Protokoll "Java Simple-Chat"

Moritz Netrwal

Inhalt

1 Server

Der Server wird als Objekt der Klasse Server erstellt.
Um sicherzustellen dass der Socket ohne Exceptions gestartet wurde, wird dies in einer eigenen Methode durchgeführt, welche einen Boolean zurückliefert.
In einem neuen Thread wird eine Methode aufgerufen, welche in einem folgenden Paragraphen genauer erläutert wird.

Um dem Server zu gestatten mit den Clients interagieren zu können wird nun auf Input aus der Commandline gewartet. Der Server kann entweder Chatten, eine Liste mit allen verfügbaren Kommandos anzeigen, eine Liste der verbundenen Clients anzeigen oder den Server beenden.
Die Kommandoliste ist als String definiert und wird als solcher ausgegeben.
Die Clientliste wird durch Abrufen der String-Values (also der Client Namen) einer Map generiert.
Das Schließen des Servers wird durch das Senden der Nachricht "exit" an alle clients eingeleitet, welche diese schließt (genauer erläutert in **??**). In Folge werden alle Threads mittels .interrupt() beendet. Hierbei traten erste Probleme auf, da immer ein leerer Thread übrig blieb, dessen Ursprung ich nicht feststellen konnte. Da dieser jedoch wie bereits erwähnt leer erschien wurde das Programm an dieser Stelle mittels System.exit(0) terminiert.

In dem in Paragraph 1 Absatz 2 gestarteten Thread wird in einer Endlosschleife auf client-Sockets gewartet und diese jeweils einem Thread übergeben. Dieser ist als Subklasse des Servers, welche von Runnable, erbt umgesetzt.

In dem Thread werden je ein PrintWriter und BufferedReader erstellt und eine Endlosschleife erstellt, in welcher auf Nachrichten des clients gewartet wird.
Es wird auf eine erste Nachricht gewartet, in welcher der Clientname geschickt wird. Sollte dieser leer gelassen worden sein wird dem Client ein aus der Anzahl der bisher verbundenen clients generierter zugewiesen und zurückgeschickt. BufferedReader und Name werden in einer Hashmap gespeichert.
Folgende Nachrichten werden darauf überprüft ob sie "exit", "ls" oder eine Chatnachricht sind.
Bei "exit" werden alle Clients mittels einer Chatnachricht informiert, dass der Client sich vom Server getrennt hat (später genauer erläutert). Dem Client wird ebenfalls ein "exit" geschickt, sodass dieser, falls er sich nicht bereits geschlossen hat, seine Verbindung zum Server beendet. Der client-Socket wird geschlossen, der PrintWriter aus der Map entfernt und aus der endlosschleife ausgebrochen, was den Thread beendet.
Bei "ls" wird eine abgewandelte Form der Methode die der Server benutzt aufgerufen. Dieser wird der PrintWriter übergeben, sodass diese dem Client einen String mit einer Liste der verbundenen Clients übergeben kann.
Chatnachrichten werden an zwei Methoden übergeben. Eine schreibt diese in die Konsole des Servers, die zweite iteriert durch die Map der PrintWriter um die Nachricht an alle Clients in dieser zu schicken.

2 Client

Ein Object der Klasse Client wird erstellt. Über einen Methodenaufruf wird die GUI des Clients erstellt, eine im nächsten Paragraphen erläuterte Methode aufgerufen und ein neuer Thread von einer Subklasse gestartet (an späterer Stelle erläutert).

Der Client versucht über einen Socket eine Verbindung zum Server herzustellen. Die dafür benötigte Addresse bezieht er aus einem Config file um simplifiziertes Ändern des Hosts zu erlauben. Je ein BufferedReader und PrintWriter werden erzeugt.
Dem Client wird mitgeteilt ob dies erfolgreich stattgefunden hat. Solte es nicht erfolgreich stattgefunden haben wird das Programm bis auf die GUI beendet.
Sollte es erfolgreich stattgefunden haben wird der User gebeten einen Username einzugeben, welcher an den Server geschickt wird. Falls der Username leer ist wird dem Server stattdessen ein String in einer anderen Syntax geschickt und darauf gewartet, dass dieser einen automatisch generierten Username zurückschickt.
Zuletzt wird noch ein WindowListener erzeugt, welcher dem Server "exit" schickt sobald das Fenster geschlossen wird um den Client sauber vom Server zu trennen.

In dem zuvor erzeugten Thread der Subklasse wird in einer Endlosschleife auf Nachrichten des Servers gewartet, welche in Folge in der GUI einer TextArea angehengt werden.
Sollte jedoch ein "exit" ankommen wird dem User mitgeteilt dass der Server die Verbindung geschlossen hat und der Client bis auf die GUI beendet.

Der GUI werden zwei Listener zugewiesen. Sollte ein 'Enter' oder ein Druck auf den Button registriert werden, wird jene Nachricht die im TextField steht an den Server gesendet. Sollte diese mit einem "/" beginnen, wird die Nachricht jedoch escaped.
Bei "/exit" wird "exit" geschickt und somit die Verbindung beidseitig beendet.
Bei "/ls" wird "ls" geschickt, woraufhin der Server eine Liste mit allen verbundenen Clients zurückschickt.

3 Probleme

Es sind großteils nur menschliche Probleme aufgetreten. Hierbei ist vor allem beim Absenden der Nachricht über den PrintWriter oft vergessen worden die .flush() Methode aufzurufen, wodurch die Nachricht nicht versendet wurde.
Wie bereits in der Beschreibung des Servers erwähnt trat auch beim sauberen Schließen des Servers ein Problem auf, welches jedoch leicht lösbar war.