Karlsruher Institut für Technologie

Tutorium Softwaretechnik I

Moritz Klammler | **27. Juni 2017**

Fakultät für Informatik, IPD Tichy

Themen des Tages

- Übungsblatt 3
- Übungsblatt 4
- Allgemeines zum Entwurf
- Architekturstile
- Entwurfsmuster

Übungsblatt 3 – Allgemeine Anmerkungen

- Laut Übungsleitern ist es egal beziehungsweise sogar gewünscht, dass `mvn package` nicht funktioniert.
- Exception-Handling ist bei vielen noch immer verbesserungsfähig.
- Bitte lest die DocStrings der Methoden, die Ihr implementieren sollt, und implementiert sie im Einklang mit der Dokumentation.
- Den `Comparator` für `PluginPriority` kann man mehr oder weniger elegant implementieren.

Übungsblatt 4 – Allgemeine Anmerkungen

- Exception-Handling ist bei vielen noch immer verbesserungsfähig.
- GUI-Programmierung mit Swing ist nicht gerade lustig, aber eine lehrreiche Erfahrung. Wer das Blatt nicht bearbeitet hat, sollte sich dennoch mit der Technologie vertraut machen.
- Testet Eure Lösungen vor der Abgabe um zumindest triviale Fehler erkennen und fixen zu können.

Was zeichnet einen guten Entwurf aus?

Starke Kohäsion innerhalb eines Moduls

Was zeichnet einen guten Entwurf aus?

Karlsruher Institut für Technologie

- ✓ Starke Kohäsion innerhalb eines Moduls

Was zeichnet einen guten Entwurf aus?

- ✓ Starke Kohäsion innerhalb eines Moduls
  Lose Kopplung zwischen Modulen

Was zeichnet einen guten Entwurf aus?

- ✓ Starke Kohäsion innerhalb eines Moduls
- ✓ Lose Kopplung zwischen Modulen

Was zeichnet einen guten Entwurf aus?

- ✓ Starke Kohäsion innerhalb eines Moduls
- ✓ Lose Kopplung zwischen Modulen
- Zyklische Abhängigkeiten zwischen Modulen

Was zeichnet einen guten Entwurf aus?

- ✓ Starke Kohäsion innerhalb eines Moduls
- ✓ Lose Kopplung zwischen Modulen
- ✗ Zyklische Abhängigkeiten zwischen Modulen

Was zeichnet einen guten Entwurf aus?

Karlsruher Institut für Technologie

- ✓ Starke Kohäsion innerhalb eines Moduls
- ✓ Lose Kopplung zwischen Modulen
- ✗ Zyklische Abhängigkeiten zwischen Modulen

```
widget.getGadgets().add(new Gadget())
```

Was zeichnet einen guten Entwurf aus?

- ✓ Starke Kohäsion innerhalb eines Moduls
- ✓ Lose Kopplung zwischen Modulen
- ✗ Zyklische Abhängigkeiten zwischen Modulen
- ✗ `widget.getGadgets().add(new Gadget())`

Was zeichnet einen guten Entwurf aus?

- ✓ Starke Kohäsion innerhalb eines Moduls
- ✓ Lose Kopplung zwischen Modulen
- ✗ Zyklische Abhängigkeiten zwischen Modulen
- ✗ `widget.getGadgets().add(new Gadget())`

  Low-level Code ruft high-level Code direkt auf.

Was zeichnet einen guten Entwurf aus?

- ✓ Starke Kohäsion innerhalb eines Moduls
- ✓ Lose Kopplung zwischen Modulen
- ✗ Zyklische Abhängigkeiten zwischen Modulen
- ✗ `widget.getGadgets().add(new Gadget())`
- ✗ Low-level Code ruft high-level Code direkt auf.

Architekturstile

- Schichten (*layer*)
- Klient-Dienstgeber (*client-server*)
- Partnernetze (*peer-to-peer networks*)
- Datenablage (*repository*)
- Modell-Präsentation-Steuerung (*model-view-controller*)
- Fließband (*pipeline*)
- Rahmenarchitektur + Einschübe (*framwork + plugins*)
- Dienstorientierte Architektur (*service-oriented architecture*)

Architekturstile

- Schichten (*layer*)
  $\rightarrow$ Bsp: File-I/O
- Klient-Dienstgeber (*client-server*)
- Partnernetze (*peer-to-peer networks*)
- Datenablage (*repository*)
- Modell-Präsentation-Steuerung (*model-view-controller*)
- Fließband (*pipeline*)
- Rahmenarchitektur + Einschübe (*framwork + plugins*)
- Dienstorientierte Architektur (*service-oriented architecture*)

Architekturstile

- Schichten (*layer*)
- Klient-Dienstgeber (*client-server*)
  $\rightarrow$ Bsp: Keyring
- Partnernetze (*peer-to-peer networks*)
- Datenablage (*repository*)
- Modell-Präsentation-Steuerung (*model-view-controller*)
- Fließband (*pipeline*)
- Rahmenarchitektur + Einschübe (*framwork + plugins*)
- Dienstorientierte Architektur (*service-oriented architecture*)

Architekturstile

- Schichten (*layer*)
- Klient-Dienstgeber (*client-server*)
- Partnernetze (*peer-to-peer networks*)
  $\rightarrow$ Bsp: Tor
- Datenablage (*repository*)
- Modell-Präsentation-Steuerung (*model-view-controller*)
- Fließband (*pipeline*)
- Rahmenarchitektur + Einschübe (*framwork + plugins*)
- Dienstorientierte Architektur (*service-oriented architecture*)

Architekturstile

- Schichten (*layer*)
- Klient-Dienstgeber (*client-server*)
- Partnernetze (*peer-to-peer networks*)
- Datenablage (*repository*)
  $\rightarrow$ Bsp: HTTP REST
- Modell-Präsentation-Steuerung (*model-view-controller*)
- Fließband (*pipeline*)
- Rahmenarchitektur + Einschübe (*framwork + plugins*)
- Dienstorientierte Architektur (*service-oriented architecture*)

Architekturstile

- Schichten (*layer*)
- Klient-Dienstgeber (*client-server*)
- Partnernetze (*peer-to-peer networks*)
- Datenablage (*repository*)
- Modell-Präsentation-Steuerung (*model-view-controller*)
  $\rightarrow$ Bsp: Git
- Fließband (*pipeline*)
- Rahmenarchitektur + Einschübe (*framwork + plugins*)
- Dienstorientierte Architektur (*service-oriented architecture*)

Architekturstile

- Schichten (*layer*)
- Klient-Dienstgeber (*client-server*)
- Partnernetze (*peer-to-peer networks*)
- Datenablage (*repository*)
- Modell-Präsentation-Steuerung (*model-view-controller*)
- Fließband (*pipeline*)
  $\rightarrow$ Bsp: Print-Server
- Rahmenarchitektur + Einschübe (*framwork + plugins*)
- Dienstorientierte Architektur (*service-oriented architecture*)

Architekturstile

- Schichten (*layer*)
- Klient-Dienstgeber (*client-server*)
- Partnernetze (*peer-to-peer networks*)
- Datenablage (*repository*)
- Modell-Präsentation-Steuerung (*model-view-controller*)
- Fließband (*pipeline*)
- Rahmenarchitektur + Einschübe (*framwork + plugins*)
  $\rightarrow$ Bsp: Maven
- Dienstorientierte Architektur (*service-oriented architecture*)

Architekturstile

- Schichten (*layer*)
- Klient-Dienstgeber (*client-server*)
- Partnernetze (*peer-to-peer networks*)
- Datenablage (*repository*)
- Modell-Präsentation-Steuerung (*model-view-controller*)
- Fließband (*pipeline*)
- Rahmenarchitektur + Einschübe (*framwork + plugins*)
- Dienstorientierte Architektur (*service-oriented architecture*)
  $\rightarrow$ Bsp: So ziemlich jede App auf Eurem SmartPhone...

Architekturstile

- Schichten (*layer*)
- Klient-Dienstgeber (*client-server*)
- Partnernetze (*peer-to-peer networks*)
- Datenablage (*repository*)
- Modell-Präsentation-Steuerung (*model-view-controller*)
- Fließband (*pipeline*)
- Rahmenarchitektur + Einschübe (*framwork + plugins*)
- Dienstorientierte Architektur (*service-oriented architecture*)

Design Patterns (Gang of Four)

- ***Creational Patterns***
  - *Abstract Factory*
  - *Builder*
  - *Factory Method*
  - *Prototype*
  - *Singleton*
- ***Structural Patterns***
  - *Adapter*
  - *Bridge*
  - *Composite*
  - *Decorator*
  - *Facade*
  - *Flyweight*
  - *Proxy*
- ***Behavioral Patterns***
  - *Chain of Responsibility*
  - *Command*
  - *Interpreter*
  - *Iterator*
  - *Mediator*
  - *Memento*
  - *Observer*
  - *State*
  - *Strategy*
  - *Template Method*
  - *Vistor*

Entwurfsmuster (Vorlesung)

- **Entkoppelungsmuster**
  - Adapter
  - Beobachter
  - Brücke
  - Iterator
  - Stellvertreter
  - Vermittler
- **Variantenmuster**
  - Abstrakte Fabrik
  - Besucher
  - Erbauer
  - Fabrikmethode
  - Kompositum
  - Schablonenmethode
  - Strategie
  - Dekorierer
- **Zustandshandhabungsmuster**
  - Einzelstück
  - Fliegengewicht
  - Memento
  - Prototyp
  - Zustand
- **Steuerungsmuster**
  - Befehl
  - Master-Worker
- **Virtuelle Maschinen**
  - Interpretierer
- **Bequemlichkeitsmuster**
  - Bequemlichkeitsklasse
  - Bequemlichkeitsmethode
  - Fassade
  - Null-Objekt