13.1.2 Bemerkungen zur Realisierung in Java

Der Anbieter definiert mit dem Aufruf eines Konstruktors

```
ServerSocket sS = new ServerSocket(port);
```

zuerst einen Anbieter-Socke für einen bestimmten Port vom Typ `int` und blockiert dann mit der Anweisung

```
Socket s = sS.accept()
```

zur Rückgabe eines Sockels bis zum Empfang der Botschaft eines Kunden auf dem Port (Diese Anweisungsfolgen müssen in `try`-/`catch`-Anweisungen eingeschlossen werden.)

Wenn eine Botschaft eingetroffen ist ist, holt er sich z. B. mit

```
InputStreamReader i = new InputStreamReader(s.getInputStream());
BufferedReader r = new BufferedReader(i);
```

einen gepufferten `Reader`, auf dem er mit der Anweisung

```
String r = b.ReadLine();
```

die von einem Kunden gesendete Zeichenkette empfänt und der Variablen `r` zuweisen kann.

Er sendet dann an diesen Kunden eine Antwort mit einem `PrintWriter`, den er mit dem Aufruf eines Konstruktors

```
PrintWriter p = new PrintWriter(s.getOutputStream, true);
```

definiert hat und darauf mit

```
p.println(r);
```

eine von ihm konstruierte Zeichenkette `r` an den Kunden zurück.

Ein Kunde verfährt im Prinzip ganz ähnlich; er muss natürlich den Server angeben, dessen Dienst er in Anspruch nehmen will. Dazu parametrisiert er den Aufruf des `Socket`-Konstruktors mit dessen Namen (oder IP-Adresse)

```
String host = "..."; // ... = Hostname oder IP-Adresse
int port = 50000;
Socket s = new Socket(host, port);
```

Die `Reader` und `Writer` besorgt er sich mit den gleichen Anweisungen wie oben beim Anbieter angegeben.

Ein (Anbieter-)Sockel `s` wird mit der Anweisung s.close(); geschlossen.

Details zu den benutzten Aufrufen findet man in den Klassen `java.net.Socket`, `java.net.ServerSocket`, `java.io.BufferedReader`, `java.io.InputStreamReader` und `java.io.PrintWriter`.