E Java & Komponentenmodelle & Jini

E.1 Überblick

- Software-Komponenten
- Komponentenmodelle
- Java & Komponentenmodelle: JavaBeans
  - Architektur
  - Properties
  - Ereignisse (Events)
  - Introspection
- Jini
  - ein verteiltes Komponentenmodelll
- Anwendbarkeit von Software-Komponenten

E.2 Literatur

Edw99. W. Keith Edwards. *Core Jini*. Prentice Hall, Upper Saddle River, NJ, 1999.

Edw00. W. Keith Edwards. *Core Jini*. Prentice Hall, München, 2000 (deutsche Übersetzung, umfasst bereits Jini 1.1!).

Szy98. Clemens Szyperski. *Component Software — Beyond Object-Oriented Programming*. Addison-Wesley, Harlow, England, 1998.

Grif98. Frank Griffel. *Componentware — Konzepte und Techniken eines Softwareparadigmas.* dpunkt-Verlag, Heidelberg, 1998.

JavaBeans. http://www.sun.com/beans/

Jini. http://www.sun.com/jini/specs/

E.3 Komponentenmodelle

1 Software Komponenten

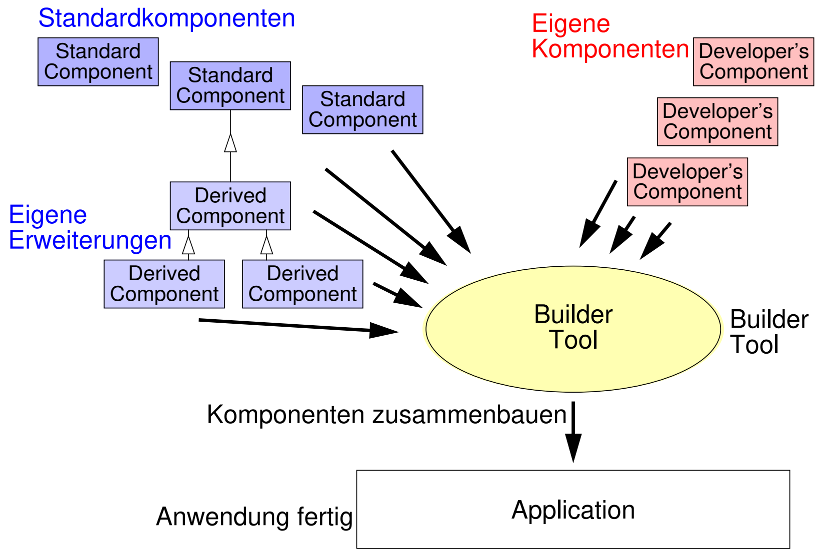
- Definition (Szyperski)
  Eine Softwarekomponente ist ein wiederverwendbares Stück Software
  - ➤ hat eine gut spezifizierte Schnittstelle
  - ➤ kann in nicht-vorhergeplanten Kontexten eingesetzt werden
  - ➤ ist eine eigenständig vermarktbare Einheit
- Beispiele
  - ➤ Elemente einer Fenster-Oberfläche
  - ➤ Software-Modul, das ein Gerät repräsentiert (Messgerät, Drucker, …)
  - ➤ Rechtschreibprüfungs-Modul für Textverarbeitungssysteme

? also einfach Klassen im objektorientierten Sinn — oder was ist da neu dran

2 Softwarekomponenten (2)

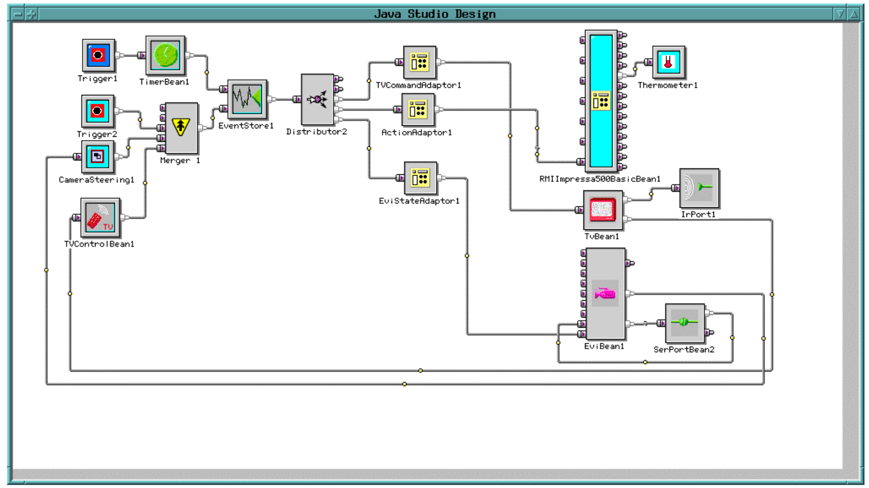
- Standard-Konventionen für die Schnittstellen von Software-Komponenten
  - ⬆ **Komponenten-Modell**
  - ➥ Einfache Verwendbarkeit + Wiederverwendbarkeit
- Software-Komponenten sind selbst-beschreibend
  - ➤ Automatische Analyse von Schnittstelle und Eigenschaften möglich
    - ➤ Introspection / Reflection - Mechanismen
    - ➤ Interface-Repository
    - + Namens-Konventionen
- Zusammenstellung zu komplexen Anwendungen mit grafischen Werkzeugen (Builder-Tools)
  - ➤ Software-Baukasten

3 Philosopie



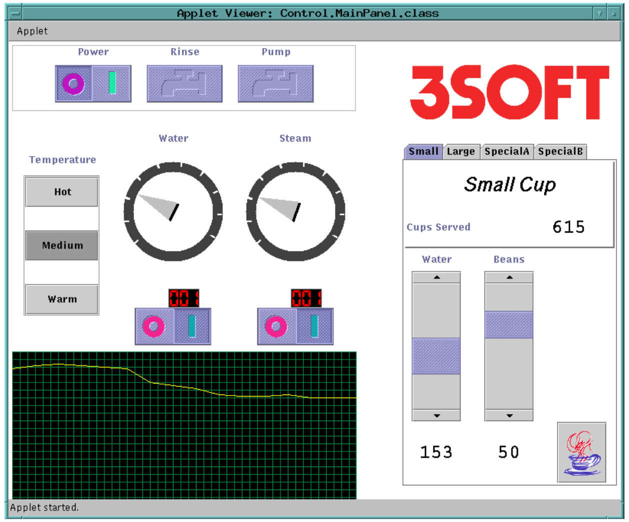
4 Beispiel für ein Builder Tool

- JavaStudio



5 Beispiel für eine Anwendung

- Bedienoberfläche unserer Kaffeemaschine



6 Komponentenmodell

- Richtlinien für Software-Komponenten
  - ➤ Namenskonventionen
  - ➤ Repositories, Introspection-Mechanismus
  - ➤ Schnittstellen-Semantik
  - ➤ Programmiermethodik
- Existierende Komponentenarchitekturen
  - ◆ JavaBeans
  - ◆ CORBA Component Model
  - ◆ ActiveX
    - ➤ nicht portabel, proprietär
  - ◆ OpenDOC
    - ➤ ziemlich tot
  - ◆ Hersteller-spezifische Speziallösungen
    - ➤ GUI-Builder Klassenbibliotheken

E.4 Java & Softwarekomponenten

1 Java — Ziele

- Lösung der verbreiteten Probleme bei der Entwicklung und Verteilung von Software
  - ◆ verschiedene Betriebssysteme (Unix, Windows, MacOS, …)
  - ◆ verschiedene Hardware-Architekturen
- Java: Sprache und Ausführungsumgebung für *sichere*, *schnelle* und *sehr robuste* Anwendungen auf *verschiedenen Plattformen* in *heterogenen verteilten Netzen*

2 Java — wesentliche Eigenschaften für Komponenten

- objektorientiert
- Polymorphismus bei Methodenaufrufen
- sehr flexibel (dynamisches Laden und Binden)
- Reflection/Introspection Mechanismus

3 Java-Komponentenmodelle

- JavaBeans
- Jini

E.5 JavaBeans

1 Definition

- JavaBeans ist eine Schnittstellen-Spezifikation
  für wiederverwendbarer Software-Komponenten in Java
  - ➤ definiert die Java-Komponenten
  - ➤ und wie sie zusammenarbeiten
- Eine Bean ist eine beliebige Java-Klasse,
  die den JavaBeans-Konventionen folgt
- Die offizielle Definition:

  A Java Bean is a reusable software component that can be visually manipulated in builder tools.

2 Architektur

- Eigenschaften (Properties)
  - ➤ initiale Einstellung von Eigenschaften (Zustand) einer Bean
- Methoden
  Ereignisse (Events)
  - ➤ die Verknüpfungspunkte für Beans
- Adapter
  - ➤ zur Schnittstellenanpassung, wenn Beans nicht einfach zusammenpassen
- Introspection
  - ➤ statt eines Interface-Repositories:
    einfach in die Bean reinschauen was sie kann

3 Beispiele

- Grafische Beans einer Benutzeroberfläche
  - ➤ Knöpfe, Regler, Textfenster
  - ➤ HTML rendering Bean
  - ➤ OpenGL canvas
- Unsichtbare Beans
  - ➤ Datenbank-Schnittstelle
  - ➤ Timer
    - – löst Ereignisse in bestimmten Abständen aus
    - – kann komplexe Zeit-Logik kapseln
- Softwarekomponenten, die Geräte/Geräteteile repräsentieren
  - ➤ Schalter
  - ➤ Sensoren, Aktuatoren
  - ➤ Video-Rekorder
  - ➤ Kaffeemaschine

4 Properties

- Beschreiben Eigenschaften von Komponenten
- Jede Property hat
  - ➤ einen **Namen** — symbolische, aussagekräftige Beschreibung der Eigenschaft (Color, Font, …)
    `private Color color;` (Instanzvariable der Bean-Klasse)
  - ➤ einen **Typ** —Java-Klasse, kapselt den Wert
  - ➤ Constraints — optionale Einschränkungen (z. B. read-only)
- Namens-Konventionen für Methoden zum Zugriff (Methoden der Beam-Klasse)
  - ➤ get-Methode zum Lesen

```
public Color getColor(){ return color; }
```

  - ➤ set-Methode für Modifikationen

```
public void setColor(Color newColor){
        color = newColor;
        repaint();
}
```