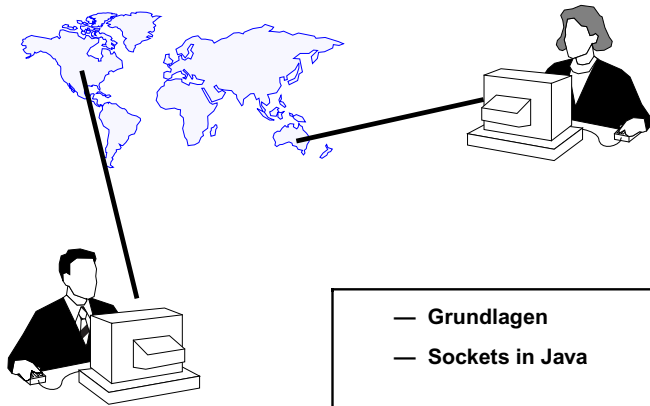


## Java und Netzwerkkommunikation

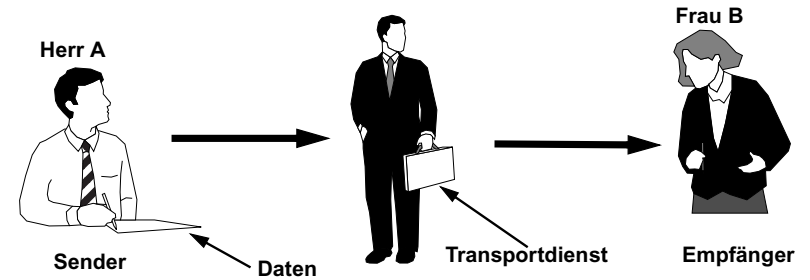
**Ziel: Kommunikation über Rechengrenzen hinweg**



- Grundlagen
- Sockets in Java

## Grundlagen

**Senden und Empfangen von Daten !**

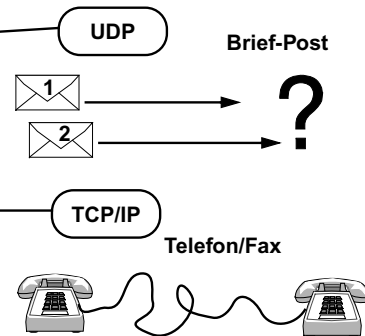


- Identifizieren des Empfängers/ Senders
  - global eindeutige Adressen
  - hierarchische Adressierung
- Leistung des Transportdienstes
  - Kosten
  - Sicherheit
  - Reihenfolge der Nachrichten

## Grundlagen

**Internet-Transportdienste**

- **verbindungslos**
  - Ankunft nicht garantiert
  - Reihenfolge nicht garantiert
  - billig
- **verbindungsorientiert**
  - Ankunftsgarantie
  - FIFO Reihenfolge
  - teuer



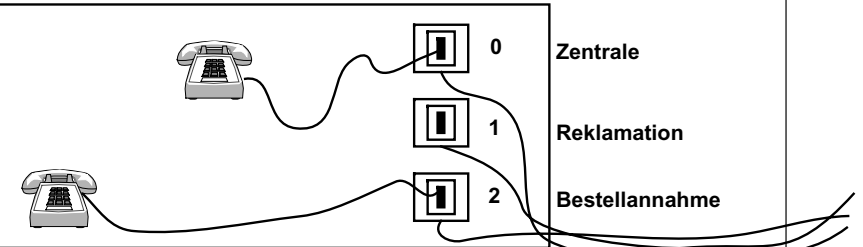
**Protokoll = Vereinbarung zur geregelten Kommunikationsabwicklung**

- abhängig vom benutzten Transportdienst !

## Grundlagen

**Fa. Heisse Luft**

**06151 / 4578-**



**Socket = Kommunikationsendpunkt**

- Datenstruktur
- Verwaltet Informationen
  - Daten
  - Verbindung

**Port = Adressen möglicher Komm-Endpunkte**

- Abstraktion vom tatsächlichen Kommunikationsendpunkt
- Adressen von Diensten

## Grundlagen

### Daten verbindungslos versenden:

- Socket erzeugen
- Socket an einen lokalen Port binden
- Daten über Socket senden/empfangen
- Socket schließen

Internetadresse + Zielport

### Daten verbindungsorientiert versenden:

- Socket erzeugen
- Socket an einen lokalen Port binden
- Verbindung mit Zieladresse herstellen
- Daten über Socket lesen/schreiben
- Socket schließen

www.inf.ethz.ch  
129.132.178.197

## Grundlagen

Der Empfänger muß empfangsbereit sein!

Herr A

Port 80



Am Socket "lauschen"

Frau B



## Grundlagen

Unbemerkt Binden an lokalen Port

Port 80



Herr A

Frau B



3. ankommende Verbindung wird signalisiert  
4. Verbindung annehmen

1. Socket erzeugen  
2. Verbindung zu Port 80 herstellen



Verbindung kann von beiden Seiten geschlossen werden



Zu einem kommunikationsbereiten Socket gehört immer ein Port

## Networking in Java

### Java Package java.net:

- Internetadresse
- Datagramm Pakete
- Datagramm Socket
- TCP/IP- Empfänger Socket
- TCP/IP- Sender Socket
- URL
- URL-Verbindungen

## Internetadressen

### Class InetAddress

```
public final class java.net.InetAddress
extends java.lang.Object
{
    // Methods
    public static InetAddress getByName(String host);
    public static InetAddress[] getAllByName(String host);
    public static InetAddress getLocalHost();
    public String getHostName();
    public String toString();
    public boolean equals(Object obj);
    ....
}
```

Kein Subclassing möglich!

Klassenmethoden

"www.inf.ethz.ch"  
"129.132.178.197"

```
InetAddress myaddress;
String myname;
.....
myaddress = java.net.InetAddress.getLocalHost();
myname = myaddress.getHostName();
.....
```

## Verbindungsorientierte Sockets

### Class Socket

```
public class java.net.Socket
extends java.lang.Object
{
    // Constructors
    public Socket(InetAddress address, int port);
    public Socket(String host, int port);

    // Methods
    public void close();
    public InetAddress getInetAddress();
    public InputStream getInputStream();
    public OutputStream getOutputStream();
    public int getLocalPort();
    public int getPort();
    ....
}
```

Lesen und Schreiben  
wie von/auf Datei

```
socket = new Socket(toaddr, toport);
in = socket.getInputStream();
out = socket.getOutputStream();
.....
in.read(.....);
out.write(.....);
.....
socket.close();
```

Lesen

Schreiben

## Verbindungsorientierte Sockets

### Class ServerSocket

```
public class java.net.ServerSocket
extends java.lang.Object
{
    // Constructors
    public ServerSocket(int port);
    public ServerSocket(int port, int count);

    // Methods
    public Socket accept();
    public void close();
    public InetAddress getInetAddress();
    public int getLocalPort();
    ....
}
```

Anzahl gleichzeitiger  
Verbindungswünsche

wartet auf ankommende  
Verbindung (blockiert)

neuer Socket !  
kein blockieren  
des Serversockets

## Client - Server

```
// öffnen des Serversockets, Port 2000
s_socket = new ServerSocket(2000);

// endlosschleife
while (true) {
    // warten auf nächste Verbindung
    c_socket = s_socket.accept();
    doWork(c_socket);
}

void doWork(Socket s) {
    in = s.getInputStream();
    out = s.getOutputStream();
    ....
    in.read(....);
    ....
    out.write(....);
    ....
    s.close();
}
```

```
// Verbindung zum Server herstellen
socket = new Socket("Server", 2000);

// Streams zum lesen/schreiben
in = socket.getInputStream();
out = socket.getOutputStream();
.....

// Anfrage an Server senden
out.write(.....);

// Antwort von Server lesen
in.read(.....);
.....

// Socket schließen
socket.close();
```