RWTH AACHEN UNIVERSITY

Komponentenbasierte Softwareentwicklung

Jens Hollmann

Rechen- und Kommunikationszentrum (RZ)

Übersicht

- **Motivation von Komponenten**
- **Komponenten durch Ableiten**
  - → JavaBeans
  - → Reflection
  - → Applet (Lifecycle)
- **3-tier Architektur**
- **Annotations**
- **Enterprise JavaBeans**
  - → Stateless, Statefull, Message Driven Beans
  - → Persistenz

Organisatorisches

- **Source-Code-Schnipsel, Abgabe von Hausaufgaben in Ilias**
  https://www.ili.fh-aachen.de/
  Anmelden am Kurs mit Passwort „`Komp,2012`“

Motivation von Komponenten

RWTH AACHEN UNIVERSITY

Monolithische Software

- **Start-Eintrittspunkt (main)**

```
public static void main(…) {
   MyClass mc = new MyClass();
   mc.doIt();
}
```

- **Kann Module haben, multithreaded sein, etc.**
- **Erweiterung heißt meistens: Alles Ausliefern**

Motivation von Komponenten

Beispiel: Software mit Berechnung von Steuersätzen

Einführung einer neuen Steuer:

- **Software-Entwicklung**
  - → Erweiterung der Auftragsbearbeitung um neue Steuer
- **Test**
  - → Sicherheitshalber alles testen
- **Ausliefern der neuen Software**
  - → Evtl. Erweiterung der Konfiguration (Benutze Steuer XYZ mit Steuersatz 6%)

Motivation von Komponenten

Problem: Ausliefern der neuen Software

- **Kunde glaubt nicht, dass eine neue Software keine neuen Fehler mitbringt**
- **Und er hat Recht!**
- **Ausliefern bedeutet auch: Neuentwicklungen bis zu diesem Zeitpunkt kommen mit**
- **Hoher Aufwand für Tests, Konfiguration (so dass neue Funktionen evtl. nicht aktiv werden), eigene Branches für jeden Kunden/Auslieferung/Patch etc.**

Motivation von Komponenten

RWTH AACHEN UNIVERSITY

Schöner wäre:

- **Entwickeln einer „Komponente“ für die neue Steuer**
- **Die Software ist so geschrieben, dass man in einer Konfiguration eintragen kann, welche Komponenten zur Steuerberechnung benutzt werden**
- **Test der neuen Komponente**
- **Ausliefern nur der neuen Komponente**
- **Konfiguration des Systems, so dass die neue Steuerberechnung mit Steuersatz 6% benutzt wird.**

```
<Auftragsbearbeitung>
  <Steuern>
    <Mehrwertsteuer steuersatz=0.19/>
    <NeueSteuer steuersatz=0.06/>
  </Steuern>
</Auftragsbearbeitung>
```

Motivation von Komponenten

„Software aus Legosteinen“

- **Kann ein bisschen funktionieren**
- **Analogie ist sehr treffend**

  → Bausteine werden so einfach, dass der Zusammenbau entsprechend kompliziert wird

- **Ein wesentlicher Teil guter Software sind gut definierte Schnittstellen**

Motivation von Komponenten

Es geht bei Komponenten um diese Dinge:

1. **Sie exportiert/implementiert eine oder mehrere Schnittstellen**
   - → Kein Eintrittspunkt (main), sondern die exportierte Schnittstelle wird von einem Framework benutzt
2. **Sie importiert andere Schnittstellen**
   - → Optional
3. **Versteckt die Implementierung**
   - → Dadurch kann eine Komponente durch eine andere (mit derselben exportierten/implementierten Schnittstelle ersetzt werden)
4. **Eignet sich zur Wiederverwendung**
   - → Sonst muss man keine Komponente daraus machen

Komponenten durch Ableiten: JavaBeans

RWTH AACHEN UNIVERSITY

JavaBeans

- **Software-Komponenten für GUI-Klassen**
- **Abgeleitet von JComponent**
  - → Die öffentlichen Methoden von JComponent bilden die exportierte Schnittstelle der Komponente
  - → Das Framework (Swing) benutzt diese Schnittstelle
- **In jedem GUI wieder verwendbar**

Komponenten durch Ableiten: JavaBeans

RWTH AACHEN UNIVERSITY

Übung

- **Schreiben einer JavaBean FaceBean**
  - → Malt einen Smiley
  - → Property MouthWidth
  - → Funktionen smile/frown
  - → Verbinden der MouthWidth mit einem Slider
  - → Verbinden der Funktionen smile/frown mit Radio-Buttons

Komponenten durch Ableiten: JavaBeans

- **New Java Project**
- **New Swing Application**
- **Im Editor: Auf Design-Ansicht wechseln**
- **Eigene Bean:**
  - → Neue Java-Klasse (Abgeleitet von JComponent)
  - → Überschreiben: paint
  - → Neu: get/setMouthWidth, smile/frown
  - → Source für paint unter https://www.ili.fh-aachen.de/