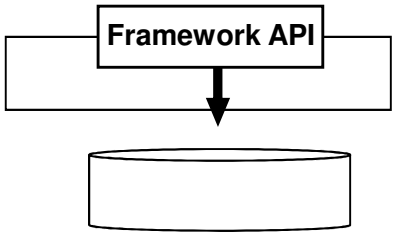
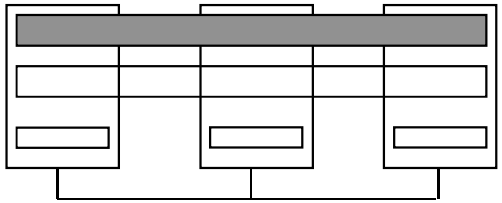
Software Engineering III – Wo sind wir?

Softwarearchitekturen

Softwarearchitekturen:
Begriffe & Fallbeispiel
Persistenz-Framework

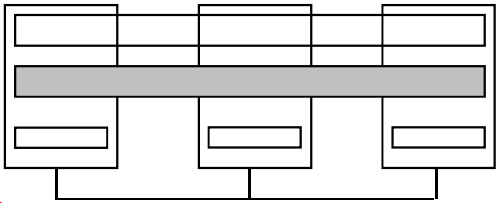


Verteilte
Softwarearchitekturen



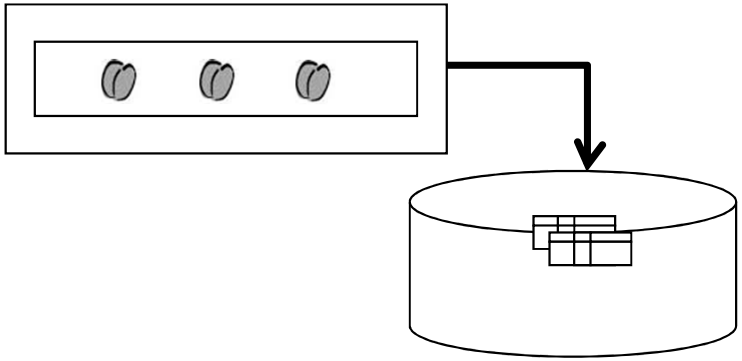
Middleware/Verteilung

Sockets, RMI, MoM/JMS
Web Services



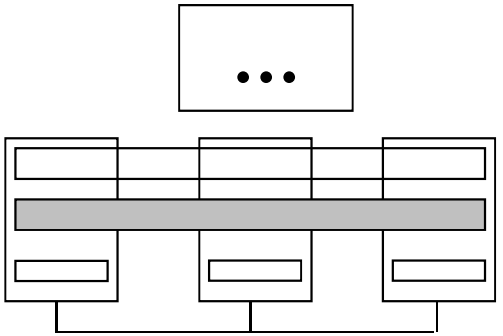
Enterprise Java

Java EE (EJB, …)



Verteilung – weitere Konzepte

Weitere Middleware



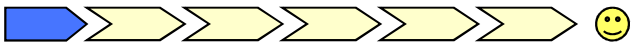
Gliederung

Transport-Protokolle

- **physikalischer Austausch von Daten**
- Standard: TCP/IP-Protokoll
  - verschiedene Schichten mit speziellen Aufgaben
  - Identifikation eines Rechners: durch Hostname oder IP-Adresse

Ports

- Port-Nummer **identifiziert Prozess** auf einem Rechner
- für **bidirektionale Kommunikation** zwischen Client und Serverdienst
  - **Server-Portnummer**: ist **festgelegt** und dem Client bekannt (telnet: 23, http: 80, https:443, ...)
  - **Client-Portnummer**: vom Betriebssystem **dynamisch zugewiesen** (wird für Datenaustausch: Server → Client benötigt)



Sockets

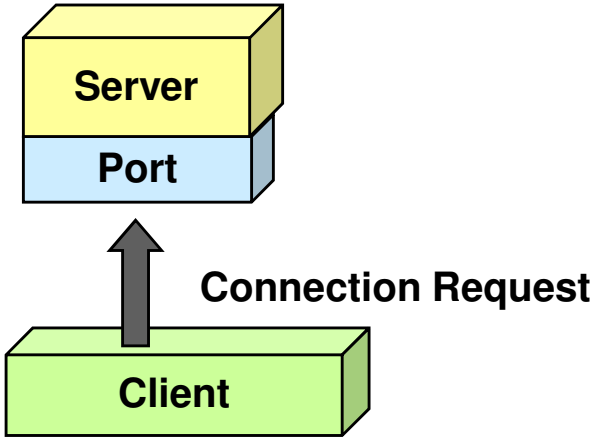
- **Basis-Mechanismus** für alle komplexeren Verfahren (RMI,...)
  - **Austausch von Byteströmen** auf Programmierebene
  - Socket ist Verbindung zwischen zwei Kommunikations-Endpunkten (= IP-Adresse + Port)
  - es gibt zwei verschiedene Rollen: Client und Server

- **Nachteile von Sockets**
  - Parsen des Bytestroms erforderlich
  - keine Objekte, d.h. keine Typsicherheit
  - Sonderfall: XML-Daten als Alternative zu Objekten

Client:

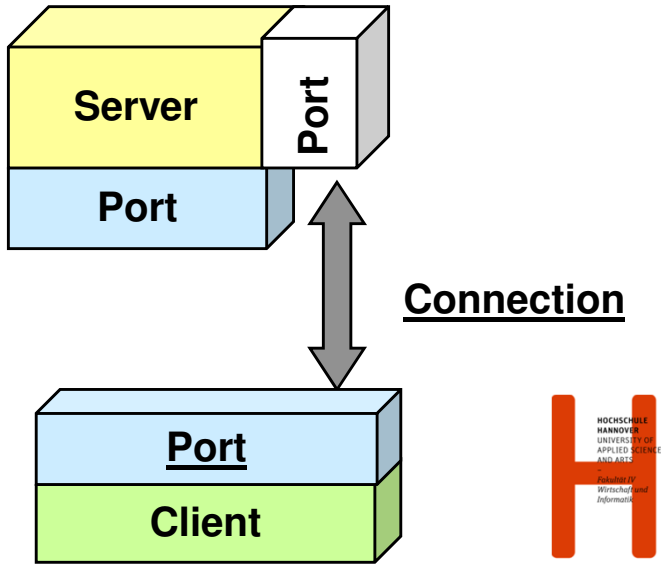
- kennt Hostname und Port für Connect mit Server

1. stellt Connect her
2. erhält Socket-Objekt
3. kommuniziert mit Methoden des Socket-Objekts



Server:

1. Listening: wartet auf Client-Request über Connect-Port
2. akzeptiert Client-Request
3. erzeugt für jeden Client-Request neuen Socket
4. ursprünglicher Socket steht für weitere Requests zur Verfügung



Bemerkung: Anwendung des Server/Handler-Musters

Client-Implementierung :

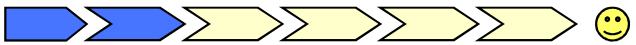
1. öffnet Socket zu einem Server
2. öffnet Input- und Output-Stream für Server-Socket
3. liest und schreibt über die Stream-Objekt (schließt anschließend Stream-Objekte und Sockets)

```
String host = "nbak";   int port = 1234;
Socket toServer = new Socket(host, port);
BufferedReader in = new BufferedReader(new InputStreamReader(toServer.getInputStream()));
String message = in.readLine();
```

Server-Implementierung:

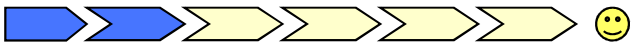
1. öffnet Server-Socket für bestimmten Port
2. akzeptiert Client-Request und erzeugt Client-Socket für Kommunikation mit Client
3. öffnet Input- und Output-Stream für Client-Socket
4. liest und schreibt mittels Stream-Objekt (schließt anschließend Stream-Objekte und Sockets)

```
int port = 1234;
ServerSocket server = new ServerSocket(port);
Socket client = server.accept();
PrintWriter socketOut = new PrintWriter(client.getOutputStream(), true);
socketOut.println("Hello world from the server");
```

4.1.1 Interprozess-Kommunikation und Sockets: **TimeServer-Beispiel**

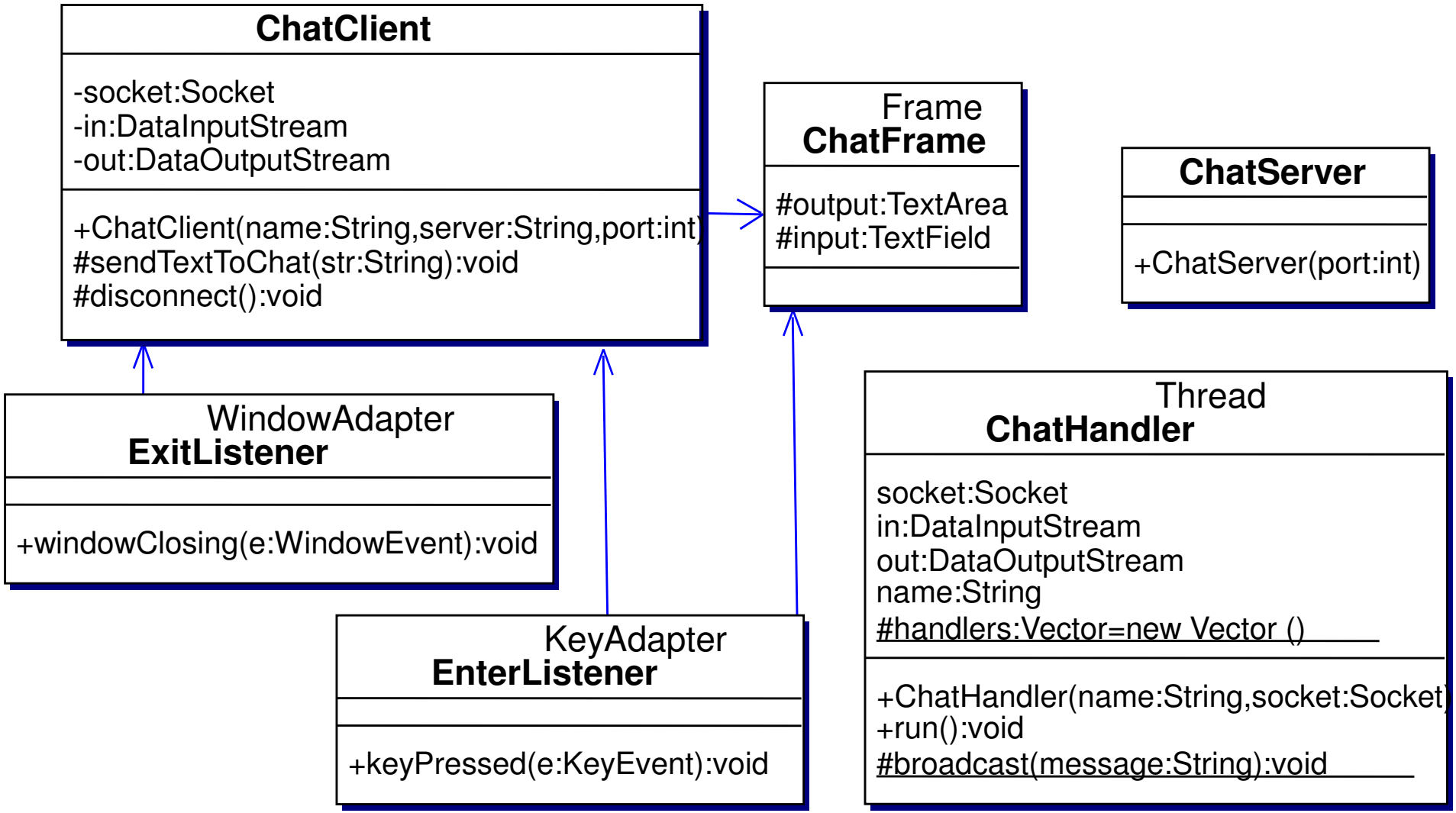
```
import java.net.ServerSocket;

public class TimeServer {

    public static void main(String args[]) throws Exception {
        int port = 2342;

        // erzeuge neuen Server-Socket f∏r sepziellen Port
        ServerSocket server = new ServerSocket(port);

        while (true) {

            System.out.println("server > Waiting for client...");
            // Server akzeptiert Anfragen von Client und erzeugt dazu Client-Socket
            // Vorsicht: die Methode blockiert bis der Client eine Connection aufmacht
            Socket client = server.accept();
            System.out.println("server> Client from " + client.getInetAddress() + " connected.");

            // Stream zum Schreiben an den Client
            PrintWriter out = new PrintWriter(client.getOutputStream(), true);
            Date date = new Date();
            out.println(date);
        }
    }
}
```

4.1.1 Interprozess-Kommunikation und Sockets

```
import java.net.Socket;

public class TimeClient {

    public static void main(String args[]) throws IOException {
        String host = "localhost";
        int port = 2342;

        // Socket zur Verbindung zum Server
        Socket server = new Socket(host, port);
        System.out.println("client> Connected to " + server.getInetAddress());

        // Stream zum Lesen von Server
        BufferedReader in =
            new BufferedReader(new InputStreamReader(server.getInputStream()));
        String date = in.readLine();
        System.out.println("client > Server said: " + date);
    }
}
```

4.1.2 Socket-Beispiel: Chat-System – Klassendiagramm



4.1.2 Beispiel: Chat-System

- **Chat-Server:**
  - erzeugt Server Socket
  - wartet auf Client-Anmeldung
    - erzeugt ClientSocket
    - delegiert Kommunikation mit Client an Handler-Objekt
      (Server/Handler-Muster: eigener Thread für jeden neuen Client)

- **Chat-Handler:**
  - verwaltet statischen Vector mit allen Client-Handlern (für Broadcast an alle Clients)
  - erzeugt BufferReader und Printwriter für ClientSocket
    - liest Nachricht und führt Broadcast durch

- **Chat-Client(s):**
  - baut Verbindung zum Server auf
    - schickt und empfängt Nachrichten an Server
  - besitzt GUI: Eingabe neuer Nachrichten u. empfangene Nachrichten anzeigen

Demo

Chat mit Sockets

Code → Anhang

Zusammenfassung

- Sockets
  - Basis-Mechanismus für verteilte Aufrufe
  - IP-Adresse & Port
  - Nur einfache Byte-Ströme
  - Kein Programmierkomfort

Literatur

Jürgen Dunkel, Andreas Eberhart, Stefan Fischer, Carsten Kleiner, Arne Koschel: Systemarchitekturen für Verteilte Anwendungen - Client-Server, Multi-Tier, SOA, EDA, Grid, P2P, …, Hanser, 2008.

M. Boger: Java in verteilten Systemen, dpunkt.verlag, 1999

G. Bengel: Grundkurs Verteilte Systeme, vieweg, 3. Auflage, 2004

D. Abts: Aufbaukurs Java: Client/Server-Programmierung mit JDBC, Sockets, XML-RPC und RMI, vieweg, 2003

R. Oechsle: Parallele und verteilte Anwendungen in Java, Hanser, 2. Aufl., 2007.

C.S. Horstmann, G. Corell: Core Java 2 – Volume 2 Advanced Features,7. Edt, Prentice Hall 2005

D. Lea: Concurrent Programming in Java, Design Principles and Patterns,Addison Wesley, 1997

Oracle/Sun-Online-Dokumentation