Übung 08: Threads, IO und Networking

Abgabetermin: 19. 05. 2016, 10:00

Name: _____ **Matrikelnummer**: _____

Informatik: ☐ G1 (Grimmer) ☐ G2 (Prähofer) ☐ G3 (Prähofer)

WIN: ☐ G1 (Khalil) ☐ G2 (Khalil) ☐ G3 (Hummel)

Aufgabe	Punkte	abzugeben schriftlich	abzugeben elektronisch	korr.	Punkte
Übung 6	24	Java-Programm, Ausgabe eines Testlaufs UML-Diagramme	Java-Programm	☐	

ChatServer und ChatClient (20 Points)

Erstellen Sie unter Verwendung von Threads, Sockets und Streaming eine einfache Chat-Anwendung, die einen Chat zwischen zwei Teilnehmern über das Internet ermöglicht. Folgend ist ein möglicher Ablauf skizziert.

```
>java chat.ChatServer
Port number: 2000
Waiting for client request
++ Connected to client
================================================
Press enter for input! Enter 'x' to terminate!
================================================
 Received: Hallo server
------------------------------
Your message: Hallo client
------------------------------
------------------------------
Your message: Was gibt's?
------------------------------
 Received: Ich kenn mich bei SW2 nicht aus.
 Received: Ich weiss nicht wie das Chatten geht.
------------------------------
Your message: Das ist ja einfach.
------------------------------
------------------------------
Your message: Du musst einen Socket oeffnen.
------------------------------
------------------------------
Your message: Und dann Messages sc
------------------------------
 Received: Ah! Das ist ja einfach.
 Received: Good Bye
------------------------------
Your message: Haettest auch selber wissen koennen.
------------------------------
x
Wait for next client? (y | n):
```

Enter
Enter
Enter
Enter
Enter
'x' Enter

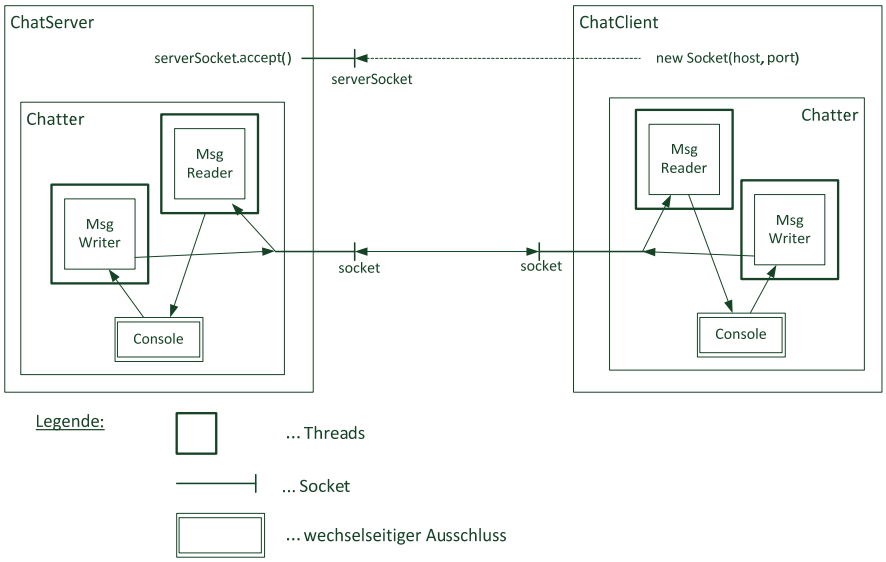
```
>java chat.ChatClient
Server IP: localhost
Port number: 2000
++ Connected to server
================================================
Press enter for input! Enter 'x' to terminate!
================================================
------------------------------
Your message: Hallo server
------------------------------
 Received: Hallo client
 Received: Was gibt's?
------------------------------
Your message: Ich kenn mich bei SW2 nicht aus.
------------------------------
------------------------------
Your message: Ich weiss nicht wie das Chatten geht.
------------------------------
 Received: Das ist ja einfach.
 Received: Du musst einen Socket oeffnen.
 Received: Und dann Messages schic
------------------------------
Your message: Ah! Das ist ja einfach.
------------------------------
x
 Received: Haettest auch selber wissen koennen.
 Received: Good Bye
```

Enter
Enter
Enter
Enter
'x' Enter

- Ein Teilnehmer startet einen ChatServer unter einem einzulesenden Port (Port sollte > 2000 sein).
- Der andere Teilnehmer startet einen ChatClient, liest die Server-Adrresse (zum Testen "localhost") und den Port ein und verbindet sich zum Server.
- Die Teilnehmer drücken die Enter-Taste, um eine Message zu schicken. Sie können dann eine Zeile eingeben, die zum anderen Teilnehmer geschickt wird.
- Die Teilnehmer empfangen jeweils die Messages des anderen und geben sie auf der Konsole aus.
- Mit Eingabe von "x" kann ein Teilnehmer den Chat beenden. Beenden des Chats soll folgend ablaufen:
  - Es wird die Nachricht „Good bye“ gesendet. Der Teilnehmer kann dann selbst keine weiteren Nachrichten mehr senden.
  - Der Chatpartner kann noch weitere Messages schicken, die vom Partner empfangen und ausgegeben werden.
  - Der Chatpartner muss auch mit "x" beenden. Beide Chat-Abläufe terminieren
  - Der ChatServer fragt "`Wait for next client? (y | n):`", d.h., er kann entscheiden, ob er auf weitere Client-Verbindungen warten will.

Architektur

Die folgende Abbildung zeigt die Architektur des Systems. Der ChatServer wartet auf Verbindungen und der ChatClient baut eine Verbindung auf. Die Kommunikation wird dann über Chatter-Objekte auf beiden Seiten abgewickelt. Das Lesen der Nachrichten vom Socket und Schreiben auf die Konsole passiert in einem Thread, die Eingabe der Nachricht und das Senden über den Socket in einem anderen Thread. Der Zugriff auf die Konsole passiert unter wechselseitigem Ausschluss.



Hinweise:

Bei der Realisierung sind folgende Anforderungen zu erfüllen:

- Empfangen und Senden der Nachrichten muss in zwei unterschiedlichen Threads erfolgen.
- Es ist wichtig, dass die Ausgabe auf die Konsole und die Eingabe von der Konsole unter wechselseitigem Ausschluss passieren. Sonst würden ja in die Tastatureingabe empfangene Messages geschrieben werden. Daher müssen Sie die Ein- und Ausgabe auf die Konsole in einem eigenen Objekt kapseln und synchronieren. Achtung: Die Eingabeaufforderung durch Drücken der Entertaste und die Eingabe von "x" dürfen Sie aber nicht synchronisieren, weil sonst die Konsole immer gelockt bleibt.
- Achten Sie darauf, dass der Chat am Client und am Server richtig beendet wird. Das Beenden eines Chats nach Eingabe von `"x"` soll folgend implementiert werden:
  - zuerst wird eine Abschlussmeldung (z.B. `"Good Bye"`) geschickt
  - dann wird die Ausgabe mit `socket.shutdownOutput();` geschlossen
  - der Thread zum Lesen der Nachrichten läuft weiter bis die Gegenseite auch den Output schließt und keine Nachrichten mehr empfangen werden
  - im anderen Thread mit `join` auf das Ende des Threads zum Lesen der Nachrichten warten
  - dann erst den Socket schließen und terminieren

Diagramme (4 P)

Zeichnen Sie für Ihre Anwendung folgende 2 Diagramme:

- Klassendiagramm
  - Zeichnen Sie ein UML Klassendiagramm
- Sequenzdiagramm
  - Stellen Sie einen kurzen Ablauf dar
  - A sendet eine Nachricht
  - B empfängt und antwortet
  - A beendet
  - B beendet

Es reicht eine Handskizze.