Kommentare, Client-Server, Protokolle

Grundlagen für die erste Praktikumswoche

Michael Barth

19. Oktober 2006

Dokumentationsziel

Zweck und Funktionsweise sollten so dokumentiert werden, dass ein Programmierer eine Klasse und ihre Methoden verstehen und verwenden kann, ohne den Quellcode betrachten zu müssen.

Technik

- Klassen, Methoden und Attribute werden kommentiert
- JavaDoc-Kommentare `/** blabla */`
- mittels Kommando `javadoc` *package* | *files* werden dann HTML-Seiten erzeugt.
- Javadocs stehen vor Klassen-, Variablen-, und Methodendeklarationen
- Kommentar darf spezielle Tags enthalten

Javadoc - Tags

- Querverweise (Hyperlinks)
  - `@see` *classname*
  - `@see` *full_classname*#*method_name*
- Vor Methoden
  - `@param` *param_name description*
  - `@return` *description*
  - `@exception` *full_classname description*
- vor Klassen
  - `@author` *text*
  - `@version` *text*

Beispiel

```
public class Complex {
 private double im;
 private double re;
 ...

 /** Addiert Parameter
 * @param x komplexe Zahl
 */
 public void add(Complex x) {
 im = im + x.im();
 re = re + x.re();
 }}
```

JavaDoc-Beispiel II

```
/** Berechnet den Absolutbetrag
* @return Laenge des Vectors
* @see java.lang.Math
*/
public double absolutbetrag() {
return Math.sqrt( re*re + im*im );
}
```

JavaDoc-Compiler

```
/text % javadoc -d [Destination] [FileName]
```

- `-d` bestimmt das Ziel der Übersetzung (Beispiel): Name eines Verzeichnisses.
- `javadoc` erzeugt Html-Dokumentation für die Datei: `FileName`.
- Joker `*` ist erlaubt.

JavaDoc-Beispiel-Output

Es werden mehrere HTML-Files erzeugt:

- `AllNames.html` (Index für Klassen und Methoden)
- `Complex.html` (unsere Klassenbeschreibung)
- `tree.html` (Alle API-Klassen in baumartiger Übersicht)
- `packages.html` (Package-Übersicht)
- `index.html` (Die Übersicht mit den drei Frames...)

Architekturen

- Client-Server
- Peer-to-Peer

Praktikum: Individuell zugeschnittener Spezialfall **Client-Server**

Begriffe:

- Login-Server
- Server / Client
- Stub / Skeleton

Netzwerkfunktionalität der Java-API

TCP-IP-Netzwerkkommunikation im Paket: `java.net`

- ServerSocket - für den Loginserver
  - `public Socket accept() throws IOException`
- Socket - für Server und Client
  - `public Socket(String host,int port) throws UnknownHostException,IOException`
  - `public InputStream getInputStream() throws IOException`
  - `public OutputStream getOutputStream() throws IOException`
  - `public void close() throws IOException`

Ablauf einer Client-Server-Kommunikation

1. Loginserver starten (S)
2. LS lauscht über ServerSocket an einem Port (S)
3. Client starten (C)
4. Socket erzeugen und Verbindung zum Host an bestimmten Port aufbauen (C)
5. Loginserver akzeptiert die Verbindung und erzeugt einer serverseitigen Socket (S)
6. LS erzeugt Server und übergibt serverseitigen Socket (S)
7. ...wechselseitige Kommunikation über Streams...

Eingabe-Strom

- Auswahl durch:
  `public InputStream getInputStream() throws IOException`
- Zweckmäßig:
  Höherwertige Funktionalität durch Einbettung in
  - `InputStreamReader`
  - `BufferedReader` (siehe `java.io`)

(BufferedReader stellt eine `readln()`-Methode bereit.)

Ausgabe-Strom

- Auswahl durch:
  `public InputStream getInputStream() throws IOException`
- Zweckmäßig:
  Höherwertige Funktionalität durch Einbettung in
  - `PrintWriter`

(PrintWriter stellt eine `writeln()`-Methode bereit.)
...siehe Beispiel...

Textuelle Protokollspezifikation

- Menge an Kommandos, dabei jeweils:
  - Beschreibung der Vorbedingung (Ausgangszustand)
  - Syntax (Kommando und Parameter)
  - Ablaufbeschreibung (optional)
  - Beschreibung der Nachbedingung (Folgezustand)
- Menge an Zuständen

Beispiel: POP3-Protokoll

Graphische Spezifikation

Zustandsautomat bzw. Transitionssystem

- Menge an Zuständen
- Menge an Transitionen
  (gerichtete Kanten, die Zustände verbinden)

Kantenanschriften spezifizieren z.B.:

- gesendete Nachrichten
- Aktionen
- empfangene Nachrichten
- Bedingungen (Wächter)

Standards: zB. UML-Zustandsdiagramme

UML Zustandsdiagramme

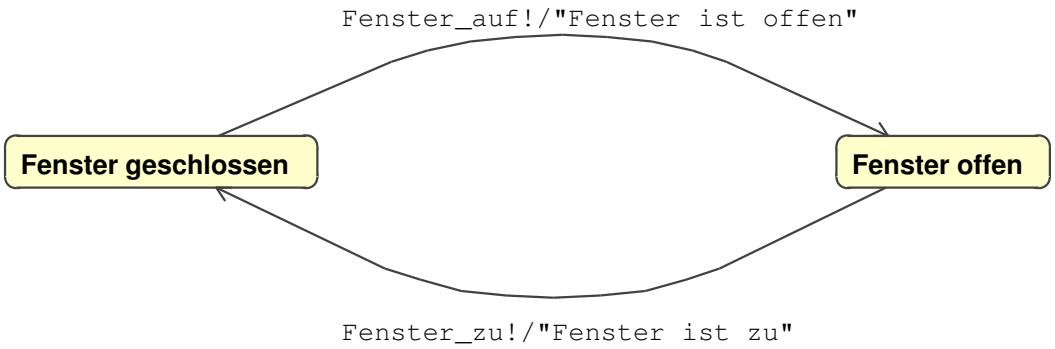
(Teil-)Syntax für Transitionen in UML Zustandsdiagrammen:

- `[Ereignis][/Aktion]`
- Ereignis: zB. Empfang einer Nachricht, Methode wird (von aussen) aufgerufen
- Aktion: zB. Senden einer Nachricht, Methodenaufruf (selbst)

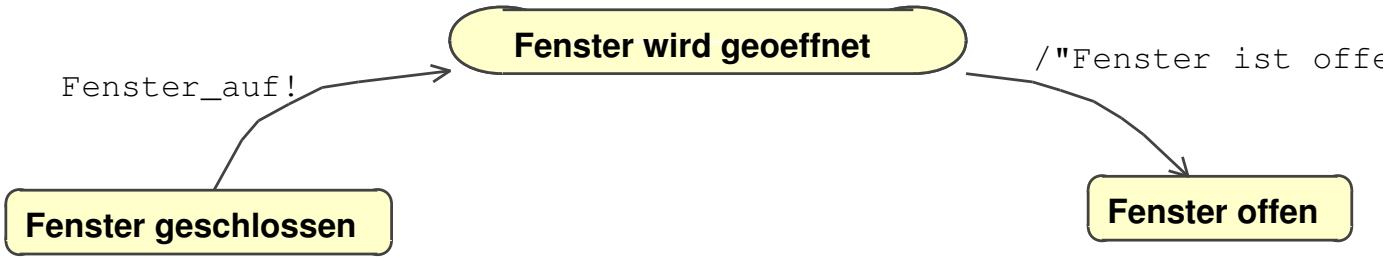
Ereignis und Aktion müssen nicht immer zusammen an einer Transition notiert werden. (Steht ein Ereignis allein, kann der Schrägstrich weggelassen werden)

Beispiele

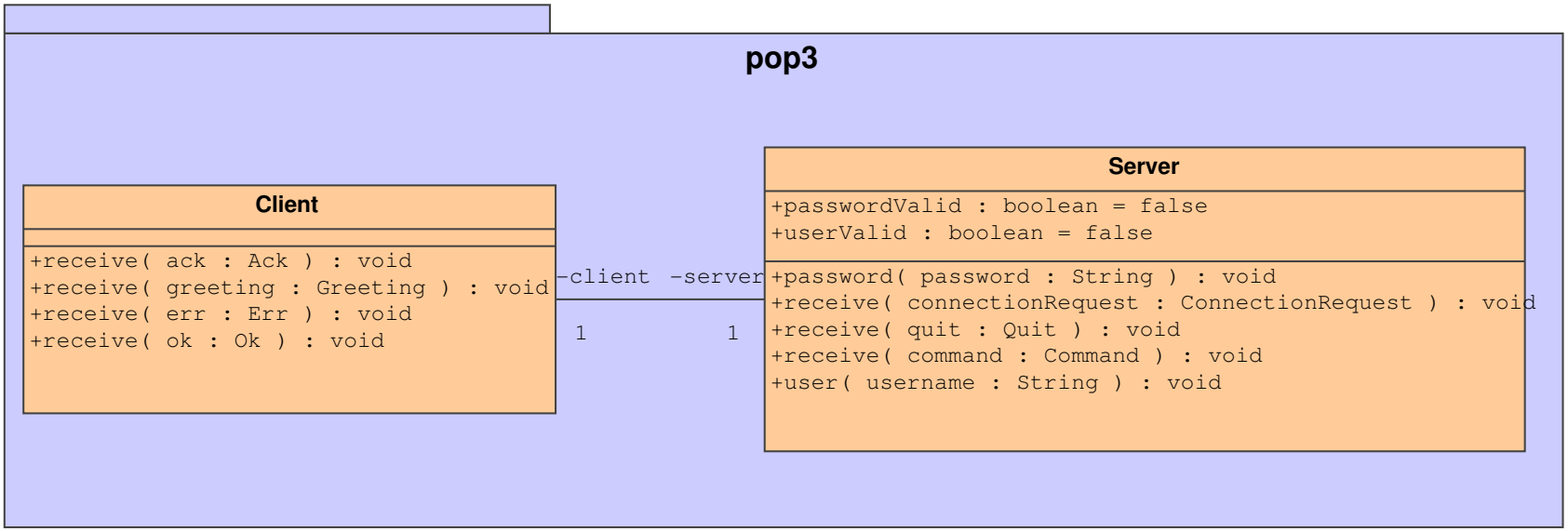
Öffnen und Schließen einer Tür:



Öffnen in zwei Schritten:

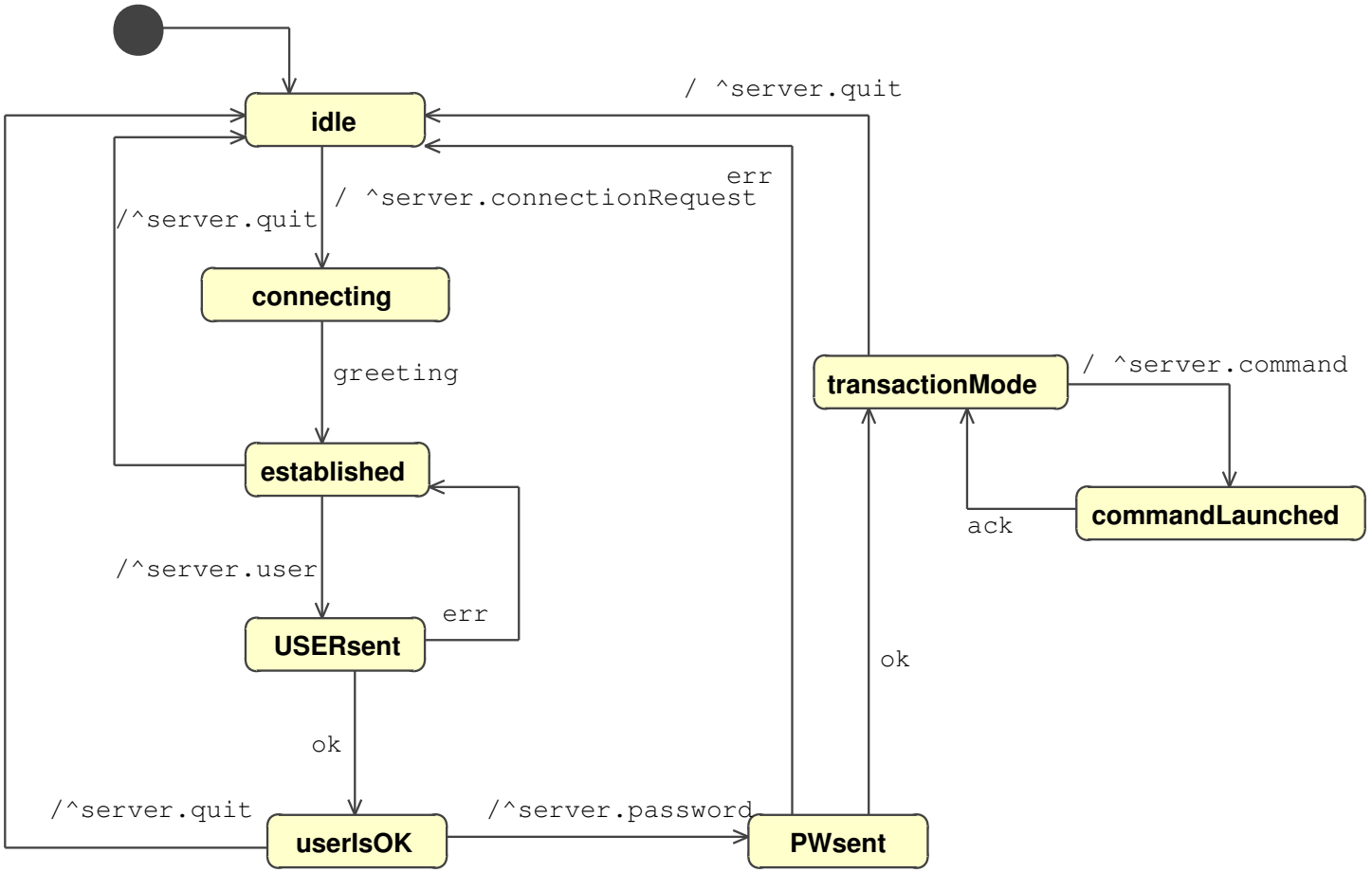


Beispiel: POP3 in UML



POP3 Client-Server in UML

Beispiel: POP3 in UML



POP3-Protokollspezifikation in UML

...zum Schluß!

Hänsel und Gretel...

...vertreiben sich die Zeit!