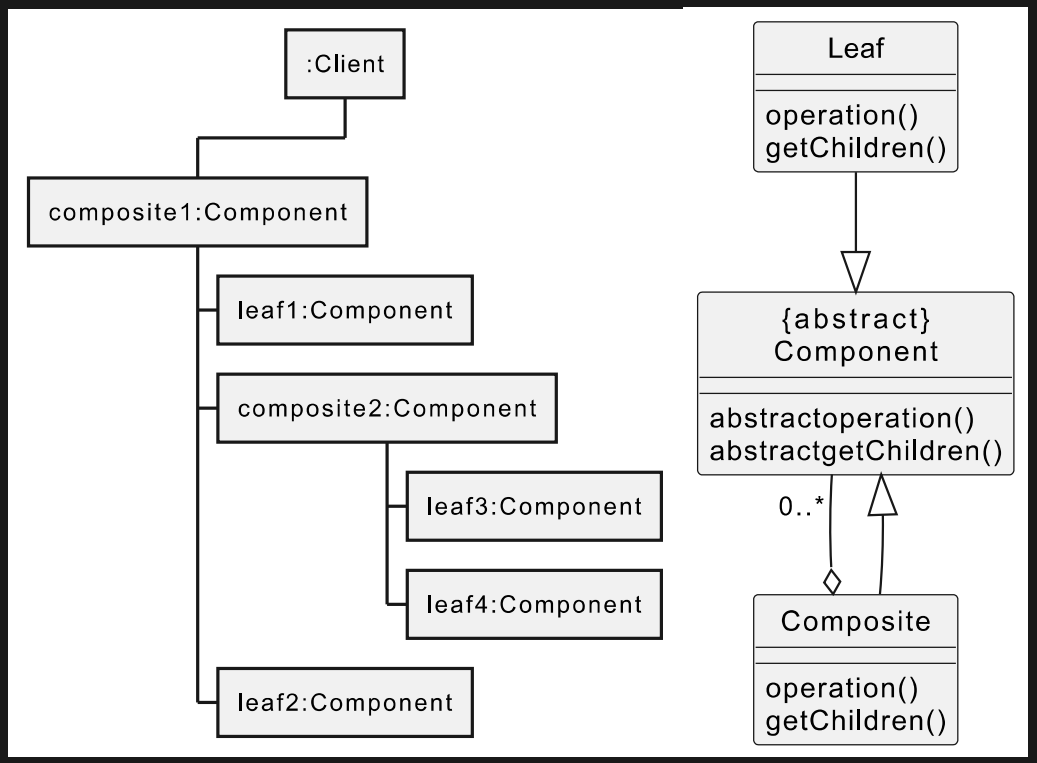
- Zusammensetzung/Beziehungen von Klassen
- Bildung größerer Strukturen

Beispiele

- Kompositum
- Adapter -> Technologie-VL
- Proxy -> Technologie-VL

Kompositum

- **Problem:** Modellierung + Umsetzung hierarchischer Strukturen
- **Lösung:** Einheitliche abstrakte Oberklasse (Component)



Mehr zu Kompositum

- Wird genutzt, wenn der Client den Unterschied zw. Composites und Leafs ignoriert nutzen
- Als Beispiel schaut euch ein Filesystem an
  - Ordner (Composites) – Files (Leafs)

Verhaltensmuster

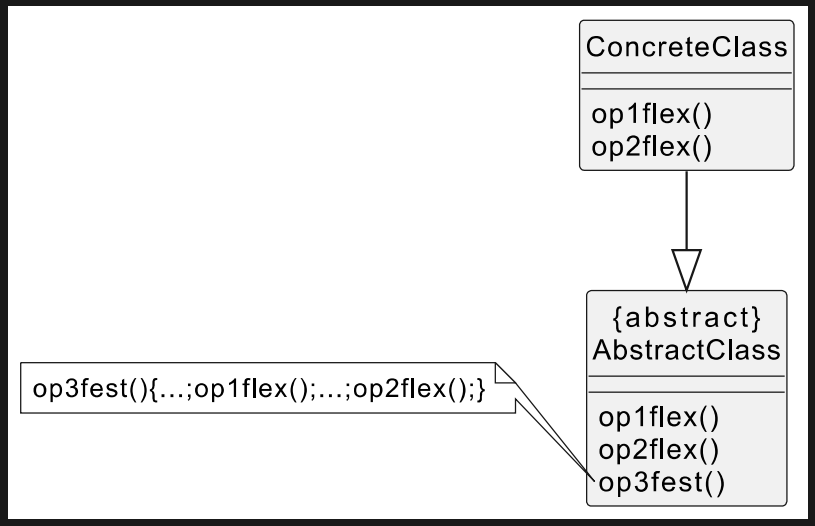
- Algorithmen und Zuweisung von Zuständen zu Objekten
- Beschreiben auch Interaktion Klassen/Objekte

Beispiel

- Template Method
- Zuständigkeitskette -> Technologie-VL
- Beobachter -> Technologie-VL
- Strategie -> Technologie-VL
- Besucher -> Technologie-VL

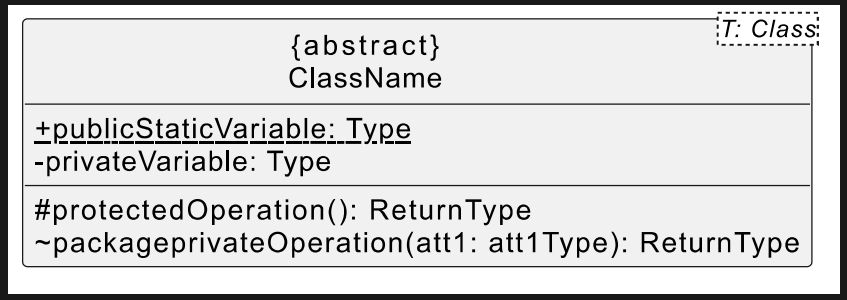
Template Method

- **Problem:**
  Operation besteht aus festen und veränderlichen
- **Lösung:**
  - Definieren von Templates
  - Algorithmus in Unterklassen beschreiben
  - feste Bestandteile in Oberklasse



Einschub Klassendiagramme

Aufbau einer Klasse



Interface

-
- Referenztyp in Java
- Sammlung abstrakter Methoden
- **kein Konstruktor**, keine Instanzen
- Interface wird nicht geerbt, nur implementiert
- Klasse kann mehrere Interfaces erben
- Nur public visibility

```
1 public interface NameOfInterface {
2 }
```

Übungsblatt 08

8.1 (Testat, Einzeln)

- Dialogmodell vervollständigen
- Tabelle ausfüllen und erweitern
- H vs H*

8.2 (Testat, Team)

- Entwurfsmuster im Code finden
- Andere Beispiele als in Technologie-VL finden

8.3 (kein Testat, Team)

- Entwurfsmuster bewerten

8.4 (Testat, Einzeln)

- Fragen unter eurem Namen bearbeiten
- Partner 1: Fragen beantworten
- Partner 2: Antworten kontrollieren
- DANACH prüfe ich nach

8.5 (kein Testat, Einzeln)

- Vorbereitung Inspektion

Danke fürs Zuhören und Mitmachen :)