# Chatserver-Aufgabe

## Einleitung

Wir wollen einen Chatserver erstellen, auf dem sich Benutzer anmelden und gegenseitig Nachrichten schicken können. Dies soll mit Hilfe des Telnet-Protokolls geschehen.

Der Telnet-Client gehört zu Windows, starten Sie ihn, indem sie im Startmenü auf „Ausführen" gehen, dort „cmd" eingeben, bestätigen, und in dem dann auftauchenden Prompt

```
telnet <zielrechner>  <portnummer>
```

eingeben, also z.B.

```
telnet 127.0.0.1 4999
```

Telnet (Telecommunication Network) ist der Name eines im Internet weit verbreiteten Netzwerkprotokolls.

Das IETF-Dokument STD 8 (RFC 854 und RFC 855), in dem es beschrieben wird, beginnt folgendermaßen: „Der Sinn des TELNET-Protokolls besteht darin, eine ziemlich allgemeine, bidirektionale, 8-bit-pro-Byte-orientierte Kommunikationsmöglichkeit zu bieten." Es wird üblicherweise dazu verwendet, Benutzern den Zugang zu Internetrechnern über die Kommandozeile (CLI) zu bieten. Aufgrund der fehlenden Verschlüsselung wird es kaum noch eingesetzt.

Der Vorteil dieses extrem einfachen Protokolls für uns ist, dass wir einfach mit den Standardmethoden Zeichen auf den geöffneten Socket schreiben können, die dann beim Anwender im Telnet-Client „auftauchen".

## Aufgabe

Erstellen Sie eine Klasse „ChatServer". Beim Erstellen einer Instanz dieser Klasse soll eine Portnummer angegeben werden können, auf der der Chatserver dann beginnt zu lauschen. (1 Punkt)

Wenn sich ein neuer Benutzer mit dem Server verbindet, soll er begrüßt werden und nach seinem Nickname gefragt werden. Sorgen Sie dafür, dass keine Nicknames doppelt vergeben werden, auch nicht solche, die sich nur durch angehängte oder vorangestellte Leerzeichen von existierenden Nicknames unterscheiden.
(2 Punkte)

Nachdem der User seinen Nickname gewählt hat, soll er Nachrichten senden und empfangen können.
Eine Nachricht wird immer an alle User versandt und hat das folgende Format:

```
[14:05:13] [username-des-senders] Eine Seefahrt, die ist lustig
```

(4 Punkte)

Neu hinzukommende oder verlassende User sollen vom Server allen Usern mitgeteilt werden. (1 Punkt)

**Tipps**

Lassen Sie den Server alle Aktionen und Gespräche auf der Standardausgabe protokollieren.

Benutzen Sie eine Endlosschleife, um den Server dauerhaft laufen zu lassen.

Speichern Sie alle offenen Sockets in einem Array.

Während Sie auf Eingabe der Benutzer warten, benutzen Sie die folgende Funktion und lassen Sie sie das Array der offenen Sockets überwachen:

*select( readArray (, writeArray ( errorArray ( timeout ) ) ) ) -> anArray or nil*

*Performs a low-level select call, which waits for data to become available from input/output devices. The first three parameters are arrays of IO objects or nil. The last is a timeout in seconds, which should be an Integer or a Float. The call waits for data to become available for any of the IO objects in readArray, for buffers to have cleared sufficiently to enable writing to any of the devices in writeArray, or for an error to occur on the devices in errorArray. If one or more of these conditions are met, the call returns a three-element array containing arrays of the IO objects that were ready. Otherwise, if there is no change in status for timeout seconds, the call returns nil. If all parameters are nil, the current thread sleeps forever.*
D.h. im ersten Element des von dieser Funktion zurückgelieferten Arrays finden Sie die Sockets, auf die Daten von Clients geschrieben und folglich vom Server gelesen worden sind.

Verlassende Benutzer senden auf dem Socket ein End-of-file (EOF).