# 4300 SOCKET PROGRAMMIERUNG

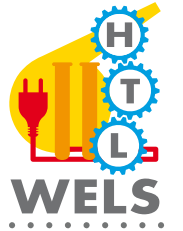
*SEW 4*

*DI Thomas Helml*

HTL WELS

IT INFORMATIONSTECHNOLOGIE


# INHALT



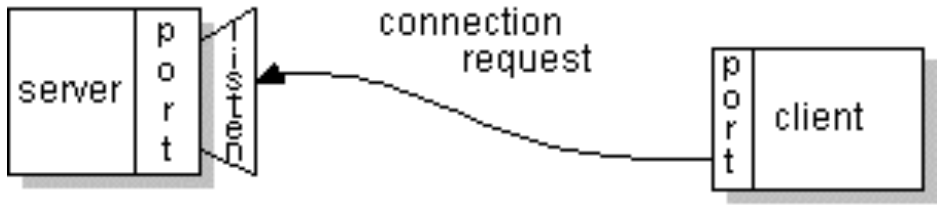

- Netzwerkkommunikation über Sockets
- Multithreaded Server
- HTTP

# SOCKETS

- Socket
  - repräsentiert Endpunkte einer Verbindung zw. 2 Hosts
  - vom OS zur Verfügung gestellt
  - bidirektional

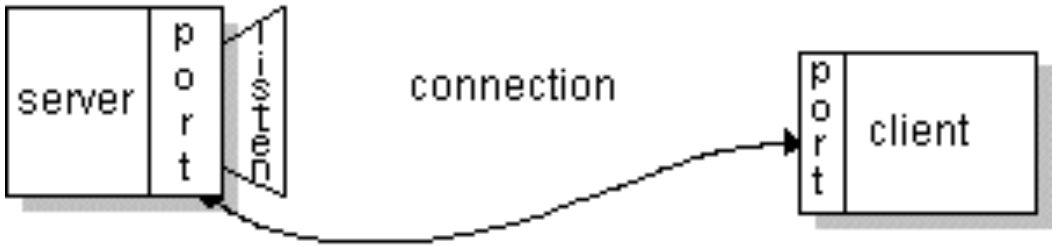
# SERVER

- Server:
  - üblicherweise eigener Rechner
  - Socket, der auf bestimmten Port gebunden ist
  - wartet auf eingehende Verbindungen

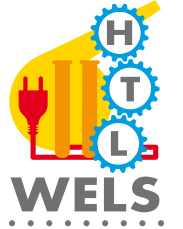


# SERVER

- Server:
  - alles ok -> Verbindung wird akzeptiert
  - Server bekommt neuen Socket für die akzeptierte Verbindung
  - „Ursprünglicher“ Socket lauscht weiter auf eingehende Verbindungen

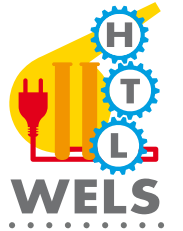



# CLIENT




- Client:
  - Socket erstellen - Verbindungsaufbau zu Server
  - wird Verbindung akzeptiert, so kann der Client über diesen Socket mit Server kommunizieren
  - Client+Server können jeweils am Socket lesen/schreiben


# CLIENT-/SERVER SOCKETS




- Unterscheide Client- und Server-Seite!
  - Client:
    - Klasse java.net.Socket
  - Server:
    - Klasse java.net.ServerSocket

# ABLAUF CLIENT

1. Socket erzeugen

```
Socket(String host, int port) throws IOException
```

2. Input stream & Output stream am Socket öffnen

```
InputStream getInputStream() throws IOException
```

```
OutputStream getOutputStream() throws IOException
```

3. Lesen/Schreiben in den Stream – abhängig vom Protokoll des Servers

4. Streams schließen

5. Socket schließen

```
void close() throws IOException
```

# ABLAUF SERVER

1. ServerSocket (Port!) öffnen (event. Timeout)

```
ServerSocket(int port) throws IOException
```

2. auf Verbindung warten (blockierend!) -> Client Socket

```
Socket accept() throws IOException
```

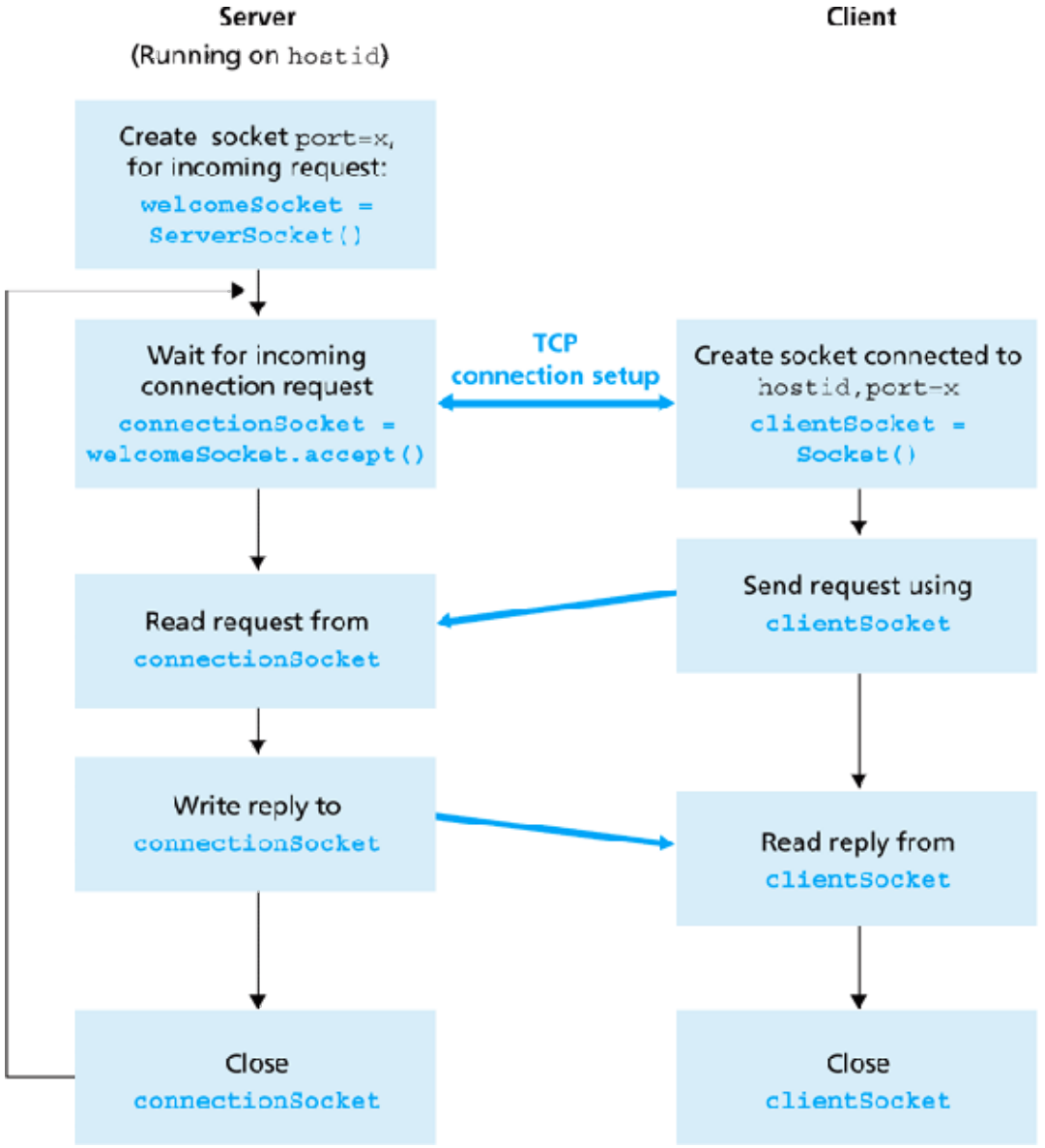
3. Kommunizieren über Client Socket (Thread?)
4. Input stream & Output stream am Socket öffnen

```
InputStream getInputStream() throws IOException
OutputStream getOutputStream() throws IOException
```

5. Lesen/Schreiben in den Stream
6. Streams schließen
7. Socket schließen

```
void close() throws IOException
```

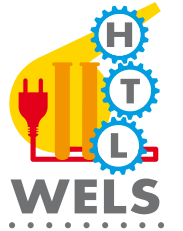
# SERVER


Server
(Running on hostid)
Client
Create socket port=x, for incoming request:
welcomeSocket = ServerSocket()
Wait for incoming connection request
connectionSocket = welcomeSocket.accept()
TCP connection setup
Create socket connected to hostid,port=x
clientSocket = Socket()
Send request using clientSocket
Read request from connectionSocket
Write reply to connectionSocket
Read reply from clientSocket
Close connectionSocket
Close clientSocket

# STREAMS – WH

| Ausgabe-Streams | | Eingabe-Streams | |
|---|---|---|---|
| **Byte-Streams** | **Character-Streams** | **Byte-Streams** | **Character-Streams** |
| OutputStream | Writer | InputStream | Reader |
| FileOutputStream | FileWriter | FileInputStream | FileReader |
| BufferedOutputStream | BufferedWriter | BufferedInputStream | BufferedReader |
| ByteArrayOutputStream | CharArrayWriter | ByteArrayInputStrem | CharArrayReader |
| FilterOutputStream | FilterWriter | FilterInputStream | FilterReader |
| PipedOutputStream | PipedWriter | PipedInputStream | PipedReader |
| PrintStream | PrintWriter | | |
| | | PushbackInputStream | PushbackReader |


# STREAMS - WH




- `DataInputStream + DataOutputStream`
  - für Senden/Empfangen von Basisdatentypen!
  - `writeInt, writeDouble, …`
  - `readInt, readDouble, …`
- `ObjectInputStream + ObjectOutputStream`
  - für Senden/Empfangen von Objekten
  - `readObject`
  - `writeObject`