Karlsruher Institut für Technologie

Tutorium Softwaretechnik I

Moritz Klammler | **11. Juli 2017**

Fakultät für Informatik, IPD Tichy

Themen des Tages

- Übungsblatt 5
- Implementierung
  - Endliche Automaten
  - Innere Klassen in Java
- Grundlegende Konzepte der parallelen Programmierung
- Falls gewünscht: Wiederholung Entwurfsmuster

Implementierung von endlichen Automaten

- `switch-case`
- Tabellengesteuert
- *State*-Pattern

Implementierung von endlichen Automaten

Ein Musikspieler hat zwei Tasten für die Benutzer-Interaktion

- *stop*
- *play/pause*

und folgende drei Zustände

- *stopped*
- *paused*
- *playing*

wobei der initiale Zustand *stopped* ist. Wird im Zustand *playing* die *play/pause*-Taste gedrückt, speichert der Spieler die aktuelle Wiedergabeposition. Neuerliches Drücken der Taste setzt die Wiedergabe an dieser Stelle fort. Das Drücken der *stop*-Taste löscht eine allenfalls gespeicherte Wiedergabeposition und beendet in jedem Fall die Wiedergabe.

- ✍ Erstelle ein UML-Zustandsdiagramm für den Musikspieler
- ✍ Implementiere den Musikspieler als endlichen Automaten

Implementierung von endlichen Automaten

- `switch-case`
  - \+ Zusätzlicher Code kann flexibel eingebaut werden
  - \+ Implementierung von Hand ist straight-forward
  - – Bei vielen Zuständen und Eingaben unübersichtlich
- Tabellengesteuert
  - \+ Sehr kompakte und effiziente Codierung
  - – Zusätzlicher Code schwierig einzubauen
  - – Low-level: Compiler kann nur sehr eingeschränkte Prüfungen durchführen
- *State*-Pattern
  - \+ High-level: Compiler kann umfangreiche Prüfungen durchführen
  - \+ Softwaretechnisch gut wartbar
  - – Viel Code
  - – Memory- und Performance-Overhead

Grundlegende Konzepte der parallelen Programmierung

- **Hardware**
  - Prozessor
  - Hauptspeicher
  - Instruktionszähler
  - Register
  - Cache
- **Betriebssystem**
  - Prozess
  - Adressraum
  - Scheduler
  - Stack
  - Thread (Faden)
  - Fiber (Faser)
- **Parallelität**
  - Single Instruction – Single Data (Sequentiell)
  - Single Instruction – Multiple Data (Vektorisierung)
  - Multiple Instruction – Single Data (Pipelines)
  - Multiple Instruction – Multiple Data (Multiproc.)
- **Nebenläufigkeit**
  - Kritischer Abschnitt
  - Lock (Mutex)
  - Atomare Operation
  - Transactional Memory
- **Techniken**
  - Fork/Join-Parallelisierung (OpenMP)
  - Tasks
  - Spooler

Wiederholung: *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*

Wiederholung: 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